

A Network Engineer's Travelogue Deploying Domain Solutions



Juniper Networks Warrior

O'REILLY® JUNIPER
NETWORKS

Peter Southwick

Juniper Networks Warrior

In this practical book, Juniper Networks consulting senior network engineer, Peter Southwick, offers unique first-person field studies on designing, configuring, and troubleshooting new systems that are changing the networking world. Each chapter-long “travelogue” follows a team of Juniper Networks warriors as they solve specific problems with emerging network platform architectures.

In these case studies, Southwick and his fellow warriors analyze a client’s particular situation, arrive at an architectural solution, and work through the deployment details. For anyone who operates, installs, designs, or works in IT, this book provides an intimate and entertaining look at what’s changing and why.

Among the case studies, you’ll discover how:

- A service provider protected customers from malicious traffic with Juniper Networks IDP systems
- SRX5800s improved connectivity and security in a data center
- Ethernet WAN technology was chosen as a storage solution, rather than a proprietary design on dark fiber
- An enterprise severed communications between different departments to comply with government personal credit card standards
- Core network and edge devices helped a power company serve local customers and ISPs in the data services market
- A hosting company migrated its core, data center, edge, and access domains to a state-of-the-art network

“In this uniquely written book, you will get a detailed view of life in the data center, the edge, the core, and the office of the customer’s CIO.”

—Steve Fazio
CEO, TorreyPoint

Peter Southwick, a senior network engineer at TorreyPoint, provides professional services support and training. He is a JNCI, and holds JNCIE-M #473 and other Juniper certifications in routing and security. Peter is co-author of *Junos Enterprise Routing, Second Edition* (O’Reilly), *JNCIE-SP Exam Workbook* (Proteus Press), and *Telecommunications: A Beginner’s Guide* (McGraw Hill).

Part of the Juniper Networks Technical Library™

JUNIPER
NETWORKS

US \$59.99

CAN \$62.99

ISBN: 978-1-449-31663-1



Twitter: @oreillymedia
facebook.com/oreilly

O'REILLY®
oreilly.com

Juniper Networks Warrior

Peter Southwick

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Juniper Networks Warrior

by Peter Southwick

Copyright © 2013 Peter Southwick. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://my.safaribooksonline.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Editors: Mike Loukides and Meghan Blanchette

Production Editor: Melanie Yarbrough

Copyeditor: Rachel Head

Proofreader: Linley Dolby

Indexer: Fred Brown

Cover Designer: Karen Montgomery

Interior Designer: David Futato

Illustrator: Kara Ebrahim & Rebecca Demarest

November 2012: First Edition

Revision History for the First Edition:

2012-11-09 First release

See <http://oreilly.com/catalog/errata.csp?isbn=9781449316631> for release details.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. *Juniper Networks Warrior*, the cover image of a Seawolf, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc., was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-1-449-31663-1

[LSI]

This book is dedicated to the real warriors of this world who keep us free and sometimes die in the process. We salute and honor you.

Table of Contents

Preface.....	xi
1. An Enterprise VPN.....	1
Company Profile	2
Network	2
Traffic Flow	3
Need for Change	4
Class of Service	4
Design Trade-Offs	6
Implementation	10
Prototype Phase	10
Class of Service	18
Cut-Over	31
Main Site	32
Remote Site JAX	32
Remote Sites PHL and IAD	36
Backup Site BNA	37
Conclusions	37
2. Maintaining IDP Systems.....	39
IDP8200 Background	40
Command-Line Interface	40
Web Management Interface	43
NSM Management	45
Support Tasks	47
Daily Tasks	47
IDP Policies	54
Rulebase Optimization	58
Other Tasks	59

Conclusion	64
3. Data Center Security Design.....	67
Discussion	68
Design Trade-Offs	72
Decision	73
Configuration	75
Take One Configuration: Clustering	76
Take 2 Configuration: Active/Active without Reths	87
Take 3 Configuration: Active/Active with One-Legged Reths	88
Testing	89
Summary	90
4. Layer 3 to Layer 2 Conversion.....	93
Problem	96
Q-in-Q Framing	99
VPLS Overhead	99
Solutions	104
RFC 4623	104
Configurations	106
Management	108
Protocols	118
Core Router Configurations	123
Distribution Switch Configurations	129
Distribution Router Configurations	131
Rate Control	133
CPE Switch Configuration	134
Conclusion	134
5. Internet Access Redress.....	137
Objective	138
Design	140
Trade-offs	143
Configuration	147
Clustering	147
Security	150
Routing	159
Implementation	169
Lessons Learned	170
Conclusion	173
6. Service Provider Engagement.....	175

Company Profile	175
Physical Network Topology	176
Services	178
Design Approach	178
Design Trade-Offs	181
Configurations	184
Boilerplate Configuration	184
MX Interfaces	187
EX Boilerplate and Interfaces	193
OSPF	199
MBGP	201
MPLS	202
RSVP	204
Layer 3 VPN	207
VPLS	214
OBM	217
Conclusion	219
7. A PCI-Compliant Data Center.....	221
Introduction	221
Client Goals	222
Design Trade-Offs	224
Recommended Design	227
Switching Layer	227
Routing Layer	229
Firewall Layer	231
Virtualization	232
Configurations	233
EX4200 Configuration	233
MX240 Configuration	239
Firewall Configuration	245
Deployment	251
Initial Connectivity	251
The Maintenance Window	252
PCI Compliance	252
Summary	254
8. Facilitating Dark Fiber Replacement Using a QFX3500.....	255
Existing Design	255
Introduction to Fibre Channel	257
Proposed Design	259
Concerns and Resolutions	259

Network Upgrade	261
Advantages and Benefits of the Solution	263
QFX3500 Fibre Channel Gateway Configurations	264
Management Configurations	264
Fibre Channel Gateway Interface Configuration	270
DCB Configuration	272
EX4500 Transit Switch Configurations	276
Interfaces and VLANs	276
Transit Switch DCB Configuration	279
Verification	282
Conclusions	285
9. MX Network Deployment.....	287
Plans and Topology	288
Phase 1	289
MX Configuration	291
Management Configuration	291
Routing Engine Protection	293
Policy Configurations	303
Protocol Configurations	311
Phase 2	315
Final Phases	320
Conclusion	320
10. A Survivable Internet Solution for a Fully Distributed Network.....	321
Original Network Architecture	321
WAN Connectivity	322
Addressing	323
Internal Connectivity	323
Firewalls	324
Problem Definition	325
Proposed Solution 1	327
Solution 1 Advantages	329
Solution 1 Details	329
Solution 1 Issues	330
Proposed Solution 2: OSPF over Tunnels	330
Early Death of Solution 2	332
Configuration for Solution 2	332
Final Solution: Static Routes over Tunnels	333
Solution Advantages	334
Solution Issues	335
Email Server Address Resolution	340

Firewall Configurations	342
Conclusion	354
11. Internet Access Rebuild.....	357
Requirements	358
Existing Network	358
Routing Protocols	359
Solution Options	363
Three-Layer Design	363
Two-Layer Design	365
One-Tier Design	367
Configurations	372
Deployment Scenario	372
Management Staging and Testing	373
Top-of-Rack Switch Testing	377
ISP Link Testing	383
Production Configuration	391
Cut-Over	396
Conclusion	397
Index.....	399

Preface

The network has changed a lot recently, with 10 years' worth of developments packed into just 2 or 3. Those changes have been in specific network domains. The industry has grown out of the “just put another rack in” approach, because putting another rack in does not necessarily equate to gaining more bandwidth or more services or more security. Patching your limping network with a new box *will give you a faster limping network*.

The rise of systemic networking has in turn given rise to the Juniper Networks warrior. While it's not a given that they know more than or are better than other vendors' professional installers, Juniper Networks warriors think in terms of network platforms and how the entire architecture works for the client. They think in terms of extra capacity in the near future and long-term scalability for the client. They also think in terms of domains: the needs for the service provider edge are different than those of a campus or branch network, but both might use the MX480. For a Juniper Networks warrior, the deployment adapts to the domain rather than the domain bending to accommodate what the deployment can't do.

An explosion of system-wide architectures and network deployments has occurred in the past five years, and I have seen it happen firsthand as a professional services networking engineer (and trainer). I am one of many, and I have encountered both warriors who are umpteen times smarter than I, and others who I have had to drag along by the scruff of the neck. Our numbers are growing.

This book presents a series of network engineer's travelogues that I hope will entertain and illuminate—they show specific configurations in this new world, where a systemic approach is actually cheaper, easier, and better than squeezing in another rack.

More specifically, I hope these chapter-length travelogues will show our warriors' ability to think on our feet, because no two networks are the same even if they fall in the same domain. A common warrior's morning lament is “OMG, how are we going to fix that!?”

But then we put on our shoes and walk into the meeting room and figure it out somehow. And we do it every day, every week, at almost every deployment. As the saying goes, *sometimes you get the bear and sometimes the bear gets you*. Thankfully, the bear does not win very often, and we're still here, gettin' the bear.

In most engagements, the equipment has been ordered, the sales deal is done, the media has over hyped the issues, everyone wants new networking power, the deadline is looming, and the politics of the client are, well, very visible. You fly in like a smoking gun, meet with a half dozen other warriors—some you know, some you don't—and you are expected to perform like a well-oiled machine for the next week or month, cooped up together, sleeping and eating like a band of foot soldiers. What you do has to be flawless, meticulous, speedy, and mindful of the whims of the client.

As the world favors these platform architectures more and more, the network warrior must perform on a systemic stage. Hats off to you, my fellow network warriors. It's showtime!

What Is the New Network Platform Architecture?

Once upon a time, it used to be just the service provider (SP) and the local area network (LAN). Then it went to SP and enterprise. Then campus and branch, WLAN, and edge joined in. Then the data center, and now user devices by the billion, with each having more communication power than any computer a decade ago. This evolution is a good thing. It means the domains of the world's networks are adapting to the needs of their entities, and they are organizing themselves by how they operate and the services they need to offer to their users. Putting another router on the rack because its cheap ain't going to cut it, because you'll eventually need more warriors and more warrior time to fix the cheap patch.

This book endorses Juniper's *New Network Platform Architecture* approach, if only because I have been installing it for years under different names, and it works. This approach is at the heart of each chapter's deployment. Any warrior worth his salt should be giddy to see such an emphasis on this platform and what that future offers.

This book darts around the domains in a random fashion because their order is not important, but I call them out at the beginning of each new chapter. This book is about network engineers and the problems and challenges they face when they deploy networks to help people communicate and share. Layer 8 of the OSI model of networking, or *politics*, is alluded to in several chapters, but I try to avoid going into gory detail about the political battles witnessed during the deployments (most warriors would rather be confronted by a downed network than two clients giving them separate and contrasting instructions—the network they can fix, while the other problems just seem to fester).

How to Use This Book

Let's look at some specifics on how this book can help you. Every network deployment is different, like trees, like snowflakes, like people. You have to have an open mind, use open standards, and be as meticulous as a warrior. My fellow warriors will enjoy these chapters as pure networking travelogues: they might remind you of that build-out in the Midwest during the Great Blizzard, or those crazy people at University X. For others, who are aspiring to be warriors, or perhaps are part of the warriors' sales and support teams, you need to know the process that happens onsite to make it all work. Upon reflection, however, I think that the people who actually spend the money and buy new networking equipment may benefit the most from this book. The warrior tribe sent to your location can work wonders if you listen and participate.

Different readers will use this book for different reasons, so each might use a different part of each chapter for their purposes. Each chapter starts off with an analysis of the client's situation and how the power of the Juniper Networks domains concept can be harnessed to improve that situation. In this portion of the chapter, the trade-offs are weighed, the requirements are outlined, and the solution's architecture is shown. The second part of each chapter gets into the nuts and bolts of how the solution was crafted. I realize that many concepts are used in most engagements, so some of the details might be skipped. But for the most part, the configuration snippets are all usable as presented. Most chapters end with the steps used by the tribe to install, migrate, or activate the client's network. If you are reading this to understand what devices we use in what environments and why, you might want to skip the gory details. If you are reading this as a means to solve your client's issues, you might skip the political section. All in all, there are many ways to use this book; my hope is that whatever your goals, you find it helpful and enjoyable.

I assume a certain level of networking knowledge on the reader's part. The less you know about the following concepts, the more each chapter will get fuzzy just when it gets down to warrior-dom:

The OSI model

The Open Systems Interconnection (OSI) model defines seven different layers of technology: the physical, data link, network, transport, session, presentation, and application layers. This model allows network engineers and network vendors to easily discuss different technologies and apply them to specific OSI levels, and allows engineers to divide the overall problem of getting one application to talk to another into discrete parts and more manageable sections. Each level has certain attributes that describe it, and each level interacts with its neighboring levels in a very well-defined manner. Knowledge of the layers above Layer 7 is not mandatory, but understanding that interoperability is not always about electrons and photons will help.

Switches

These devices operate at Layer 2 of the OSI model and use logical local addressing to move frames across a network. Devices in this category include Ethernet in all its variations, virtual LANs (VLANs), aggregate switches, and redundant switches.

Routers

These devices operate at Layer 3 of the OSI model and connect IP subnets to each other. Routers move packets across a network in a hop-by-hop fashion.

Ethernet

These broadcast domains connect multiple hosts together on a common infrastructure. Hosts communicate with each other using Layer 2 media access control (MAC) addresses.

IP addressing and subnetting

Hosts using IP to communicate with each other use 32-bit addresses. Humans often use a dotted decimal format to represent these addresses. This address notation includes a network portion and a host portion, which is normally displayed as 192.168.1.1/24.

TCP and UDP

These Layer 4 protocols define methods for communicating between hosts. The Transmission Control Protocol (TCP) provides for connection-oriented communications, whereas the User Datagram Protocol (UDP) uses a connectionless paradigm. Other benefits of using TCP include flow control, windowing/buffering, and explicit acknowledgments.

ICMP

Network engineers use this protocol to troubleshoot and operate a network, as it is the core protocol used (on some platforms) by the *ping* and *tracert* programs. In addition, the Internet Control Message Protocol (ICMP) is used to send error and other messages between hosts in an IP-based network.

Junos CLI

The command-line interface (CLI) used by Juniper Networks routers is the primary method for configuring, managing, and troubleshooting the routers. Junos documentation covers the CLI in detail, and it is freely available on the [Juniper Networks website](#). The Juniper Day One Library offers [free PDF books](#) that explore the Junos CLI step by step.

What's in This Book?

The unique advantage of Juniper Networks warriors is that they tend to think in terms of complete systems rather than adding on boxes here and there. It's a different switch you must throw in your head, but soon after, you'll start thinking in terms of network domains.

Here's what we warriors were up to at the deployments covered in this book:

Chapter 1

This New England engagement looks at a branch office domain implementation using Juniper Networks J-series and MX routers connecting to a provider-provisioned Layer 3 virtual private network (L3VPN). The client's requirements included alternate paths survivability and class of service guarantees for traffic.

Chapter 2

Most service providers are seeing the need to protect their customers from malicious traffic and attacks. The security of customers is the pervasive thread across all domains. This chapter looks at the tasks and capabilities used to ensure that Juniper Networks intrusion detection and prevention (IDP) systems are kept in optimal operating condition to assure that security.

Chapter 3

The data center domain is the home for low-latency switches and high-availability servers. With the critical nature of the data in these data centers, securing the communications is as important as getting it to the destination, and in some cases more so. This chapter looks at the deployment of SRX5800s in the heart of a data center—not only improving connectivity at low latency, but also securing that information.

Chapter 4

This Alaska-based engagement takes a new look at the WAN domain: an existing routed network of M-series and MX routers is reused to offer Ethernet services in the far north. For the readers that have not looked at Ethernet as a WAN technology, this chapter offers a deep dive into the frames, packets, and MTUs of this new entry into an old domain. It was a lesson for me, so I present it to you.

Chapter 5

The Internet edge domain can be a single router or a multitude of firewalls and security apparatus. In this engagement the multitude was replaced by the singular. This chapter details the migration of a fully distributed Internet egress system to a manageable SRX-based design.

Chapter 6

This chapter looks at an engagement that took place in my home state of Vermont. This service provider engagement offered a chance to work in the core domain and the edge domain, standing up new services for a traditional telephone company.

Chapter 7

This Eastern Seaboard engagement took on the government compliance guidelines for personal credit card information and the securing of the same. Oh, how I love regulations: the client needed to assure that different departments of the same enterprise could not talk to one another, so as to comply with the government standards. We used SRXs to secure communications to provide compliance.

Chapter 8

This chapter takes a different look at the WAN domain. This New Jersey engagement allowed a customer to realize operating expenditure savings by using an Ethernet WAN technology for a storage solution rather than a proprietary design on dark fiber.

Chapter 9

This engagement was based on the shores of the mighty Mississippi, where a power company was entering into the data services market. The use of MXs allowed the provider to deploy a core network as well as edge devices to serve both local customers and ISPs in the area.

Chapter 10

Most of the engagements presented in this book are based on the new Juniper Networks platforms, but not all engagements are based solely on these products. In this engagement, Netscreen-based firewalls were used to meet the requirements of a distributed network in New England. It looks at a secure and survivable core domain.

Chapter 11

The last chapter is a look at a Northeast hosting company that was migrating its core, data center, edge, and access domains into the current decade. This engagement explores the options, the trade-offs, and the migration to a state-of-the-art network.

A Note About This Book

This book is created from my notes and files on various clients over the past four years. The chapters have been sanitized to protect the clients and their networks. All addressing, AS numbers, and locations are made up. The configurations are functional but do not match the actual client devices. In some cases, the chapter is a mashup of multiple engagements.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, file extensions, pathnames, directories, commands, options, switches, variables, attributes, and Unix utilities

Constant width

Indicates the contents of files and the output from commands

Constant width bold

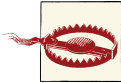
Shows commands and other text that should be typed literally by the user, as well as important lines of code

Constant width italic

Shows text that should be replaced with user-supplied values



This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your own configuration and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the material. For example, deploying a network based on actual configurations from this book does not require permission. Selling or distributing a CD-ROM of examples from this book does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of sample configurations or operational output from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN, for example: "*Juniper Networks Warrior*, by Peter Southwick. Copyright 2013 Peter Southwick, 978-1-449-31663-1."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at permissions@oreilly.com.

Safari® Books Online



Safari Books Online (www.safaribooksonline.com) is an on-demand digital library that delivers expert **content** in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of **product mixes** and pricing programs for **organizations**, **government agencies**, and **individuals**. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens **more**. For more information about Safari Books Online, please visit us **online**.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at http://oreil.ly/juniper_networks_warrior or <http://cubednetworks.com>.

To comment or ask technical questions about this book, send email to bookquestions@oreilly.com.

For more information about our books, courses, conferences, and news, see our website at <http://www.oreilly.com>.

Find us on Facebook: <http://facebook.com/oreilly>.

Follow us on Twitter: <http://twitter.com/oreillymedia>.

Watch us on YouTube: <http://www.youtube.com/oreillymedia>.

Acknowledgments

I alone put pen to paper and recorded these engagements, but I realize that I am a member of a very close-knit tribe. As such, I cannot take credit for all the thoughts, the knowledge, or the total experience that this book represents. There are many groups and individuals that I have to acknowledge as contributors to this endeavor.

First, the tribe. I would like to acknowledge, with great praise and as much fanfare as this single paragraph can raise, all the members and professionals of the global tribe of networking warriors. I am honored to be a part of your profession and the world's network deployment, and all that it means to our society and our diverse cultures.

During my engagements over the past years I have met and worked with many warriors. They are all a part of this book, and without an acknowledgment to them this book would not be complete. Because of reasons involving lawyers, I cannot identify each of these warriors by name and company. It will have to suffice to give thanks to Curtis, Corey, Adam, Eddie, John, Steve, Bill, Cliff, and Joe. Each of you may recognize an idea or a concept that we talked about when I was working with you.

Since we are network engineers, we do fall into the stereotypes of that group. All of this book's editorial cleanup, formatting, and graphical conformity has been performed by people not listed as authors or technical reviewers. It is they who deserve the accolades. I acknowledge them as the true wizards and warriors of the written word: I am grateful to Mike Loukides, Senior O'Reilly Editor, and Meghan Blanchette, Editor, who never let my sporadic schedule and warrior life worry them. Their technical expertise and attention to detail made this book better. I would also like to thank Rachel Head for her copyediting, and Kara Ebrahim and Rebecca Demarest for their artwork; their contributions have made this a better experience for you, the reader.

Patrick Ames has been the guiding light for this project. Thank you for your ideas, editorial help, patience, and eagle eye for detail. Your persistence and enthusiasm have made this project both possible and enjoyable. You gambled on a wild idea that has come to fruition. From the initial phone calls to the final edits, you have been there as the shining beacon showing the way to the home port. Thank you!

I would like to acknowledge the contributions of Juniper Networks in general, for the assistance provided on various fronts.

I also want to acknowledge my fellow warriors of TorreyPoint and Proteus Networks. You have taught me more than any class or seminar—your passion for the technology and dedication to the customer are goals that we all strive for.

When I was last published, I gave thanks to my family for allowing me to create the book. I was new to this writing stuff and at that point, the thanks seemed a little narrow minded. I am forever grateful to my family for allowing me to continue with this vocation (yes, it is!). They welcome me home with open arms week after week, put up with missed

meets, parties, and holidays, and allow me to spend evenings at home playing in my lab. To say that this book would not have been possible without their support would be an understatement; this book would not have been conceivable without them. You are my inspiration and my reason. Michele, Gabby, and Tori, thank you for being my family, friends, and partners in all that I do.

An Enterprise VPN

This book describes the jobs that I and other networking engineers have performed on client networks over the past few years. We are considered *network warriors* because of the way that we attack networking challenges and solve issues for our clients. Network warriors come from different backgrounds, including service provider routing, security, and the enterprise. They are experts on many different types of equipment: Cisco, Checkpoint, and Extreme, to name a few. A warrior may be a member of the client's networking staff, drafted in for a period of time to be part of the solution, but more often than not, the warriors are transient engineers brought into the client's location.

This book offers a glimpse into the workings of a Juniper Networks warrior. We work in tribes, groups of aligned warriors working with a client toward a set of common goals. Typically technical, commonly political, and almost always economical, these goals are our guides and our measures of success.



To help you get the picture, a quote from the 1970 movie *M*A*S*H* is just about right for us network warriors: “We are the Pros from Dover and we figure to crack this kid's chest and get off to the golf course before it gets dark.” Well, not really, but the sentiment is there. We are here to get the job done!

Over the past four years, I have been privileged to team with talented network engineers in a large number of engagements, using a tribal approach to problem solving and design implementation. It is a treat to witness when multiple network warriors put their heads together for a client. But alas, in some cases it's not possible to muster a team, either due to financial constraints, complexity, or timing, and the “tribe” for the engagement ends up being just you. Such was the case for the first domain we'll look at in this book, deploying a corporate VPN.

While I used Juniper Networks design resources for this engagement, there were no other technical team members actively engaged, and I resigned myself to do this job as a tribe of one (although with backup support only a phone call away—don't you just love the promise of JTAC if needed?).

The project came to Proteus Networks (my employer) from a small value added reseller (VAR) based in New England. “We just sold a half-dozen small Juniper Networks routers and the buyers need some help getting up and running,” I thrive on such a detailed statement of work. After a phone call to the VAR, and a couple of calls to the customer, I was able to determine the requirements for this lonely engagement.

Company Profile

The company is an enterprise with five locations in the Eastern US ([Figure 1-1](#)). The headquarters are located in Hartford, CT (BDL). This location houses the management offices, the accounting and HR departments, the primary data center, and warehouse facilities. A backup data center is located in the Nashville, TN (BNA) area in a leased facility. There are three other warehouses scattered down the Eastern Seaboard, with the southernmost being in Florida.

The company has a CEO who is a techie (he was a Coast Guard radio technician, the kind that can make a radio with nothing more than a soldering iron and a handful of sand). He has kept up with developments in CRM (customer relationship management), inventory control, and web sales. He has grown his company to be a leader in his industry segment by being able to predict when his customers are going to need his product, often before the customers themselves know it.

Network

Prior to the upgrade, the company was running on an Internet-based wide area network. All sites were connected to the Internet and had IPSec tunnels back to the headquarters, creating a virtual private network. The sites have DSL Internet connections from the local ISPs. Each location has a simple LAN/firewall network using static routes to send traffic to the Internet or the main location. At the main location, a series of static routes parse the traffic to its destinations.

In 2010, the company created a disaster recovery and business continuity site in Nashville. The original connectivity between the primary servers in Hartford and the backups in Nashville was a private line service running at 1.5 Mbps.

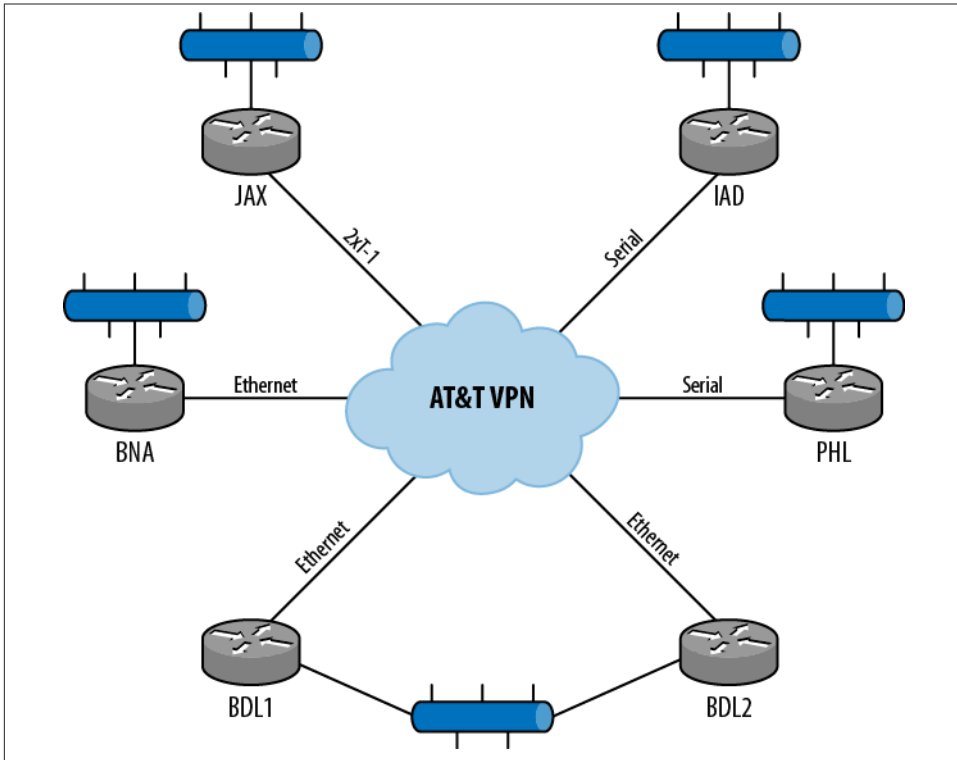


Figure 1-1. Company locations and the new network

Traffic Flow

Local inventory control servers in each of the remote locations are updated to the main servers every evening using the IPsec VPN connection over the Internet. All sales transactions are performed over the Internet, either by customers on the web pages or by sales staff over a web portal. The web servers and other backend functions are performed at the main server location (Hartford).

The Hartford servers continually update the backup servers in Nashville. In the event of a failure of the Hartford servers, the traffic is directed to the Nashville servers. This changeover is currently a manual process.

Voice communication is provided by cellular service at all locations. A virtual PBX in the main location forwards incoming customer calls to the sales forces. Each employee has a smartphone, tablet, or laptop for instant messaging and email access.

Need for Change

The CEO realized that the use of nonsecured (reliability-wise) facilities for the core business functions would sooner or later cause an issue with the company. To avert a disaster, and to add services, he decided to install a provider-provisioned VPN (PPVPN) between all the sites. Each site would operate independently, as before, for Internet access and voice service, but all interoffice communications would now be handled by the PPVPN rather than the Internet.

This change would also allow the CEO to offer a series of how-to seminars to the customer base. The videos would be shot on location and uploaded to the main location for post-production work. They would be offered on the website and distributed in DVD format to the retail stores.

After talking to a number of service providers, the CEO settled on AT&T's VPN service. It offered connectivity options that made sense for the locations and the bandwidth required from each location. An option with the VPN service is the class of service differentiation that can prioritize traffic as it passes over the infrastructure. The CEO thought that this might come in handy for the different traffic types found on the internal network.

The company looked at managed routers from AT&T and compared the price to the purchasing of new equipment. They decided on buying Juniper Networks routers for all the locations. They bought MX10s for the Hartford and Nashville locations, to take advantage of the growth opportunity and the Ethernet interfaces for the VPN. At the remote locations with private line connectivity to AT&T, POP was more economical than Ethernet, so an older J-series router (J2320) was chosen for the availability of the serial cards (T1 and RS-422).

What the CEO required for support was configurations for each of the routers that interconnected the existing LANs at each of the locations and offered a class of service for the different traffic types. He also wanted assistance during the installation of the routers at all the locations.

Class of Service

AT&T offers a variety of profiles for customers that wish to add class of service to their VPN connectivity. AT&T provides customers with a **class of service planning document** and a **worksheet** to be filled out if that customer will be using AT&T's managed router service with the VPN. In this case, the CEO decided that the Multimedia Standard Profile #110 (reference **Table 1-1**) made the most sense for the company. The description of that profile from the AT&T planning document is as follows.

Multimedia Standard Profiles

Profiles in this category are recommended for high-speed connections or if the bandwidth demands of Real-time applications is small. Currently, the Multimedia profiles with Real-time bandwidth allocation are only available on private leased line access of 768K or greater. Ideal candidates are Branch sites or Remote locations that require Real-time as well as other application access. The maximum bandwidth allocated for the Real-time class is reserved but can be shared among non Real-time traffic classes in the configured proportions.

Table 1-1. Subset of AT&T CoS profiles

Traffic class	Profile #108	Profile #109	Profile #110	Profile #111	Profile #112	Profile #113	Profile #114	Profile #115	Profile #116
CoS1 (real-time)	50%	40%	40%	40%	20%	20%	20%	10%	10%
CoS2 (critical data)	0%	48%	36%	24%	64%	48%	32%	54%	54%
CoS3 (business data)	0%	6%	18%	18%	8%	24%	24%	27%	27%
CoS4 (standard data)	50%	6%	6%	18%	8%	8%	24%	9%	9%

The CEO defined four classes of traffic that sort of fit into the AT&T classes (see the design trade-offs below for the mapping and parameters). They are:

Multimedia traffic

The how-to videos, training seminars, and company-wide video meetings are all classified as multimedia traffic. This traffic, while not always present, represents the largest bandwidth hog.

Inventory control and CRM traffic

This traffic represents a constant background of traffic on the network. Traffic is generated as customers order supplies, as inventory control tracks retail floats, and as materials are received and shipped. While this traffic amounts only to a few kilobits per second overall, it is the most important traffic on the network as far as business success is concerned.

Office automation traffic

This is the normal email, IM, file transfer, and remote printing traffic that is seen in any office. This traffic is the lowest in bandwidth and has the lowest priority.

Internet traffic

While each remote site maintains an Internet presence, the servers in the Hartford (or Nashville) location process all supply searches, orders, and queries. This traffic is backhauled from the remote sites to the main servers for processing, and then

returned to the Internet at the remote locations. This traffic is a growing stream that is the future of the company. An effort in inventory control will give each member of the sales teams and the delivery force smartphones that can scan supplies and query the inventory server to locate the nearest item.

Design Trade-Offs

The design trade-offs for this project fell into three categories: routing between sites, class of service categories, and survivability. The first and the last are interrelated, so they are covered first.

Routing and survivability

The legacy network used static routes at the remote sites to connect users to the Internet and the IPSec VPN tunnels to the main location. The main site has static routes to each of the other locations. All remote locations have network address translation (NAT) for all outgoing traffic. All incoming Internet traffic that arrives at the main location is NATed to a private address and forwarded to the appropriate server.

The trade-off here is one of simplicity versus reliability. The existing system was created so that no knowledge was needed to set up the devices and have them operate. All traffic either went to the Internet or the IPSec tunnel. Once the VPN is added to the network, the simplicity of the single-path network disappears.

The installation of the VPN allows a secondary path for traffic to the Internet as well as from the Internet to the remote sites. The existing static routes could be retained and additional static routing could be used for the new equipment, but this approach requires knowledge of the possible routing outcomes, metrics, bandwidths, and outages. In the event of a failure, this knowledge would be crucial to determining the cause of the outage and getting the traffic back up and running.

The use of a dynamic routing protocol would allow redundancy and best-path routing without the need of a knowledgeable overseeing eye. It would provide a simple configuration for the routers while supporting survivability in the network. The routing protocol of choice is *open shortest path first* (OSPF) between the router and the firewall in each site.

The use of OSPF could have negative effects as well. Some implementations of IPSec tunnels (for example, policy-based tunnels in SSG devices) cannot support routing protocols. Also, the interface between customer edge (CE) routers and AT&T's VPN does not support OSPF (it supports only BGP or static routes).

A word about the technical staffing is required at this point. The company has *no technical networking staff*. It employs a software person to maintain the servers and keep the programs going, but there is no network person on staff. The CEO has made most of

the previous configurations and arrangements, with help from the equipment vendors' sales engineers. In light of this arrangement, anything that I set up has to be very easy to understand, troubleshoot, and fix. Fortunately, the remote office domain of Juniper Networks allows for hands-off operation, troubleshooting, and maintenance.

Remote locations. The decision was made to use both approaches. At the remote locations, including the new data center, static routing is used to reduce the complexity while still providing redundancy. The remote locations will retain their existing IPsec tunnels and ISP connections and add the VPN connection. The primary route to the main location is via the VPN for corporate traffic, with the IPsec tunnel as backup. Internet traffic will use an opposite approach: the local ISP connection is the primary route, with the backup being provided by the VPN. Simple route metrics (costs) are used to create the primary secondary relationship. The routes for the remote locations look like those in [Figure 1-2](#).

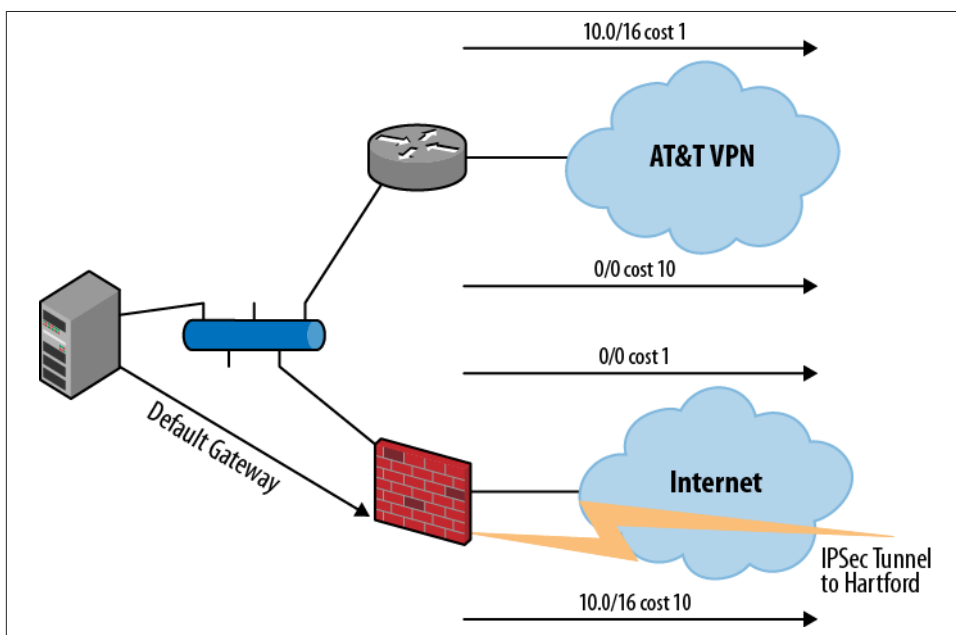


Figure 1-2. Remote location routing

The impact on the existing firewall is minimal. An additional static route is added to the firewall, identifying the location of the alternate route to the main location via the VPN router with a preferred metric. The existing static route to the main location pointing to the IPsec tunnel is modified to have a higher metric (less preferred).

Main location. Due to the number of remote locations and the possibility of adding new locations, it was decided to use OSPF in the main location. This would allow the use of the IPsec tunnels as floating statics and the dynamic learning of the remote locations from the VPN CE router. While the main location firewall needed more changes than the remotes, the firewall was up to the task. This decision increased the survivability of the network and decreased the need for changes when more locations were added. It also put the complexity where the intelligence is located in the network (close to the CEO).

The existing static routes to the remote locations were modified to have a higher administrative distance (AD)—yes, they were Cisco devices; the change would have been a route preference for Junos. OSPF was activated on the firewall and the static routes were redistributed into OSPF with a high metric. On the MX10, the same operation was performed, but this time the BGP routes from the AT&T VPN were redistributed into OSPF with a lower metric than the Cisco static routes. The arrangement required that each of the VPN tunnel's addresses be a passive interface in OSPF (that way, the MX10 can use them); the same is true for the interface to the AT&T network. The last issue is that on the MX10, the route preference of BGP is higher than that of the external OSPF routes (redistributed statics). In order for the router to choose the AT&T VPN for outgoing traffic, the OSPF external route preference must be raised to above that of BGP (170).

The static route to the Internet is also redistributed to OSPF and offered to the AT&T VPN. This allows each of the remote locations to use this route as an alternative to the local Internet connectivity.



The CEO recognized the critical nature of the local Internet access at the main location, but at this point, due to DNS and hosting issues, a true hot standby in Nashville was not possible. The development of this capability was left for another time and another engagement.

The routing arrangement at the main location is shown in [Figure 1-3](#).

Class of service

The issues for class of service center on how to assign the classes to the traffic types. The class definitions from the AT&T planning guide are:

COS1

Is for real-time traffic (voice and video) that is given the highest priority in the network. It is guaranteed the lowest latency and the assigned bandwidth. If COS1 traffic does not run at the assigned levels, then other classes can use that bandwidth. The network discards traffic above the assigned bandwidth levels.

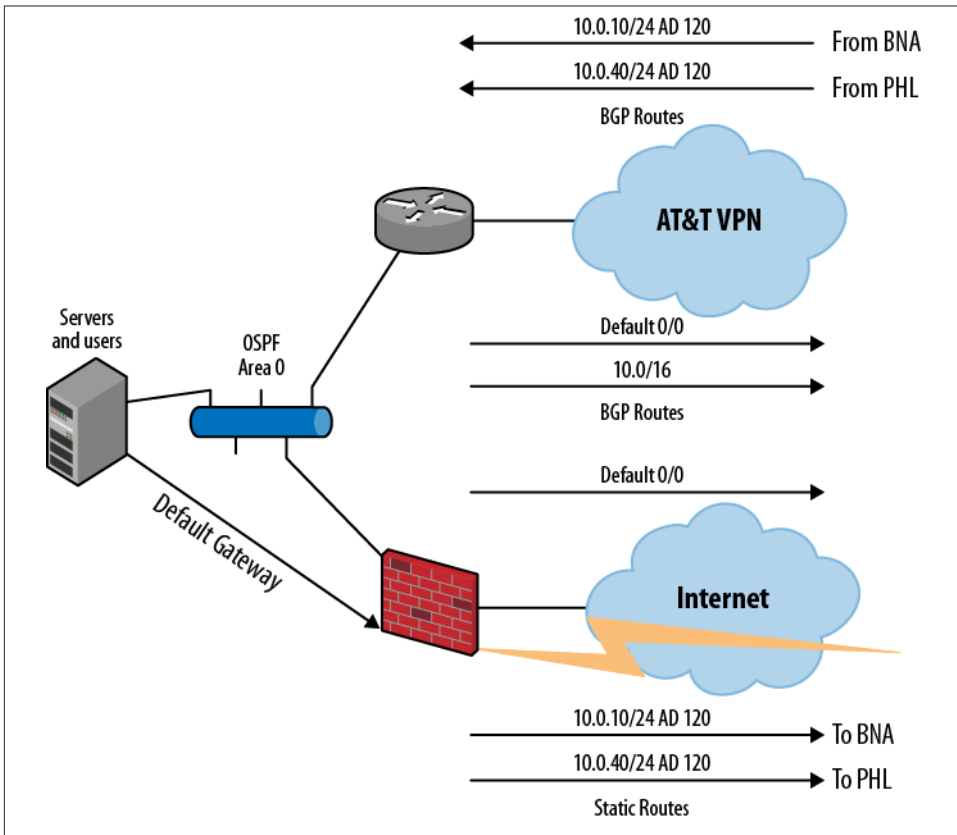


Figure 1-3. Main location routing

COS2

Is for mission-critical data traffic. This traffic is given the next highest priority and guarantee. Traffic above the COS2 level is passed through the network at the non-compliant level.

COS3

Is for office automation traffic. This traffic is offered a guarantee for delivery at the assigned rates. Traffic over these rates is passed as noncompliant traffic.

COS4

This is for best-effort traffic. If class of service is not enabled, all traffic flows are handled as best-effort traffic.

Each of the traffic classes described by the CEO had to be mapped to these AT&T classes, and definitive descriptors had to be defined to create the filters to identify the traffic.

The multimedia traffic was a good match for the COS1 class. This traffic originated from a known set of servers at a known bandwidth (the codec for the MPEG4 format runs at 768 kbps). The CEO agreed that there would not be a need to have multiple streams running at a single time.

The CRM and inventory traffic had many sources but a very limited set of servers at the main (or backup) location. The applications used a number of ports and could burst traffic during backups and inventory reconciliation between warehouses. To meet the demands of the company, 100 kbps was stated as the expected flow rate for this traffic. This traffic was to be mapped to the COS2 AT&T traffic class.

Web traffic (*http* and *https*) has such a core responsibility in the company that it was given the COS3 level of priority. This traffic can be limited to 500 kbps for any link.

The COS4 traffic was the office automation traffic that remained in the network. This traffic class was the default traffic class for all additional traffic as well and would not be policed or shaped in any fashion. It also received a minimal bandwidth guarantee.



You might have noticed that the traffic mapping is not as expected for this company—the priorities of the web traffic and office automation traffic are swapped. The Juniper Networks warrior needs to always listen to the customer, and change from the normal whenever needed. When working directly with a client, making assumptions can often have bad impacts on the company's operation.

These levels are all well below the AT&T profile for the lowest-bandwidth interface (Jacksonville at 3 Mbps). Setting the router classes to match the customer's specification assures that the network will not change the traffic priorities.

Implementation

Once the high-level design was created, reviewed, and approved by the CEO, the actual implementation of the new network was undertaken. The plan used a three-phase approach. In the first phase, a prototype network was created that verified the operation of the design. In the second phase, the equipment was installed onsite and interconnected. In the final phase, the new network was cut into the existing network.

Prototype Phase

All the Juniper Networks equipment was delivered to the main location, unpacked, and powered up in a lab environment. With the use of routing instances, all the devices of the network were created and interconnected. The J-series routers are equipped with flow-based services and full stateful security services, so these were configured as the

firewalls as well as the local routers (all in one box). The router for the backup servers (Nashville) was divided into the local router, the Internet, and the AT&T VPN. Finally, the main routers were configured as themselves. One of the remote locations was repurposed to act as the main location firewall.

This configuration was interconnected with Ethernet links rather than T1 and serial links, but other than that, all the other configuration aspects could be verified. The network diagram looked like that shown in [Figure 1-4](#).

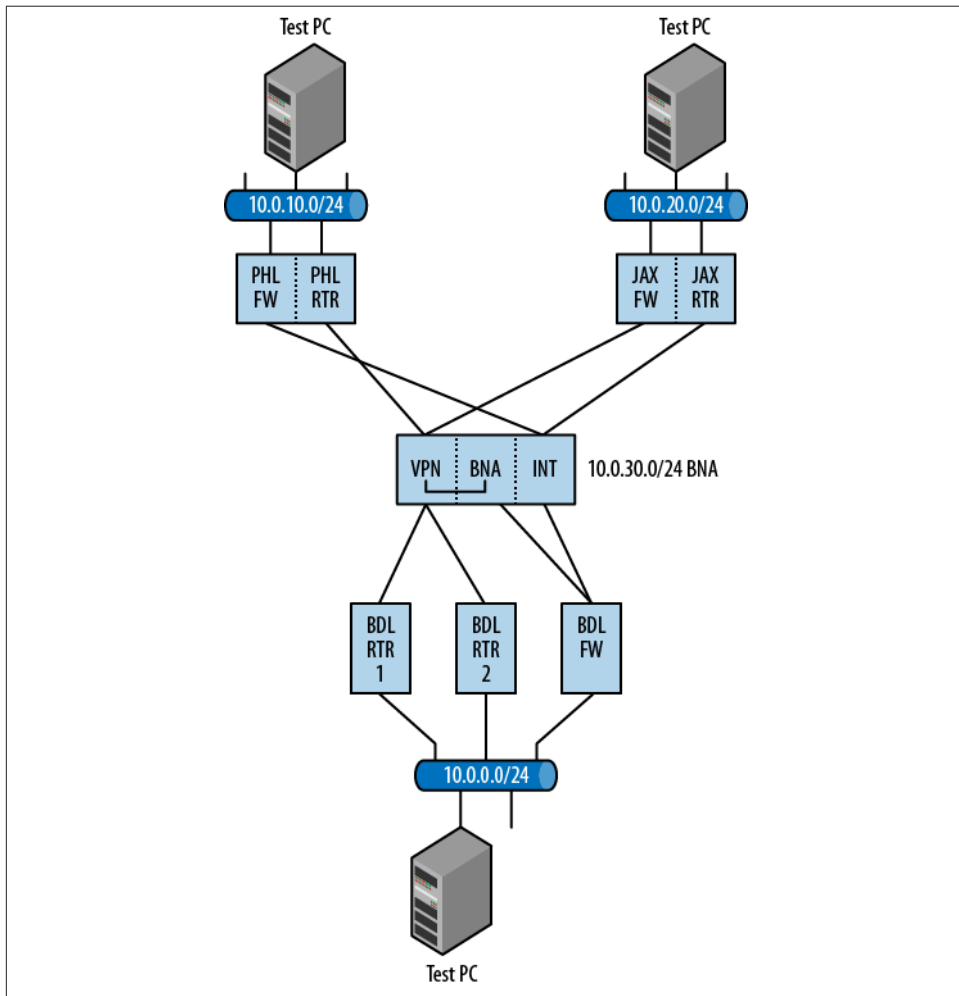


Figure 1-4. Prototype network

The configuration of the Juniper Networks routers for the firewalls with the IPSec tunnels was taking liberties when compared to the existing firewalls, but for the purposes of the prototype testing, it was acceptable. I did determine a few things that made me wonder, though:

- The IPSec tunnels had to be configured from the default routing instance. They could be set up in a routing instance, but I could not get them to come up and pass traffic except in the default routing instance.
- The four Ethernet ports on the J2320s were great for aggregating traffic to another device. A single port can act as both the VPN and the Internet ports. This saved cabling and trying to set up the T1 port adapter.
- The BNA to VPN connection had to be created internally in the device, as I had used up all the external ports. A filter with a next-table entry and a static route worked just fine.

The relevant configuration of one of the remote locations was:

```
[edit]
lab@RIGHT_TEST# show
interfaces {
    ge-0/0/0 {
        description "JAX-FW to LAN";
        unit 0 {
            family inet {
                address 10.0.10.2/24;
            }
        }
    }
    ge-0/0/1 {
        vlan-tagging;
        unit 0 {
            description "JAX-RTR to VPN";
            vlan-id 100;
            family inet {
                address 101.121.10.2/24;
            }
        }
        unit 1 {
            description "JAX-FW to Internet";
            vlan-id 101;
            family inet {
                address 101.121.20.2/24;
            }
        }
    }
    ge-0/0/2 {
        description "JAX-RTR to LAN";
        unit 0 {
            family inet {
```

```

        address 10.0.10.1/24;
    }
}
st0 {
    description "Tunnel to BDL";
    unit 0 {
        family inet {
            address 10.111.2.1/24;
        }
    }
}
}

```

While the interface addressing has been altered to preserve privacy, the descriptions of the interfaces give you an indication of their function in the test bed. The static routes used a qualified next hop for the secondary routes. These routes are the firewall routes that are shown in [Figure 1-3](#):

```

routing-options {
    static {
        route 10.0.0.0/16 {
            next-hop 10.0.10.1;
            qualified-next-hop 10.111.2.2 {
                metric 100;
            }
        }
        route 0.0.0.0/0 {
            next-hop 101.121.20.1;
            qualified-next-hop 10.0.10.1 {
                metric 100;
            }
        }
    }
}

```

The security stanza was straightforward: a route-based IPSec tunnel with standard proposals connects to the main location. Because the J-series is either a router or a firewall for all interfaces, the router side has zones and security policies just like the firewall interfaces. For the purposes of the tests, all traffic was permitted and no NAT was performed. From a routing perspective, this does not alter the testing. The policies and the host-inbound services have been deleted from the configuration to save space:

```

security {
    ike {
        policy ike {
            mode main;
            proposal-set standard;
            pre-shared-key ascii-text "$9$QrîN3/t1Ru087-V4oz36/p0BIE";
        }
        gateway ike-JAX {
            ike-policy ike;
        }
    }
}

```



```

        address 101.121.30.2;
        external-interface ge-0/0/1.100;
    }
}
ipsec {
    policy vpn {
        proposal-set standard;
    }
    vpn JAX {
        bind-interface st0.0;
        ike {
            gateway ike-JAX;
            ipsec-policy vpn;
        }
        establish-tunnels immediately;
    }
}
policies {
    ...
    zones {
        security-zone Internet {
            interfaces {
                ge-0/0/1.1;
            }
        }
        security-zone VPN {
            interfaces {
                ge-0/0/1.0;
            }
        }
        security-zone JAX-RTR {
            interfaces {
                ge-0/0/2.0 {
                }
            }
        }
        security-zone JAX-FW {
            interfaces {
                ge-0/0/0.0 {
                    st0.0 {
                    }
                }
            }
        }
    }
}
}

```

The routing instance stanza was the actual configuration that would be resident on the router when it was installed at the remote location. The routing options show the static routes between the two devices. In the event of an issue with one route, the qualified next hop would take over:

```

routing-instances {
    JAX-RTR {
        instance-type virtual-router;
    }
}

```

```

interface ge-0/0/1.0;
interface ge-0/0/2.0;
routing-options {
    static {
        route 10.0.0.0/16 {
            next-hop 101.121.10.1;
            qualified-next-hop 10.0.10.2 {
                metric 100;
            }
        }
        route 0.0.0.0/0 {
            next-hop 10.0.10.2;
            qualified-next-hop 101.121.10.1 {
                metric 100;
            }
        }
    }
}
}
}
}

```

The BDL main routers were set up identically to one another. Each was to be active for the routes that were received from the VPN and OSPF. The selection was to be based on a random choice between these two devices. If one of them failed, the other would take over the full load. The configuration for these devices is shown below. The interfaces have descriptions so that the reader can follow along on the prototype diagrams:

```

interfaces {
    ge-0/0/0 {
        description "To the VPN";
        unit 0 {
            family inet {
                address 101.121.30.2/24;
            }
        }
    }
    ge-0/0/1 {
        description "To the BDL LAN";
        unit 0 {
            family inet {
                address 10.0.0.2/24;
            }
        }
    }
}
}

```

Both OSPF and BGP were operational on these devices. The BGP protocol allowed remote sites to be learned by the main locations. If a new site was added to the network, only that site needed to be updated, and all the other sites would not need to be touched. BGP has an import policy and an export policy. The export policy allows the VPN to learn about the local addresses, and the import policy is just a safeguard, ensuring that

if things go south with AT&T the local router is not going to be making bad decisions. OSPF has an export policy as well. This allows the BGP routes learned from the remote locations to be seen by all the devices at the main location. In all the policies, the internal addressing (10/8) is allowed, as well as the default route (0/0). The last piece is the external route preference that has been assigned to OSPF. This allows the BGP routes to be used as the primary and the IPSec tunnels to the remote locations to be used as a backup (learned via OSPF redistributed from the firewall):

```
protocols {
  bgp {
    group VPN {
      type external;
      import Safe-BGP;
      export OSPF-to-VPN;
      peer-as 7018;
      local-as 65432;
      neighbor 101.121.30.1;
    }
  }
  ospf {
    external-preference 190;
    export BGP-to-OSPF;
    area 0.0.0.0 {
      interface ge-0/0/0.0 {
        passive;
      }
      interface ge-0/0/1.0;
    }
  }
}
policy-options {
  policy-statement BGP-to-OSPF {
    term BGP {
      from {
        protocol bgp;
        route-filter 10.0.0.0/8 orlonger;
        route-filter 0.0.0.0/0 exact;
      }
      then accept;
    }
    term other {
      then reject;
    }
  }
  policy-statement OSPF-to-VPN {
    term OSPF {
      from {
        protocol ospf;
        route-filter 0.0.0.0/0 exact;
        route-filter 10.0.0.0/8 upto /24;
      }
    }
  }
}
```

```

        then accept;
    }
    term other {
        then reject;
    }
}
policy-statement Safe-BGP {
    term Valid-routes {
        from {
            route-filter 10.0.0.0/8 orlonger;
        }
        then accept;
    }
    term default {
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then accept;
    }
    term Other {
        then reject;
    }
}
}

```

With the prototype up and running, here's what the route table at the main location showed:

```

lab@BDL-R1> show route

inet.0: 10 destinations, 11 routes (9 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[OSPF/150] 00:22:04, metric 0, tag 0
                   > to 10.0.0.1 via ge-0/0/1.0
                   [BGP/170] 00:37:47, localpref 100
                   AS path: 7018 I

```

The default route is learned from the local firewall (OSPF) and from the remote location (BGP). This allows the users to access the Internet in the event of a failure of the local ISP:

```

10.0.0.0/24        *[Direct/0] 01:04:07
                   > via ge-0/0/1.0
10.0.0.2/32        *[Local/0] 01:14:12
                   Local via ge-0/0/1.0
10.0.10.0/24       *[BGP/170] 00:50:51, localpref 100
                   AS path: 7018 I
                   > to 101.121.30.1 via ge-0/0/0.0
                   [OSPF/190] 00:04:13, metric 0, tag 0
                   > to 10.0.0.1 via ge-0/0/1.0
10.0.20.0/24       *[BGP/170] 00:50:51, localpref 100
                   AS path: 7018 I

```

```

> to 101.121.30.1 via ge-0/0/0.0
[OSPF/190] 00:04:13, metric 0, tag 0
> to 10.0.0.1 via ge-0/0/1.0
10.111.2.0/24 * [OSPF/190] 00:04:13, metric 0, tag 0
> to 10.0.0.1 via ge-0/0/1.0

```

The internal network addresses (10.0/16) were seen here as local addresses and learned addresses from the VPN (BGP) as well as the firewall (OSPF). The route preference of these routes was modified so that the BGP routes were preferred. The last local route was the local end of the IPSec tunnel. If the tunnels were ever to use a routing protocol instead of static routing, this address would be necessary to resolve the addresses that were learned over the tunnel. For our purposes, this address was really not necessary.

A careful eye will have noticed a hidden route in the display above: I added a bogus route to the routes advertised from the VPN to the main location. This verified that the import policies were working. The 12.0.0.0/24 route was hidden when received from the VPN:

```

lab@BDL-R1> show route hidden

inet.0: 10 destinations, 11 routes (9 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

12.0.0.0/24      [BGP ] 00:38:19, localpref 100
                  AS path: 7018 I
                  > to 101.121.30.1 via ge-0/0/0.0

```

Once the routing mechanisms had been defined and the primary/secondary arrangements had been worked out, it was time to look at the class of service configurations. Because the same traffic was seen on all the interfaces at all the locations, all the configurations were the same. The interface names changed, but the remaining parts of the configurations were exactly the same.

Class of Service

Class of service configuration on Junos is not as simple as other elements, like routing protocols and interfaces—some even consider it to be a bear to configure. There are multiple components that have to be configured, and they all have to agree with one another to ensure that traffic is handled in a consistent manner throughout the network. During the initial talks with the CEO, it was observed that the existing firewalls would not meet the demands of the class of service requirements. It was further determined that these devices would only be used as backups for corporate traffic, so this shortfall was OK.



Whenever I have the chance, I do push the One Junos concept. The same features on the MX can be ported to the SRXs. In this case, the CEO acknowledged the effort but stated that the firewalls that were in place would remain there. Hey, I tried!

Class of service is implemented as follows in the Junos environment:

- Incoming traffic is policed and categorized into forwarding classes.
- Internal traffic and egress traffic are handled based on the forwarding classes.
- Egress traffic is shaped and marked based on the forwarding class.

A filter on the ingress interface performs the categorization of incoming traffic. The filter is either a multifield classifier (input firewall filter) or a bandwidth aggregator (BA) classifier. Compliance with bandwidth limits is enforced with the multifield classifier. The difference between the two classification schemes is that the multifield classifier can perform its classification based on any “filterable” field in the packet, while the BA classifier only looks at the class of service marks on the packet.

AT&T’s classes of service are distinguished by the use of the differentiated service code points (DSCPs), or the IP precedence. These fields are part of the packets’ IP headers. AT&T prefers to use the DSCP marking, but will use the IP precedence if the customer’s equipment does not handle the DSCP marking. The classification that AT&T uses (from the [class of service planning document](#)) is shown in [Table 1-2](#).

Table 1-2. AT&T class of service coding

TOS (first 6 bits)	Standard per hop behavior	AT&T class
101 110	DSCP Expedite Forwarding (EF)	COS1
101 000	IP Precedence 5	COS1
011 010	DSCP Assured Forwarding 31 (AF31)	COS2 compliant
011 100	DSCP Assured Forwarding 32 (AF32)	COS2 noncompliant
011 000	IP Precedence 3	COS2 compliant
010 010	DSCP Assured Forwarding 21 (AF21)	COS3 compliant
010 100	DSCP Assured Forwarding 22 (AF22)	COS3 noncompliant
010 000	IP Precedence 2	COS3 compliant
000 000	DSCP Best Effort (DEFAULT)	COS4
011 xxx	DSCP Assured Forwarding 3x (AF3x)	COS2 noncompliant
110 xxx	Reserved for control and signaling	Highest class
111 xxx	Reserved for control and signaling	Highest class
010 xxx	DSCP Assured Forwarding 2x (AF2x)	COS3 noncompliant
101 xxx		COS4

TOS (first 6 bits)	Standard per hop behavior	AT&T class
001 xxx	DSCP Assured Forwarding 1x (AF1x)	COS4
100 xxx	DSCP Assured Forwarding 4x (AF4x)	COS4
000 xxx		COS4

For the traffic to be handled by the appropriate class of service in the VPN network, it has to be marked appropriately in the customer's network. Junos supports a default set of code points for both DSCP and IP precedence. Matching the AT&T classes of [Table 1-2](#) to the output below shows that the two systems are in sync with each other:

```
lab@BDL-R1> show class-of-service code-point-aliases inet-precedence
```

```
Code point type: inet-precedence
```

Alias	Bit pattern
af11	001
af21	010
af31	011
af41	100
be	000
cs6	110
cs7	111
ef	101
nc1	110
nc2	111

```
lab@BDL-R1> show class-of-service code-point-aliases dscp
```

```
Code point type: dscp
```

Alias	Bit pattern
af11	001010
af12	001100
af13	001110
af21	010010
af22	010100
af23	010110
af31	011010
af32	011100
af33	011110
af41	100010
af42	100100
af43	100110
be	000000
cs1	001000
cs2	010000
cs3	011000
cs4	100000
cs5	101000
cs6	110000
cs7	111000
ef	101110
nc1	110000
nc2	111000

The only piece that seems to be out of sync is the compliant and noncompliant classes. In effect, AT&T handles six classes of service, in the following order:

1. COS1
2. COS2 compliant
3. COS2 noncompliant
4. COS3 compliant
5. COS3 noncompliant
6. COS4

The noncompliant traffic may be discarded at the egress of the VPN if there is not enough bandwidth available for that service class. The compliant and noncompliant markings can be assigned prior to traffic entering the network, or by the network itself based on the profile chosen by the customer.

In this case, the marking is done on the customer's edge for each class (reference [Figure 1-5](#)). This configuration allows the router to shape the traffic as it enters the network. The classification of traffic is done at two points on the Juniper routers: traffic from the local LANs is classified by a multifield classifier and traffic from the VPN is classified by a BA classifier. Traffic shaping and remarking is done only on the interface to the VPN. The remarking is only necessary on the VPN interface because the other portions of the network do not participate in the class of service operations.

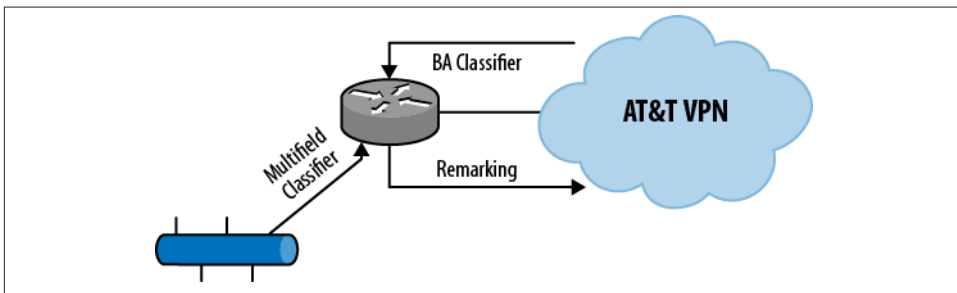


Figure 1-5. Class of service functions

The first order of business was the creation of the multifield classifier. This firewall filter accepted the incoming traffic and marked each packet into a forwarding class. The filter was installed on the LAN-facing interfaces for incoming traffic:

```
ge-0/0/1 {
  description "To the BDL LAN";
  unit 0 {
    family inet {
      filter {
```



```

        input CLASS-OF-SVC;
    }
    address 10.0.0.2/24;
}
}
}

```

The filter had four terms to divide traffic into the four forwarding classes. The calculation of the maximum transmission rate for each class was handled previously (see the section “[Design Trade-Offs](#)” (page 6)), but the calculation of the maximum burst size was a point of discussion here: select too small of a maximum burst size and the traffic would be restricted; select too high of a value and no traffic would be policed. The recommended value is the amount of traffic (in bytes) that can be transmitted on an interface in 5 milliseconds, while the minimum value is 10 times the maximum transmission unit (MTU) of the interface. Considering that all the incoming interfaces in this environment were Fast Ethernet (100 Mbps), the recommended value can be calculated to be 62,500 bytes and the minimum value would be 15,000 bytes. The only traffic type that might have a problem with these values is the COS1 video traffic. To avoid any loss of multimedia traffic, the recommended value was doubled to 125,000 bytes.

The first term of the filter looked for traffic destined for or originating from the multimedia servers (10.0.10.10 and 10.0.10.12). This might catch some non-video traffic, but would have a minimal impact on the network. There were three parts to this term—the prefix list that identified the servers (we used the list rather than hardcoding the addresses in the filter just to make things easier to modify later), the policer that limited the traffic (all excess COS1 traffic is discarded), and the filter term that put all the pieces together:

```

lab@BDL-R1# show policy-options
prefix-list MM-SVR {
    10.0.10.10/32;
    10.0.10.11/32;
}
[edit]
lab@BDL-R1# show firewall
family inet {
    filter CLASS-OF-SVC {
        term COS1 {
            from {
                prefix-list {
                    MM-SVR;
                }
            }
            then {
                policer COS1;
                forwarding-class COS1;
                accept;
            }
        }
    }
}

```

```

    }
}
policer COS1 {
    if-exceeding {
        bandwidth-limit 768k;
        burst-size-limit 125k;
    }
    then {
        discard;
    }
}

```

The other terms were very similar to the first. For each, a prefix list and a policer complemented the term for traffic identification and handling. The last term was a catchall for any other traffic:

```

[edit]
lab@BDL-R1# show firewall
family inet {
    filter CLASS-OF-SVC {
        term COS1 {
            from {
                prefix-list {
                    MM-SVR;
                }
            }
            then {
                policer COS1;
                forwarding-class COS1;
                accept;
            }
        }
        term COS2 {
            from {
                prefix-list {
                    CRM-SRV;
                }
            }
            then {
                policer COS2;
                forwarding-class COS2;
                accept;
            }
        }
        term COS3 {
            from {
                protocol tcp;
                port [ http https ];
            }
            then {
                policer COS3;
                forwarding-class COS3;
            }
        }
    }
}

```

```
[edit]
lab@BDL-R1# show policy-options
```

24 | Chapter 1: An Enterprise VPN

a loss priority of *low* is the COS2 conforming traffic, while COS2 traffic with a loss priority of *high* is the COS2 nonconforming traffic. This change in terminology is reflected in the BA classifiers and the rewrite rules that are presented in the following paragraphs.

The multifield classifier was copied to all the other router configurations and applied to the corporate LAN interface.

The bandwidth aggregate classifiers were the next piece of the class of service configuration to be created. These looked at incoming traffic and placed the packets marked with the proper code points into the proper forwarding classes. The BA classifiers were assigned to interfaces facing the VPN. The base configuration for the BA classifiers was:

```
[edit class-of-service]
lab@BDL-R1# show
classifiers {
    dscp ATT {
        forwarding-class COS1 {
            loss-priority low code-points [ ef cs5 ];
        }
        forwarding-class COS2 {
            loss-priority low code-points [ af31 cs3 ];
            loss-priority high code-points [ af32 af33 ];
        }
        forwarding-class COS3 {
            loss-priority low code-points [ af21 cs2 ];
            loss-priority high code-points [ af22 af23 ];
        }
        forwarding-class COS4 {
            loss-priority low code-points [ be af11 af12 af13 af41
                                           af42 af43 ];
        }
        forwarding-class network-control {
            loss-priority low code-points [ nc1 nc2 ];
        }
    }
}
interfaces {
    ge-0/0/0 {
        unit 0 {
            classifiers {
                dscp ATT;
            }
        }
    }
}
```

The information for this configuration was gleaned from [Table 1-1](#) and the Junos default code point aliases. The rewrite rules for the routers were the opposite of the classifier rules, minus the duplicates. The configuration for the rewrite rules was:

```

[edit class-of-service]
lab@BDL-R1# show
classifiers {
...
interfaces {
    ge-0/0/0 {
        unit 0 {
            classifiers {
                dscp ATT;
            }
            rewrite-rules {
                dscp ATT;
            }
        }
    }
}
rewrite-rules {
    dscp ATT {
        forwarding-class COS1 {
            loss-priority low code-point ef;
        }
        forwarding-class COS2 {
            loss-priority low code-point af31;
            loss-priority high code-point af32;
        }
        forwarding-class COS3 {
            loss-priority low code-point af21;
            loss-priority high code-point af22;
        }
        forwarding-class COS4 {
            loss-priority low code-point be;
        }
        forwarding-class network-control {
            loss-priority low code-point nc2;
        }
    }
}
}

```

In most cases, the rewrite rules and the classifier rules should match. I included the IP precedence code points (*cs1–cd7*) in the classifier just in case these arrive from the network. They should not, but I was making sure all bases were covered. The rewrite rules only show what is leaving the router to the VPN (only the use of DSCP code points for outgoing traffic).

The next piece of the configuration was to define the actual forwarding classes that have been referenced in each of the other configured portions. The forwarding classes are the internal reference points for traffic handling in the router. Traffic is assigned to forwarding classes based on some criteria. There is no specific coding (in the packet

header) that identifies the forwarding class except for the mapping defined in the classifiers (multifield and BA). The configuration for the forwarding classes associated one of the queues to each of the forwarding class names. In this case, the names and the queues were an ordered set. The configuration was:

```
[edit class-of-service]
lab@BDL-R1# show
classifiers {
...
forwarding-classes {
    queue 0 network-control;
    queue 1 COS1;
    queue 2 COS2;
    queue 3 COS3;
    queue 4 COS4;
}
```

A fifth class of service was added to the mix for network control traffic (a.k.a. *routing traffic*); BGP had to have a place in the class of service scheme. The network control traffic was assigned its own DSCP codes (*nc1* and *nc2*), as was seen in the rewrite rules and the BA classifier.

Up to now, everything seemed to fit together in a logical sense. The next configuration steps, however, were where the abstract concepts were added to the CoS configuration. The first concept was the schedulers. These associate a forwarding class (and its queue) to a priority for outgoing queuing; they can also identify drop profiles for weighted random early detection (WRED) traffic. In our case, the default drop profiles (linear) were adequate.

For each of the schedulers, a transmit rate and a priority were defined. The rates can be defined as exact (cannot exceed) or as allowing additional traffic to use other idle rates. In our case, the use of idle rates was allowed. The configuration for the schedulers was:

```
[edit class-of-service schedulers]
lab@BDL-R1# show
COS1 {
    transmit-rate 768k;
    buffer-size temporal 200k;
    priority high;
}
COS2 {
    transmit-rate 100k;
    priority medium-high;
}
COS3 {
    transmit-rate 500k;
    priority medium-low;
}
COS4 {
    priority low;
```

```

}
Network-Control {
    transmit-rate percent 10;
    priority high;
}

```

When we created the schedulers, we named them the same as the forwarding classes and the filters. This approach can be confusing, but it assures that all the elements are the same for all the classes. The schedulers are the inverse of the multifold classifiers, in the sense that traffic entering an interface is subjected to the classifier, while the traffic exiting the interfaces is subjected to the scheduler. The rates for both are often the same; if high-speed interfaces are mixed with low-speed interfaces the classifier and the scheduler might have different values (so as not to overrun the slower interface), but due to the traffic patterns in this engagement, that was not the case here.

Once the schedulers were defined, they had to be mapped to the forwarding classes with a scheduler map. The scheduler map brings all the elements together and is referenced on the interfaces that need egress queuing (all interfaces in our environment, LAN and VPN). It was configured as follows:

```

[edit class-of-service]
lab@BDL-R1# show
classifiers {
...
forwarding-classes {
...
interfaces {
    ge-0/0/0 {
        scheduler-map ATT;
        unit 0 {
            classifiers {
                dscp ATT;
            }
            rewrite-rules {
                dscp ATT;
            }
        }
    }
    ge-0/0/1 {
        scheduler-map ATT;
    }
}
rewrite-rules {
...
scheduler-maps {
    ATT {
        forwarding-class COS1 scheduler COS1;
        forwarding-class COS2 scheduler COS2;
        forwarding-class COS3 scheduler COS3;
    }
}
}

```

```

        forwarding-class COS4 scheduler COS4;
        forwarding-class network-control scheduler Network-Control;
    }
}

```

Once this configuration was checked and committed on one router, it was copied to all the routers in the network. The interface names were adjusted to match the locations, but all other aspects were copied wholesale. The class of service was verified on the assigned interfaces with the operational command:

```

lab@BDL-R1> show interfaces ge-0/0/0 extensive
Physical interface: ge-0/0/0, Enabled, Physical link is Up
...
Traffic statistics:
...
Egress queues: 8 supported, 5 in use
Queue counters:      Queued packets  Transmitted pkts  Dropped pkts
  0 network-cont      1169             1169              0
  1 COS1               0                0                 0
  2 COS2               0                0                 0
  3 COS3               0                0                 0
  4 COS4               0                0                 0
Queue number:        Mapped forwarding classes
  0                   network-control
  1                   COS1
  2                   COS2
  3                   COS3
  4                   COS4
Active alarms   : None
Active defects  : None
MAC statistics:
...
CoS information:
  Direction : Output
  CoS transmit queue  %    Bandwidth    %    Buffer Priority Limit
                   %    bps    %    usec
  0 network-control   10   10000000    r    0    high  none
  1 COS1              0    768000    0 200000    high  none
  2 COS2              0    100000    r    0 medium-high none
  3 COS3              0    500000    r    0 medium-low  none
  4 COS4              r      r      r    0    low   none
Interface transmit statistics: Disabled

Logical interface ge-0/0/0.0 (Index 68) (Generation 133)

```

The output here is truncated to save space, with the remaining portions showing the class of service information. The queues are shown first, with the stats for each queue, the number of dropped packets, and the relative queue names. The CoS information

shows the scheduler information for each class (transmit queue). Note that the network control class has the highest bandwidth associated with it. This is the 10% that was assigned to the background traffic. As network bandwidth grows, this number can be reduced to smaller percentages (the default is 5% or 5 Mbps on a Fast Ethernet interface).

The class of service attributes showed each of the configured items. This display can be used to verify the configurations. The specific AT&T information was shown with the following commands:

```
lab@BDL-R1> show class-of-service classifier name ATT
Classifier: ATT, Code point type: dscp, Index: 7594
  Code point      Forwarding class      Loss priority
  000000          COS4                  low
  001010          COS4                  low
  001100          COS4                  low
  001110          COS4                  low
  010000          COS3                  low
  010010          COS3                  low
  010100          COS3                  high
  010110          COS3                  high
  011000          COS2                  low
  011010          COS2                  low
  011100          COS2                  high
  011110          COS2                  high
  100010          COS4                  low
  100100          COS4                  low
  100110          COS4                  low
  101000          COS1                  low
  101110          COS1                  low
  110000          network-control      low
  111000          network-control      low
```

```
lab@BDL-R1> show class-of-service rewrite-rule name ATT
Rewrite rule: ATT, Code point type: dscp, Index: 7594
  Forwarding class      Loss priority      Code point
  network-control      low                111000
  COS1                  low                101110
  COS2                  low                011010
  COS2                  high               011100
  COS3                  low                010010
  COS3                  high               010100
  COS4                  low                000000
```

```
lab@BDL-R1> show class-of-service scheduler-map ATT
Scheduler map: ATT, Index: 3797

Scheduler: Network-Control, Forwarding class: network-control,
Index: 40528
  Transmit rate: 10 percent, Rate Limit: none,
  Buffer size: remainder,
  Buffer Limit: none, Priority: high
  Excess Priority: unspecified
```

Drop profiles:...

Scheduler: **COS1**, Forwarding class: **COS1**, Index: 46705
Transmit rate: 768000 bps, Rate Limit: none,
Buffer size: 200000 us,
Buffer Limit: none, Priority: high
Excess Priority: unspecified
Drop profiles:...

Scheduler: **COS2**, Forwarding class: **COS2**, Index: 46706
Transmit rate: 100000 bps, Rate Limit: none,
Buffer size: remainder,
Buffer Limit: none, Priority: medium-high
Excess Priority: unspecified
Drop profiles:...

Scheduler: **COS3**, Forwarding class: **COS3**, Index: 46707
Transmit rate: 500000 bps, Rate Limit: none,
Buffer size: remainder,
Buffer Limit: none, Priority: medium-low
Excess Priority: unspecified
Drop profiles:...

Scheduler: **COS4**, Forwarding class: **COS4**, Index: 46708
Transmit rate: unspecified, Rate Limit: none,
Buffer size: remainder,
Buffer Limit: none, Priority: low
Excess Priority: unspecified
Drop profiles:...

The drop profiles have been deleted from each of the schedulers to save space.

Once the class of service was up and running, a series of traffic tests were conducted to verify that the traffic was mapped to the proper classes. The policing and shaping were not verified, due to the limits of the test lab. Once the CEO was satisfied that things were going to work as expected, the system implementation was planned, the equipment was staged and tested, and the system was cut over to the new network.

Cut-Over

Once I completed the prototype testing, the configurations were scrubbed and each device was set up with its final configuration. A final test was performed to verify that each device could communicate with the LAN Ethernet port for management access. I double-checked all the addresses, subnet masks, and default routes.

The reason for all the caution was that local personnel were installing the remote locations, but all troubleshooting was going to be performed from the main location. We

decided that the cut-over was going to be done over a period of evenings and that each remote location was going to be cut over in a separate maintenance window. This reduced the coordination effort for the remote personnel and also the overall stress level of the installation.

As a note, this also allowed us to learn from the initial cut-over. On the first evening, we had some issues with the firewall rules and patching that I had not anticipated, and dealing with them made the other evenings go a lot easier.



Another of the lessons learned for this book: “What you did not think could happen will happen at the most inopportune time and bite you in the butt.”

The cut-over plan was to bring the main location up and make it operational with the VPN. When a remote site was brought online, the traffic for that site would be cut over to the VPN links. Testing and verification were performed on a site-by-site basis.

Main Site

The installation of the routers at the main site went without a problem. The Ethernet links from AT&T connected to the routers, BGP came up, and routes were seen being exchanged over the interfaces. The internal BGP link between the routers came up and the routes were being exchanged between the two MX10s. We installed a test PC on the LAN switch for later testing (this PC had a default gateway of the MX10s). With the MX10 ready for traffic, only the BDL firewall needed to be altered to allow communications over the new VPN.

With this site installed, the remote sites were attacked. The first site to come online was the JAX site.

Remote Site JAX

The initial cut-over started well enough but then went downhill fast. The equipment arrived at the remote site on time and in good condition. I walked the remote person through the process of installing the router in the proper rack and connecting the cables (power and Ethernet).

The first problem was that when the router was powered up, the lights came on green for the processor and the Ethernet interface, but remained red for the VPN interface. The first remote site was connected to the VPN with two T1 circuits, bundled together

to form a 3 Mbps circuit. The configuration for the interfaces, the multilink point-to-point protocol, and the addressing were unique to this site. The cabling between the router and the T1 “smart jack” was supposed to be a straight-through cable, but things did not look good.

The next problem occurred when I attempted to SSH to the router via the existing infrastructure. No-Go, with a big N&G. The connections timed out for both SSH and Telnet. I was not getting a connection refused response, just a lost packet timeout. I checked with the CEO for the firewall access, gained access to that device, and tried again from there to the router. This operated OK, so I backed out and looked at the Cisco rules. I added a rule to allow SSH access through the device from the main addresses to the router, and vice versa.

Back to problem 1—once I had access to the router, the T1 interfaces showed down with the following output (both interfaces were the same):

```
lab@JAX> show interfaces t1-2/0/0
Physical interface: t1-2/0/0, Enabled, Physical link is Down
  Interface index: 145, SNMP ifIndex: 524
  Link-level type: Multilink-PPP, MTU: 1510, Clocking: External,
  Speed: T1,
  Loopback: None, FCS: 16, Framing: ESF
  Device flags   : Present Running Down
  Interface flags: Hardware-Down Point-To-Point SNMP-Traps
  Internal: 0x4000
  Link flags     : None
  Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
  Keepalive: Input: 0 (never), Output: 0 (never)
  LCP state: Down
  CHAP state: Closed
  PAP state: Closed
  CoS queues      : 8 supported, 8 maximum usable queues
  Last flapped    : 2012-02-13 23:14:13 UTC (05:49:51 ago)
  Input rate      : 0 bps (0 pps)
  Output rate     : 0 bps (0 pps)
  DS1 alarms     : LOF, LOS
  DS1 defects    : LOF, LOS
```

This indicates that the physical connection on the interface is not operational. I spoke to the AT&T technician working on the case, and she verified that the interface was showing down at their end.

I went into Troubleshooting 101 mode, “Start at the Physical Layer.” The patch cables were new LAN cables, so the probability that they were both bad was low. A quick search showed that the T1 smart jack was wired as shown in [Figure 1-6](#).

Pin	Pair	Signal	Color
1	R	RX Ring	 Orange/White
2	T	RX Tip	 White/Orange
3		reserved	 White/Green
4	R1	TX Ring	 Blue/White
5	T1	TX Tip	 White/Blue
6		reserved	 Green/White
7		shield	 White/Brown
8		shield	 Brown/White

Figure 1-6. RJ48C pin assignments

I walked the guy on the other end through the process of creating a loopback jack (twist wire 1 to 4 and wire 2 to 5), and we tested each interface.



When in doubt, look it up! Warriors are looked to as walking references of data communications, but while I have seen a lot in the last 30 years, I cannot remember it all. I am the first to say I don't know, but also the first to look it up and move forward.

All the interfaces transitioned to up with the loopback plug in place, so we knew the problem was in the patch cables. We traded out the patches for another set with no improvement. At this point, we searched for a store close to the site that had various patch cables in stock and located a T1 crossover cable. Once these were procured and installed, the interfaces all came up—for whatever reason, the patch cables needed to be rolled for these interfaces.

We checked whether the interfaces were up and operational with the AT&T technician, and she indicated that they all looked up and good.

Now it was time to see if the VPN was up and operational. We checked the BDL site and found that it showed that BGP was receiving the prefix for JAX. That was a good sign. We did a ping test from BDL and got a response from both ends of the CE to PE link at JAX, and vice versa.

Logging into the JAX router allowed us to ping the servers in BDL using the LAN address as the source address. This did not work because the return route was still through the IPSec tunnel—oops! We satisfied ourselves with pings to the test PC on the LAN side of the BDL routers.

Once the JAX to BDL VPN link was verified as operational, it was time to make the plans for swapping traffic from the IPSec tunnel to the VPN. The steps for that transition were:

1. Change the existing static route for the 10/8 addresses at the JAX firewall to have a higher metric (10)—this would be the backup route in case the VPN failed.
2. Add a second static route on the firewall pointing to the VPN router, leaving the metric at the default value—traffic from the devices at the site have a default gateway pointing to the firewall that will redirect the traffic to the VPN router.
3. Install OSPF at the BDL firewall and redistribute the static routes into OSPF.
4. Change the administrative distance of the static routes to be higher than that of OSPF (110).

These changes were scheduled for the next maintenance window, and all went as planned (to everyone's surprise after the issues of the initial install). The routing table from the BDL router showed that the JAX remote site learned from BGP and also from the OSPF external routes (JAX and default):

```
lab@BDL1> show route

inet.0: 16 destinations, 26 routes (16 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[OSPF/180] 00:00:10, metric 0, tag 0
                  > to 10.1.10.130 via ge-0/0/0.5
                  [BGP/190] 02:10:05, localpref 100
                  AS path: 1234 I
10.0.30.0/24      *[BGP/170] 01:53:02, localpref 100
                  AS path: 7018 I
                  > to 10.1.10.5 via ge-0/0/3.0
                  [OSPF/180] 00:10:44, metric 0, tag 0
                  > to 10.1.10.130 via ge-0/0/0.5
224.0.0.5/32      *[OSPF/10] 01:53:40, metric 1
                  MultiRecv
```

A couple of notes from the routing table. First, in Junos, the default route preference for external OSPF routes is 150, while the route preference for BGP is 170. In the BDL router, I wanted to prefer the BGP routes to the remote locations over the OSPF routes being redistributed from the firewall. To this end, the external OSPF routes are given a higher route preference in the configuration:

```
lab@BDL1> show configuration protocols ospf
external-preference 180;
area 0.0.0.0 {
    interface ge-0/0/0.5;
    interface lo0.0;
}
```

The default route received from the remote site allows alternate access to the Internet in the event that the local Internet access is lost. For the remote addresses, the BGP route is preferred over the OSPF route, but in this case, the local OSPF route (the redistributed static route from the firewall) must be preferred over the remote site's default route. To accomplish this, a route policy had to be created that changed the route preference for the incoming static route. The route policy looks like this:

```
lab@BDL1# show policy-options
policy-statement Modify-default {
    term 1 {
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then {
            preference 190;
            accept;
        }
    }
    term 2 {
        then accept;
    }
}
```

All incoming BGP routes are accepted, but the default route receives a different route preference.

When the routes were complete for the remote site, the traffic flows to the remote sites showed that the VPN route was being used for outgoing and incoming traffic. When the interface to the VPN on BDL1 was disabled, all traffic flowed over BDL2. When that link was taken down, the traffic reverted to the firewall tunnel. The use of the firewall tunnel caused most users to reinitiate their Internet sessions (the firewall does not like half-open sessions). Once the users reinitiated the Internet sessions, this secondary backup performed as expected. This scenario was acceptable to the CEO.

Remote Sites PHL and IAD

The other remote sites were cut in a similar manner to the JAX site. Each had a few little glitches (for example, static routes not having the right next-hop address), but all the sites came up and traffic was pushed to the new VPN connectivity.

As a tip that the architecture was correct and that the class of service settings were correct, nobody knew that the cut-over was complete. Nothing broke.

Backup Site BNA

The remaining site to bring online was the backup data center location in BNA. With this site, like the others, the cut-over went off without a hitch. In fact, it was easier because there was no user traffic to interrupt. This site is used when a server crashes in the main data center. Traffic is rerouted to the backup site by the server load balancers; the switchover is totally transparent to the users.

Once the router was up and connected to the VPN, I altered the static routes and tested traffic on the VPN interfaces.

Conclusions

I started this engagement as a tribe of one, but when I got to the finish, I had adopted a new tribe; they were not network engineers or networking professionals, but salespeople and warehouse personnel. We worked together and completed the installation of a new communications backbone for the company that allowed it to grow and reach new heights.

Each engagement is different: you see different technologies and meet a world of different people. And often you find that folks who have no interest in data communications can be productive members of the tribe. The warehouse person in Jacksonville had no idea what an RJ48C connector was, but with a little instruction and patience, he was able to create a usable loopback jack for testing the T1 interfaces. I have no idea what he used to strip the patch cord or what he used to connect the wires together, but it worked and we got the silly thing up and running. I am sure that he does not consider himself part of the tribe, but I sure do.

The technology for this engagement is well baked, and the Juniper Networks routers that were involved were both state of the art (MX10) and legacy (J2320s), yet all are supported, all run the same code, and none brought any surprises to the game. That is the joy of Junos.

Maintaining IDP Systems

It is said that for every combat soldier firing a weapon in battle, there are nine soldiers keeping that individual in boots, bullets, beans, and beds. Some of the time, being a network warrior means being on the front lines designing, installing, troubleshooting, and configuring the devices that service providers and enterprises rely on for their communications. Other times, it means being behind the front lines providing the support. These assignments may not be quite as exciting, but they can be just as interesting if you put your mind in the right space. So bear with this chapter as it figures out the right maintenance levels for a complicated setup.

For this engagement, a group of us were providing support—bullets-and-beans support—to a major wireless carrier. The gig was tuning the maintenance and feeding of a gaggle of IDP8200s dispersed throughout the network. The IDP8200s are part of a bulletproof intrusion protection/detection system provided by Juniper Networks. The other warriors in this tribe consisted of a resident engineer and the wireless provider's application security engineers. This made us quite a tribe, as each of us brought a honed skill set and all the corresponding special interests.

For myself, it was as much of a learning experience as a support assignment. I had previously installed and initially configured IDP8200s, but I had never gotten into the guts of the things. In fact, the attentive reader will notice that this chapter is somewhat different than the others, not just because the title includes the word “Maintaining,” but because of the technologies we were applying, and my deep dive into them. So, here was an opportunity to get down and dirty with a new device, operating system, and the processes that are used to keep it operating.

The IDP8200s are configured and monitored by a Juniper Networks Network and Security Manager (NSM) server and report back to a Juniper Networks Security Threat Response Manager (STRM). SSH access to the IDPs completes the management suite for these devices.

IDP8200 Background

I did my homework on the IDP devices during the trip to the client, and will do my best to recap it here.

The official datasheet put it bluntly:

Juniper Networks® IDP Series Intrusion Detection and Prevention Appliances provide comprehensive management of unwanted applications and easy-to-use in-line protection that stops network- and application-level attacks before they inflict any damage, minimizing the time and costs associated with maintaining a secure network. Using industry-recognized stateful detection and prevention techniques, the IDP Series provides zero-day protection against worms, trojans, spyware, key loggers, and other malware from penetrating the network or spreading from already infected users.

The IDP8200s can be installed as inline active devices, detecting and blocking threats that meet security policy rules. They can also be installed as tap devices, monitoring the traffic and reporting the results, as shown in [Figure 2-1](#). The IDP8200s at our client carrier's network were installed in this latter *tap mode*; the traffic they see is mirrored traffic from the transport devices in the system.

The IDP8200s are placed to protect the critical portions of the carrier's network and to monitor *customer traffic* for anomalies. Due to the dynamic nature of attacks and the attack signatures that counter them, these devices need constant attention. The amount of traffic on this carrier's network is totally amazing. Some of the IDP8200s in the hottest network locations had to be fine-tuned to reduce the CPU load on the line cards, again requiring constant vigilance.

The IDP8200 supports six processors (line cards). In this deployment, each slot was configured with a 1 Gigabit Ethernet card. The IDP8200 supposedly can handle 5 million sessions and has a throughput of 10 Gbps, and from what we saw on the deployed devices, we believe those numbers. These things were screaming with traffic.

Command-Line Interface

The guts of the IDP8200 is a control processor running a variant of Unix. There is a custom set of IDP commands (called SCIO commands) that allow access to the processes, interfaces, and policies of the device. These cryptic commands are reached on the command line by secure shell access to the device and logging in as the root user.

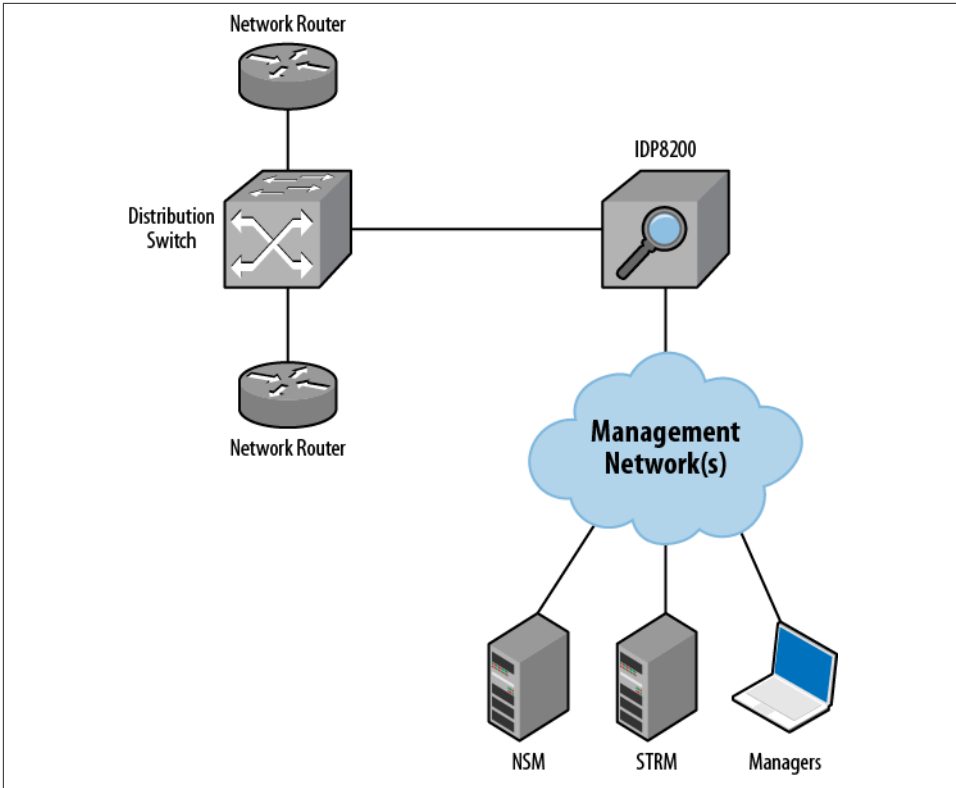


Figure 2-1. IDP8200 deployment

The most common commands and their descriptions are shown in [Table 2-1](#).

Table 2-1. Common SCIO commands

SCIO command	Description
<i>scio idp-cpu-utilization</i>	Shows the current CPU status for processor 0
<i>scio subs service detail s0</i>	Shows the number of sessions per protocol type
<i>scio var -s s0 sc_attack_table</i>	Displays the attack tables from each processor
<i>scio policy list s0</i>	Displays the current policy assigned per processor
<i>scio subs status s0</i>	Displays the traffic flows per processor
<i>scio reset -s s0 sc_attack_table</i>	Clears the attack tables
<i>scio getsystem</i>	Displays the IDP system information

An example of a *scio* command is the *getsystem* command. It shows the version, serial number, and mode of the IDP device:

```
[root@idp ~]# scio getsystem
Product Name: NS-IDP-8200
Serial Number: 0254107766552212
Software Version: 5.0.137283.d1
IDP Mode: sniff
HA Mode: Disabled
Detector Version: 5.0.110110223
Software License: Permanent
Software Expiration Date: never
```

A number of other commands are useful as well, and in the following paragraphs, these will be called out along with a description of their uses.

The security policy that controls the actions of the IDP device is stored in the directory `/usr/idp/device/state/s0`. Issuing the `cat policy.set` command displays the contents of the policy. The policy can be updated using the `vi` editor, but this is not a recommended procedure.



The information found with the CLI commands can also be found from the administrator's interface of NSM, if you prefer. Although it's great for managing devices, this warrior needs to be on the command line to see what is happening.

The interface configurations and interface statistics are displayed with the `ifconfig` command. This shows the transmitted and received packets as well as the errors seen on the interfaces. The `ethtool` commands can be used to configure the interfaces on the device (although again, as with the security policy, using NSM is safer.)

The set of commands that are used to display the statistics gathered on the IDP can be displayed using the `sctop` command. The options for the command are:

```
[root@idp]# sctop
'h' - Display this help          'v' - reverse sort order
'a' - ARP/MAC table             '0' - disable sorting
'i' - IP flows                  '1' - sort by bytes/session
'c' - ICMP flows                '2' - sort by packets/session
'u' - UDP flows                 '3' - sort by expiration
't' - TCP flows                 '4' - sort by service
'o' - APE flows                 '5' - sort by dst port
'r' - RPC table                 '6' - sort by src ip
'x' - RPC XID table             '7' - sort by dst ip
's' - Subscriber's status       '8' - sort by vlan
'm' - Memory statistics
'l' - Q-Module statistics
'e' - Rulebase statistics
'g' - Aggregate statistics
'k' - Attack statistics
'p' - Spanning tree protocol
'b' - IP Action table
```

```
'z' - Packet distribution
'd' - Strip Chart
'f' - Fragment chain
'w' - HA status
'y' - IDS cache statistics
'q' - Quit the program
```

When cases are opened, the Juniper Technical Assistance Center (JTAC) asks for the technical support information from the device. The IDP8200 creates a compressed file of all the relevant information when the *tech-support* command is entered. The output defines the filename and location. The embedded FTP client allows this file to be sent directly to JTAC's *ftp* server (e.g., [root@idp] ftp.juniper.net/public/incoming/2011-1128-123).

The commands to *stop*, *reboot*, and *start* the IDP process are called the *idp.sh* commands. These should be used only when the device is not in service, as they totally disrupt the device and its protection. Also, it takes the device up to 10 minutes to reboot. Be very careful with these commands (*idp.sh start*, *stop*, *reload*, and *restart*). The commands also allow the display of the status and version of the device (*isp.sh status*, *version*). An example of the output of the *idp.sh status* command is shown below:

```
[root@idp ~]# idp.sh status
Retrieving status...
idpinit (pid 6750).....ON
idpengine_0 (pid 7338).....ON
idpengine_1 (pid 7337).....ON
idpengine_2 (pid 7334).....ON
idpengine_3 (pid 7339).....ON
idpengine_4 (pid 7382).....ON
idpengine_5 (pid 7416).....ON
```

The IDP8200 also supports the normal Unix-based tools (*ping*, *traceroute*, *tcpdump*). If you are familiar with Unix, the normal commands are used to check memory usage (*df -kh*), compress logfiles (*tar czvf*), and edit files.

The path to the system logfiles on the IDP8200 is */usr/idp/devices/var/sysinfo/logs/*, and the path to the core files in the case of core dumps is */usr/idp/devices/var/corefiles/*.

Web Management Interface

The IDP supports a web interface in addition to the CLI. The web interface can be used for the initial setup of the device and modification of the networking parameters. The main screen of the web user interface is shown in [Figure 2-2](#).

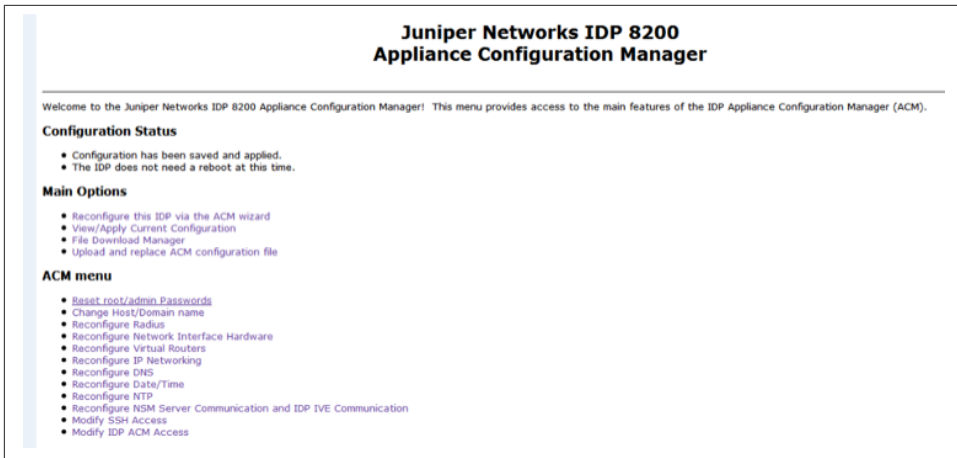


Figure 2-2. IDP8200 web user interface

If you use a good pair of glasses, you can see that the GUI allows the configuration of the administrative parameters that are necessary to get the IDP up and running. The ACM wizard is a step-by-step sequence that walks an administrator through the setup process.

The one piece of this puzzle requiring additional information is the virtual router configuration for the IDP. On most Juniper devices, the virtual routers are optional constructs that allow the administrators to segment the device. On the IDP, the virtual routers are mandatory. Each virtual router is assigned to a pair of interfaces: an input and an output interface. The virtual routers define the interface modes and the failure capabilities of the IDP8200. In our deployment, the IDP is not actually passing traffic, so the failover capabilities are not important and the virtual routers are set on sniffer mode.

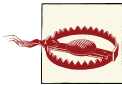
In the example shown in [Figure 2-3](#), interfaces *eth2* and *eth3* are associated with virtual router *vr0* and are in sniffer mode. Changes that are made here are saved and applied to the device. For an interface to be associated with a security policy and monitor traffic, it must be associated with a virtual router, which is then associated with a security policy. There is a default set of virtual routers, but these are not initially associated with the security policy.

In this step, you must configure the interfaces that the IDP Senses

For each pair of interfaces, select the mode you want each pair

Active?	Interfaces	Virtual Router	Mode
☒	eth2,eth3	vr0	☐ Transparent ☒ Sniffer
☐	eth4,eth5	vr4-5	☐ Transparent ☒ Sniffer

Figure 2-3. Virtual router configuration



During the initial discovery phase with these IDP8200s, the tribe found that a number of interfaces were “seeing” traffic but were not assigned to a virtual router. That meant that traffic was not being analyzed and as such could have contained potentially dangerous stuff. Once we found this little goof, we checked each and every interface in the system.

NSM Management

The recommended mechanism for managing the Juniper Networks IDP devices is via the Network and Security Management (NSM) platform. NSM is a server platform that is used to manage firewalls, routers, switches, and IDP devices. Rumor has it that Juniper Networks is doing away with NSM and that all management will, in the near future, be provided by the **Junos Space platform**. While Juniper Networks is not giving any time-lines for this migration, NSM is still the main platform for managing the IDP platforms.



The actual NSM screens show too much information that could be used to compromise the identity of this client, so the NSM demonstration mode is used to recreate and display the capabilities of this management platform.

The device is added to NSM in the same way as any other device and appears as an IDP8200 (or IDP500, as shown in **Figure 2-4**). Editing of the device configuration is performed by right-clicking on the device and following the prompts. The only part of the configuration that is not assigned at the configuration screen is the security policy for the device. The security policy is created in the Policy Manager and assigned to the device from those screens.

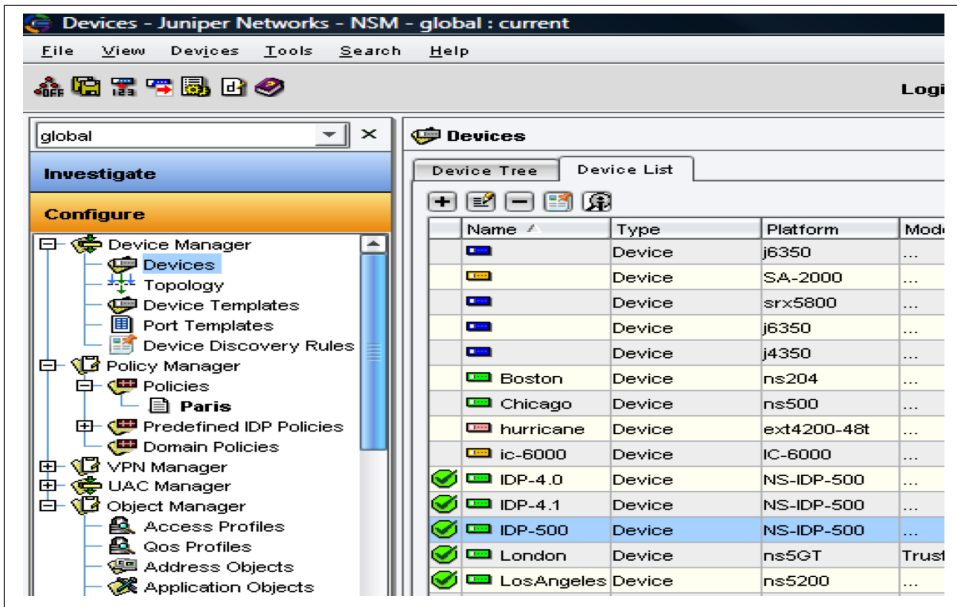


Figure 2-4. NSM devices

This separation of device from policy allows the same policy to be assigned to multiple devices. NSM supports multiple rulebases for a policy. The overall construct is the security policy (named *Paris* in Figure 2-5), which references rulebases (Firewall, Multicast, IDP, Exempt, and/or BSG Transactions). Each rulebase is made up of a set of sequential rules that are applied to the traffic. The rules provide the match criteria for the traffic and the actions to take when the traffic matches the criteria. The IDP policies are on the IDP rulebase tab on the policy screens.

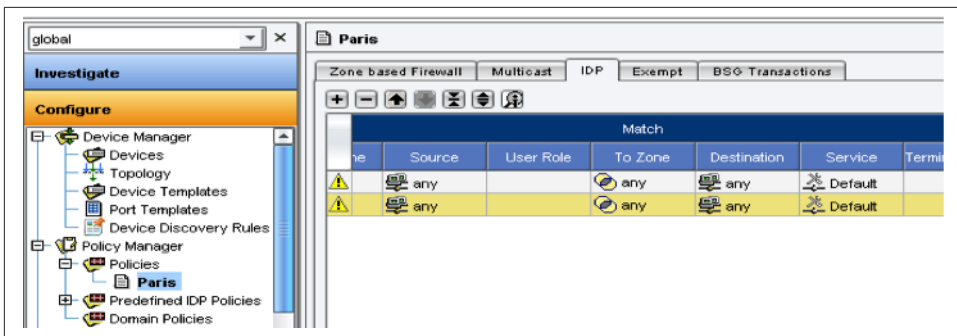


Figure 2-5. NSM IDP policy



When the IDP feature is added to the SRX, the security policy calls the IDP feature an *application service*. The “Zone based Firewall” tab identifies the traffic to be matched, and the IDP policy defines the IDP actions.

Once changes are made to the device, the changes are updated to the IDP8200 from the NSM server (again, right-click on the device from the device tree screen and follow the prompts).

Support Tasks

Enough of the background information; let’s get to why the carrier hired us for this gig. As described earlier, this job was different from the normal design-and-install project that is we warriors’ bread-and-butter livelihood.

The support tasks that were required for this assignment started out being very labor intensive, but due to the skills that were available in this tribe of warriors—which is perhaps why we were hired—they were quickly automated and reduced to daily and weekly maintenance tasks. Let’s review these daily and weekly tasks because in the IDP world they are very important.

Daily Tasks

In the carrier’s network, the IDP8200s are not installed in the traffic path, but are attached to mirrored ports of Layer 2 switches. The reason for this is that the carrier is taking on the role of neighborhood watchman rather than traffic cop. The IDP8200s identify traffic that could cause problems for the carrier’s customers. When a threat is observed, the customer is notified that this threat is present. It is up to the customer to determine the action to be performed on this traffic, not the carrier.

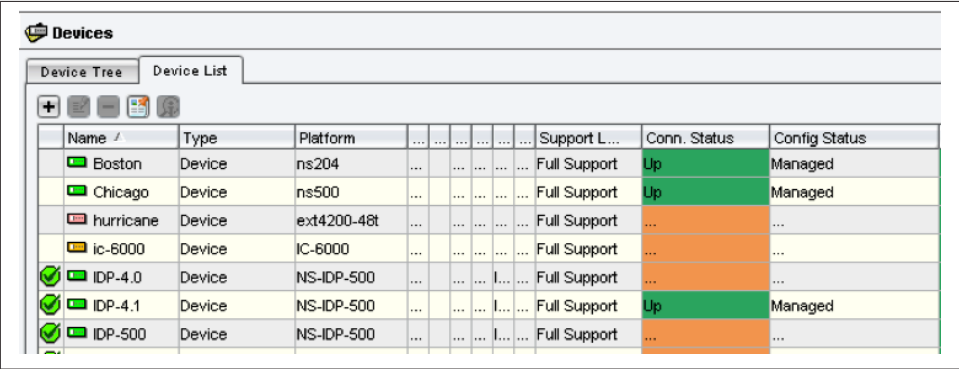


This is an area of much debate. If the carrier recognizes a threat and acts upon that threat, the customer can complain that the carrier is not delivering traffic as contracted. If the carrier recognizes a threat and does not act upon the threat, the customer can complain that the carrier is not providing protection. What to do?

The placement of the IDP8200s also does not affect the normal traffic flow, allowing the carrier to maintain “five nines” reliability and security monitoring. To maintain this high reliability, one of the daily tasks included verification that each of the devices was up and operational. This task can be performed at the NSM console or via the command-line interface (CLI). The NSM console shows whether the device is fully supported, if

it's up (operational), and that the configuration is managed, meaning that the device is communicating with the NSM server and that neither the device nor the server has updated the configuration. The CLI is used to verify that the interfaces are seeing traffic and that each is reporting attacks.

The NSM console for the device has a status column that indicates whether the device is connected to and communicating with the NSM server. The display looks like that seen in [Figure 2-6](#). Notice that the connection status is *Up* and that the configuration status is *Managed*. If the device's configuration needs to be updated, the status will indicate that the configuration on the device is different from the copy on the NSM.



The screenshot shows the 'Devices' section of the NSM console. It features a 'Device List' tab and a table with columns for Name, Type, Platform, Support Level, Conn. Status, and Config Status. The table lists several devices, including Boston, Chicago, hurricane, ic-6000, IDP-4.0, IDP-4.1, and IDP-500. The 'Conn. Status' column uses color coding: green for 'Up' and orange for 'Down'. The 'Config Status' column shows 'Managed' or 'Not Managed'.

Name	Type	Platform	Support L...	Conn. Status	Config Status
Boston	Device	ns204	Full Support	Up	Managed
Chicago	Device	ns500	Full Support	Up	Managed
hurricane	Device	ext4200-48t	Full Support	Down	Not Managed
ic-6000	Device	IC-6000	Full Support	Down	Not Managed
IDP-4.0	Device	NS-IDP-500	Full Support	Down	Not Managed
IDP-4.1	Device	NS-IDP-500	Full Support	Up	Managed
IDP-500	Device	NS-IDP-500	Full Support	Down	Not Managed

Figure 2-6. NSM status display

We used the NSM console to quickly see the status of all the devices, and then went to the command line to get the specifics for each device. The CLI commands are used to verify the proper operation of the device and to look at the interfaces and the attack table. The interface commands show the status of the interfaces and the amount of traffic that is seen on each. If there is a major difference in traffic between one day and the next, the issue is escalated to the network guys so they can see what is going on. You can see where traffic is flowing and often identify when equipment has failed over to the backup path. During our gig, the security guys (our tribe) worked hand in hand with other warriors from the network side. It was a proactive environment that made the network better for all involved.



I'm reminded of an experience decades ago when I first moved to Vermont, and our phones were served out of a mechanical central office. When you dialed a number, you could hear the switch chugging away connecting you. You could also hear when a bad piece of gear caused the call to drop. I made the mistake of reporting a bad piece of gear to the telephone company, and not only did they not understand what I was telling them, but they suspiciously wondered how I knew what was going on. Once a warrior, always a warrior.

The CLI command to check the interfaces is:

```
[root@idp] ifconfig
eth0      Link encap:Ethernet  HWaddr 00:ff:ff:00:00:ff
          inet addr:10.10.10.10  Bcast:10.10.10.255
          Mask:255.255.255.0
          inet6 addr: ffff::0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4548102 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5964299 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:487865628 (465.2 MiB)
          TX bytes:2605022861 (2.4 GiB)
          Base address:0xa020 Memory:d8c60000-d8c80000
```

This output is typical and has been reduced to show only a single interface. Each device is running between two and six interfaces. The output shows that the traffic is relatively equal input and output, and that there are no errors. We kept a list of the active interfaces and the relative traffic on each. The network deployment was in a primary/secondary arrangement with an IDP interface monitoring each side—this was how we could determine failovers. Also, a drastic increase in traffic on an active interface indicated an upstream change in the traffic patterns. All this information fed into the overall health of the network.

The CLI command to check the attack table is:

```
[root@idp ~]# scio var -s s0 sc_attack_table
sc_attack_table:
|           Attack Name           | #Hits |
|-----+-----|
SNMP:ERROR:INVALID-MSG-FORMAT    132
TROJAN:BACKORIFICE:CONNECTION    93
SNMP:COMMUNITY:ILMI              52
SSH:AUDIT:SSH-V1                 48
HTTP:SQL:INJ:CMD-CHAIN-1         34
SNMP:ERROR:CLIENT-RESPONSE       30
SNMP:COMMUNITY:SURECOM-RTR       26
HTTP:SQL:INJ:CMD-CHAIN-2         9
HTTP:STC:SSL:MD5-SIGNATURE       8
```

Again, the output is typical and has been shortened to show one processor and a very limited set of attacks. If each processor is showing attacks, then things are good. Traffic is flowing through each processor, each set of interfaces is assigned to a virtual router, and each is reporting as designed.

Juniper Networks is a zero-day security company, so there are security professionals diligently creating attack signatures on a 24/7 basis. The attack signatures can be seen at <http://services.netscreen.com/documentation/signatures/>.



This web page can be painfully slow to search, as all the signatures are on the top page. It is faster and less frustrating to save the page to a text file and use a local word processor to search the contents.

One of the daily tasks is to install the latest and greatest attack signature on the IDP8200s. This is performed via the NSM console, and is a multistep process. The first step is to update the attack signature on the NSM then push it to the IDP8200s.

Use the View/Update NSM Attack Database option on the Tools menu (Figure 2-7) to communicate with the Juniper Networks security server and determine whether your current version of the attack database is up-to-date, or if an update is required.

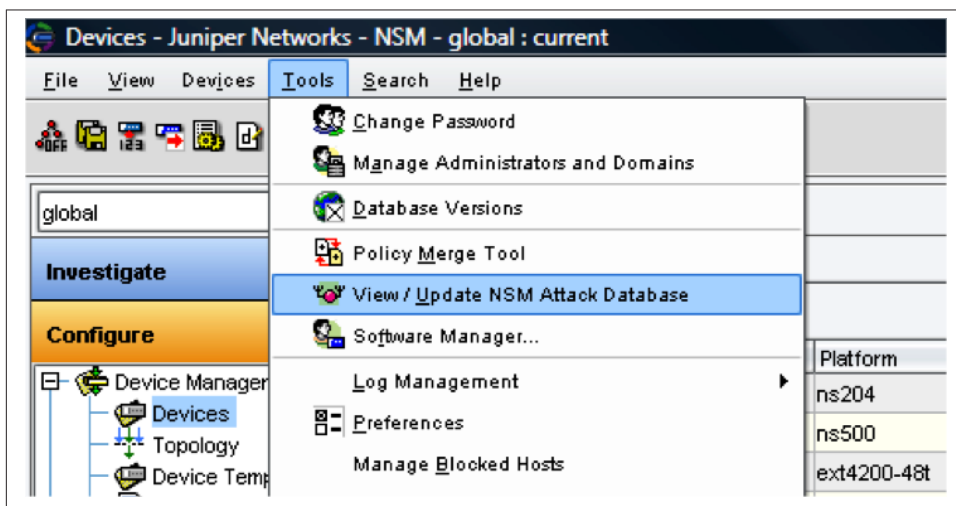


Figure 2-7. Updating the NSM attack database

When you request an update, NSM connects to the Juniper Networks attack server and downloads the most recent version of the database. As shown in [Figure 2-8](#), the completion message shows the version of the attack signature; in this case, the version of the signature is 1955. This information can also be retrieved from the NSM Job Manager screen for the completed download.

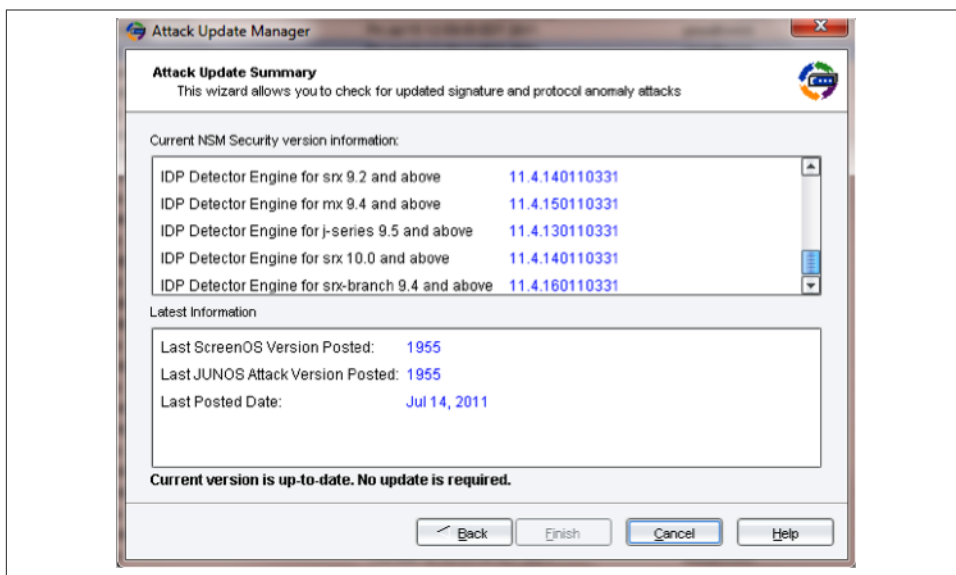


Figure 2-8. Attack update message

Once NSM is updated (it can take a couple of minutes), you can update the IDP8200s. [Figure 2-9](#) shows the menu item to use to update the attack database on the IDP8200s. You can check the attack database version on the device screen. (By default, the attack database version is the last column; the display was customized here so it was in a better position and could be seen on the first page.) The update operation can take a long time to complete, and in some cases, it does not complete. If this happens, it's time to take a look at the device to see what the problem might be. Often, we had to push a database to the IDPs twice, so watch what you are doing and don't lose focus.

Note that you may also receive warning messages about the attack groups being obsolete in the attack database. This does not cause a problem in loading the attack database, but says that the attack policy should be modified. If you take a look at [Figure 2-10](#), you can see that a Microsoft IIS attack for HTTP is now obsolete in the attack database.

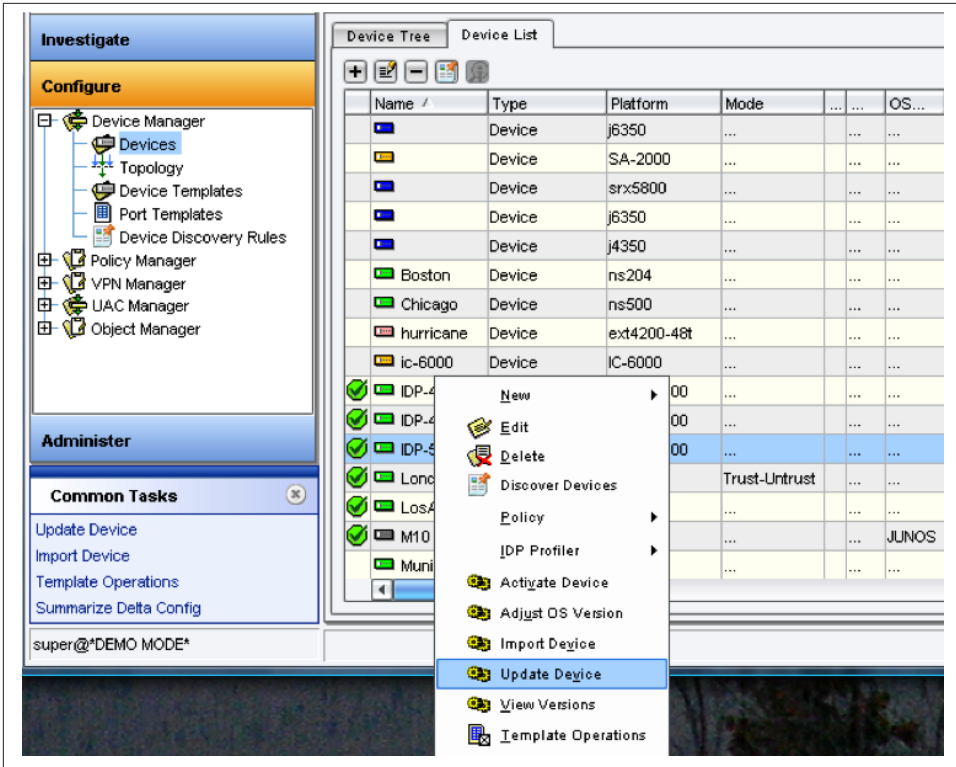


Figure 2-9. IDP8200 attack database update

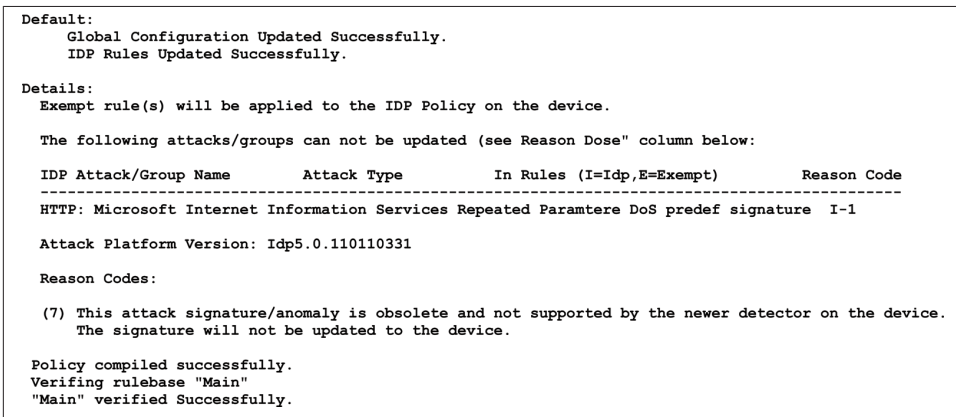


Figure 2-10. Attack update warnings

On the rare occasion that an IDP8200 does not receive an attack update after the second push, a health check has to be performed for that device. For whatever reason, the processor may be pegged, or the memory of the device may be full. In either case, the cause can be quickly determined on the CLI, and either the IDP can be restarted (processor pegged) or the filesystem can be cleaned up (memory full).

The processors of the IDP8200s typically ran in the 30 to 40% range for most of our devices, but some ran higher, and some lower. Using the Unix *top* command wasn't effective for the IDP8200s—it showed that the CPU was maxed out:

```
[root@idp ~] top
top - 13:44:01 up 38 days,23:19, 1 user, load average: 7 99,7.59,7.56
Tasks: 132 total, 10 running, 120 sleeping, 0 stopped, 2 zombie
Cpu(s): 75.3%us, 13.4%sy, 0.0%ni,11.2%id, 0.0%wa,0.0%hi,0.0%si,0.0%st
Mem: 15326044k total, 15093972k used, 232072k free, 296992k buffers
Swap: 4192924k total, 0k used, 4192924k free, 5545428k cached

  PID USER  PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 7300 root   10  -10 2824m 1.5g 5824 R   100  10.5 56156:12 idpengine
 7302 root   10  -10 2556m 1.4g 111m R   100   9.5 56132:43 idpengine
 7304 root   10  -10 2548m 1.3g 5824 R   100   8.8 56132:36 idpengine
 7310 root   10  -10 2823m 1.6g 30m R   100  10.6 56157:50 idpengine
 7346 root   10  -10 2832m 1.5g 5844 R   100  10.5 56184:16 idpengine
 9512 root    0  -20    0    0    0 R   100   0.0 56119:28 kjnetd
12520 root   10  -10 1916m 678m 4924 R   100   4.5 29300:56 idpengine
11814 root   20    0 11044 1484  936 S    1   0.0 554:57.01 peerPortModu
29264 root   20    0 12728 1080  804 R    1   0.0  0:00.12 top
```

After we had our first heart attack, one of us looked in the documentation and found that the correct way to check processor use is to use an SCIO command:

```
[root@idp ~]# scio idp-cpu-utilization
Current actual cpu utilization (CPU0): 30
Current actual cpu utilization (CPU1): 25
Current actual cpu utilization (CPU2): 40
Current actual cpu utilization (CPU3): 40
Current actual cpu utilization (CPU4): 30
Current actual cpu utilization (CPU5): 40
```

If the processors are running very high while the traffic is normal (check the interfaces to see the traffic volume), the IDP can be restarted. While this seems a draconian measure, for our environment (offline reporting only), it was acceptable. The restart command is:

```
[admin@idp ~]$ idp.sh restart
```

While this is not a normal case, now and then things may go crazy for no apparent reason. If you're reading this book, you should know that. Good warriors log these events, and if they have a device that is often in this state, they open a case and get the JTAC folks to take a look.

Unlike the CPU commands, the Unix memory and filesystem commands work just fine for the IDP8200s. If the filesystem is out of space, the new attack database cannot be loaded, so a little file maintenance is necessary. To see where the memory is being usurped, use the disk free command (I did not have an IDP8200 device with me at the time of this writing, so the following commands were run on my trusty J2320):

```
root@RTR_A% df -kh
Filesystem      Size  Used Avail Capacity  Mounted on
/dev/ad0s1a     460M  333M  122M   73%      /
devfs           1.0K   1.0K    0B  100%    /dev
/dev/md0        452M  452M    0B  100%    /junos
/cf             460M  333M  122M   73%    /junos/cf
devfs           1.0K   1.0K    0B  100%    /junos/dev/
procfs          4.0K   4.0K    0B  100%    /proc
/dev/bo0s1e     19M   25K   19M    0%    /config
/dev/md1        168M   15M  139M   10%    /mfs
/cf/var/jail    460M  333M  122M   73%    /jail/var
/cf/var/log     460M  333M  122M   73%    /jail/var/log
devfs           1.0K   1.0K    0B  100%    /jail/dev
/dev/md2        39M   4.0K   36M    0%    /mfs/var/run/utm
```

The `-kh` options cause the output to be displayed in 1024-byte units and the Meg, Kilo, and Gig units. As can be seen in this display, the *ad0* system is using 73% of the allocated storage capacity. If one filesystem is using more than its share of memory resources, the offending directories can be displayed with the disk usage command (again, run here on a J2320):

```
root@RTR_A% du -sh /var/home/*
975K   /var/home/lab
2.0K   /var/home/peter
```

This output shows that the directory *lab* has relatively high usage; cleaning up its files might solve the problem.

IDP Policies

Once the attack signatures had been updated on the devices and all were determined to be up and operational, it was time to get to the meat of the assignment—tuning the IDP policies of the IDP8200s. On a weekly basis, the top attacks were analyzed, researched for false positives, and categorized. The goal was to reduce the load on the processors of the IDP8200s and was accomplished by culling attacks from the IDP policies.

When we arrived onsite, the average processor load for the IDP8200s that were carrying traffic was approximately 80%. This load would spike under certain traffic patterns and cause the IDP8200s to miss traffic. We had work to do.

The processor load reduction process had two components: attack analysis, and a re-design of the IDP policy. The security policy is composed of multiple rulebases. The two

major rulebases are the IDP policies and the Exempt policies. If an attack is put in the Exempt database, no action is taken for that traffic (this is sort of a pass-through operation). The more attacks that are entered into the Exempt database, the fewer processor cycles are expended for that traffic.

To perform the attack analysis, the top attacks for the IDPs were gathered. Each of the top attacks was looked up in the attack signature database on the Juniper Networks site¹, and a determination was made about the attack. Those attacks that were determined to be benign were added to the Exempt database.

This process had a number of steps, many of which involved sitting around a conference table discussing hackers' principles and the threats posed by certain traffic.

This is where the warriors' experience and tribal coordination come into play. A level of trust, backed up by experience and plenty of reference material, is used to decide the status of these attacks. If beer and darts were available, I'm sure those would play a role in this process as well. I can't count the number of times that I've taken part in these onsite tribal meetings, not having a clue what the outcome would be when we went in but arriving at a solution before we left (or at least, a list of agreed-upon things to do). Sometimes, of course, I've been part of yelling matches between engineers, but that's not a network issue, it's a personality issue.

As stated previously, the attacks are logged to the NSM and STRM servers. NSM supports a set of "canned" reports that can be modified for the purposes of gathering the top attacks for the week. The easiest option is to select Report Manager/DI/IDP Reports/Top 100 Attacks from the Investigate page, as shown in [Figure 2-11](#).

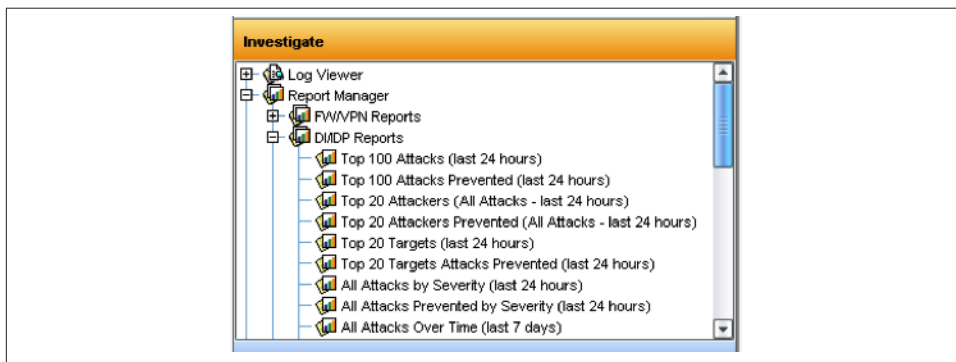


Figure 2-11. The NSM report menu

1. Remember the attack signature file that I suggested you download from the Juniper site? Use it here!

This report can be modified to gather information over the course of the week and to capture 200 rather than 100 data points. There is an icon at the top of NSM that looks like a set of gears on a bar graph. Clicking this icon displays a setup box (Figure 2-12) that allows the modification of the report. Change the duration from 24 hours to one week, and update the data point count from 100 to 200 (this captures more attacks). Accept the modifications and run the report. Again, this will take some time, maybe 10 minutes or so.

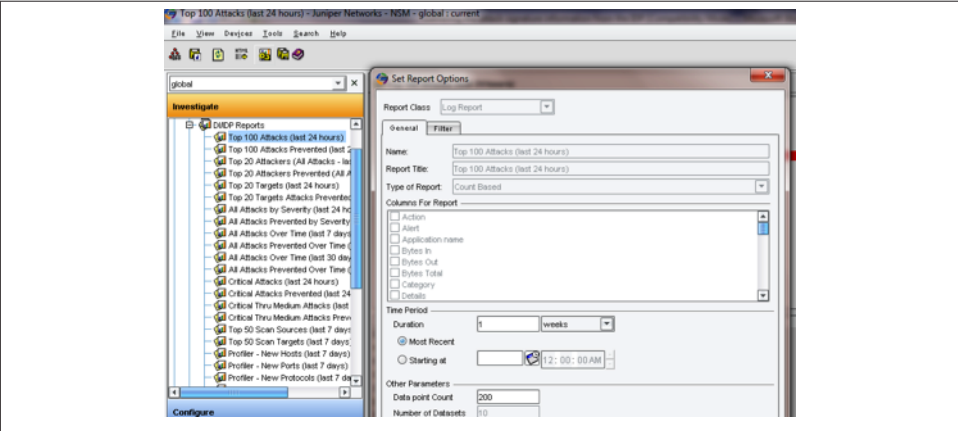


Figure 2-12. NSM top attacks

Once the report is completed, the top attacks can be looked up on the [Juniper site](#). Table 2-2 shows our top attacks. These attacks all had very high hit rates (not shown here, to protect the carrier's identity) and accounted for the top three signatures seen on all the IDP devices for a one-week period. Each of these signatures, while a possible attack signature, was determined to be benign in the carrier's network (the customer side of the network, not the management side).

Table 2-2. The top three attack signatures

Attack	Description
HTTP:PROXY:HTTP-PROXY-GET	This signature detects the presence of an HTTP proxy.
HTTP:AUDIT:HTTP-VER-1.0	This signature detects HTTP requests using HTTP version 1.0. Such requests are not inherently malicious.
HTTP:TUNNEL:PROXY	This signature detects HTTP-Proxy over HTTP. Attackers can send proxy connections over the HTTP port to circumvent firewall policies.

The first attack signature, HTTP:PROXY:HTTP-PROXY-GET, indicates a problem only if it is found on an enterprise network that does not rely on proxies; in the carrier's network, most customers do use HTTP proxies for security purposes. This attack could therefore be added to the Exempt rulebase without compromising security. The other two attacks showed the same type of attack signature and were also deemed Exempt.

When I said that this process is time-consuming and painful, I was not kidding. Each of the top 200 attacks was analyzed in this fashion, and a determination was made as to what to do with the attack signature. Thankfully, this process was only performed once or twice during the initial tuning of the attacks. Once the initial analysis was complete, the continuing analysis on a weekly basis was looking only for new and abnormal activity.

The process of adding Exempt attacks to the Exempt rulebase begins by opening the current IDP policy and clicking on the Exempt tab, as shown in [Figure 2-13](#), which shows an IDP policy called *Paris*.

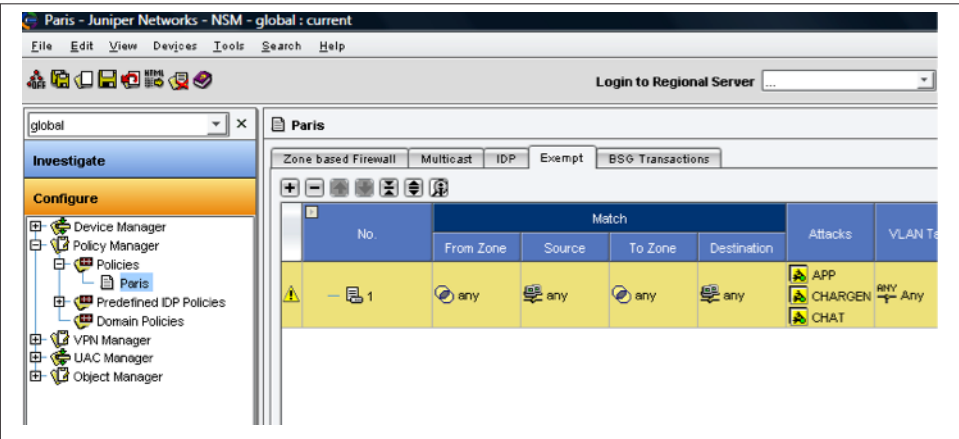


Figure 2-13. Exempt rulebase

The attack is added to the Attacks column of the first rule of the rulebase. Right-click on the Attacks column and begin typing its name. The attack will be highlighted in the left column. Once you've found the attack, click the Add button to move it to the right-hand column ([Figure 2-14](#)).

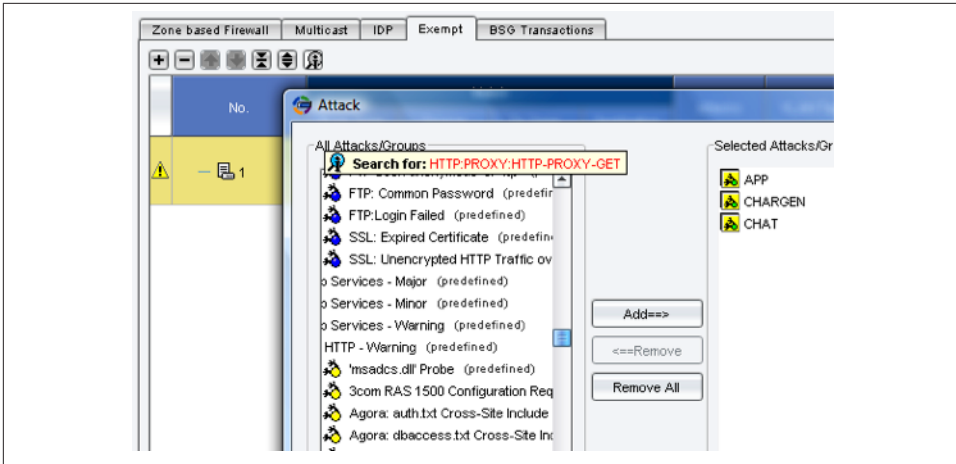


Figure 2-14. Adding attacks to the Exempt rulebase

Save the policy by clicking on the floppy disk icon at the top of the page. At this point NSM is changed, but the IDP devices are not. The IDP devices have to be updated. To do this, use the same process as that used for the attack database updates.

Rulebase Optimization

When we first arrived at this assignment, a number of open cases were delivered as part of the handover process. One of them was that the IDP8200s with the most traffic had CPUs that were pegged at 100% during the busy periods. These were beefy boxes and should not have been working so hard with the number of interfaces that were present. The devices can each handle 10 Gbps of traffic, and we were seeing only 5 to 6 Gbps. With help from JTAC, we took a closer look at these devices and determined that the devices were without defect, but that the policy that was being used was causing an overload.

After some deeper inspection, a little head scratching, and some conference calls between the tribe, what we had was a policy that was duplicated for each of the eight VLANs present on the devices. I thought initially that different policies were being used for each VLAN, but that was proved to be wrong. The tribe determined that the same policy could be used over all the VLANs.

By creating a single generic policy that was used for all the VLANs, we reduced the overall size of the rulebase by a factor of seven (making it one-eighth the size of the original rulebase).

We loaded this new policy on a single IDP8200 that was in a high-traffic area and ran the policy for a week. The peak CPU use was in the 30% range, with the same capture rate for attacks as prior to the policy change.

The result? No lost traffic and processor load reduced from 100% to 30%, all by using proper policy management.



In other chapters, I preach the KISS (Keep It Super Simple) principle. Here is another case where added complexity created a problem that didn't need to be there. KISS please! The simple policy was as effective as the more complicated ones and performed better with the same traffic.

This same policy has now been added to all the IDP8200s, and everybody is happy.

Other Tasks

From time to time, other tasks must be performed to keep the IDP8200s up to par. These include updating the detector engine, updating the OS version, and adding individual attacks to or deleting them from the attack group policies.

Updating the detector engine

The software detector engine controls the IDP process. It runs the attack signatures and determines the threats and actions. Juniper Networks updates the detector software on a quarterly basis. Two versions of the detector engine are available, the Netscreen version for IDP appliances and the Junos version for the SRX IDP feature set. The detector engine software is updated through NSM.

Launch the update wizard using the menu option shown in [Figure 2-15](#), and it will walk you through the update process. Identify the devices to be updated, as shown in [Figure 2-16](#), and click the Finish icon. NSM contacts the Juniper Networks servers, downloads the detector engines, and installs them in the IDP appliances. Once the job is complete (use the Job Manager to verify that the update is complete), the devices are updated per normal operation.

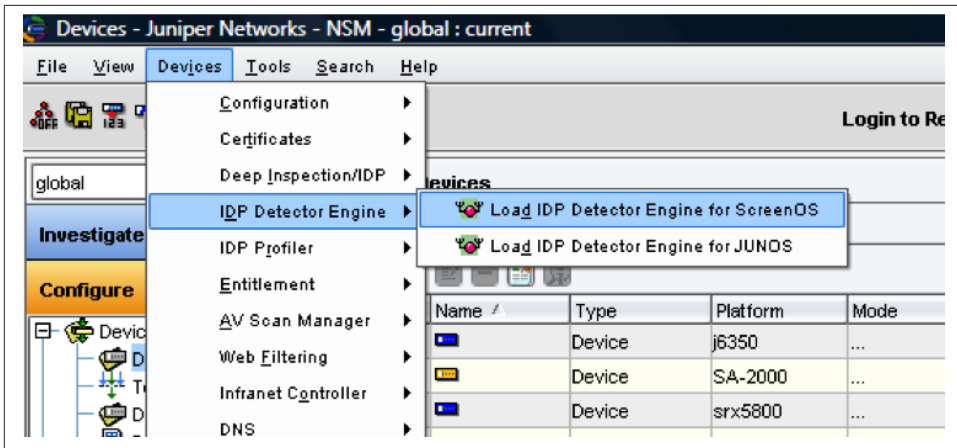


Figure 2-15. Launching the detector engine update wizard

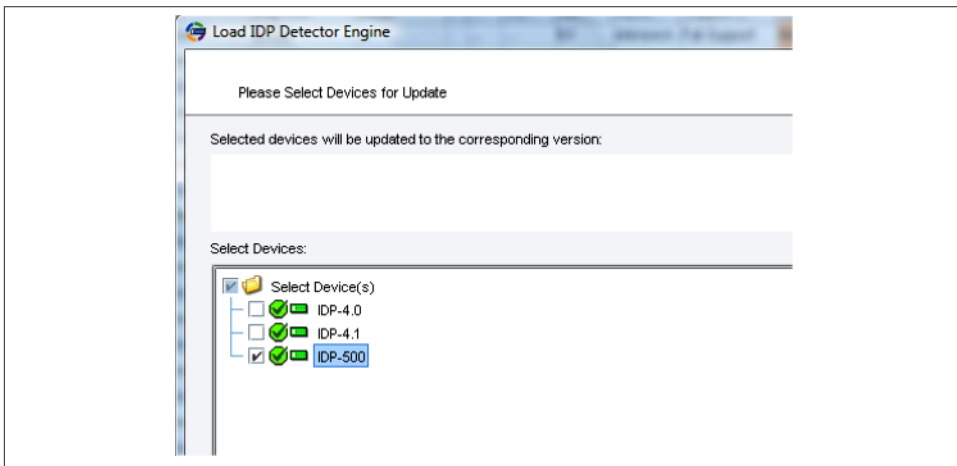


Figure 2-16. The device wizard

Updating IDP appliance OS

On some devices manufactured by other network equipment vendors, updating the operating system takes an act of congress, a PhD, and a contract that costs the same as a new car. For these IDP appliances, the cost of the new car is for the support contract, and the other two requirements are not needed at all. The OS update is performed from NSM and requires nothing more than a couple of clicks.

Locate the device to be updated in the device window, right-click on it, and select the Adjust OS Version option from the context menu (Figure 2-17). This opens a dialog box that determines the current OS and asks what supported OS can be updated to the device.

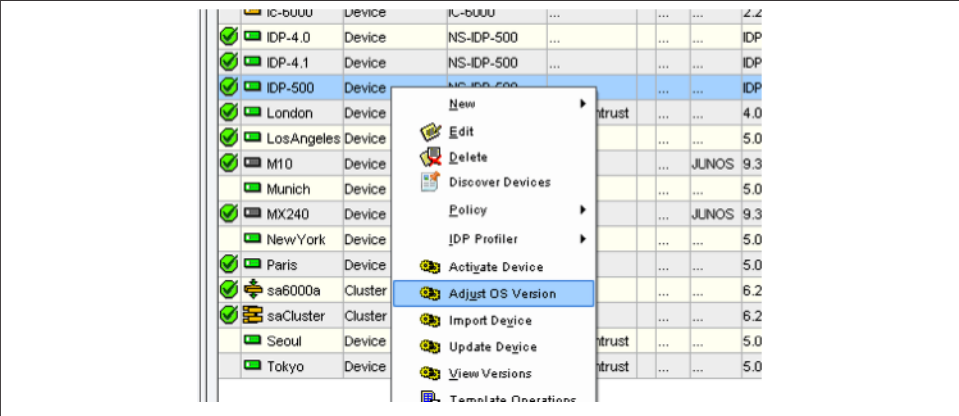


Figure 2-17. Device OS update

In this example, the current version of the OS is an ancient IDP4.0, and the only supported upgrade is OS IDP4.1. Selecting this version of the OS and clicking on the Finish button (Figure 2-18) creates a job that updates the operating system on the device.

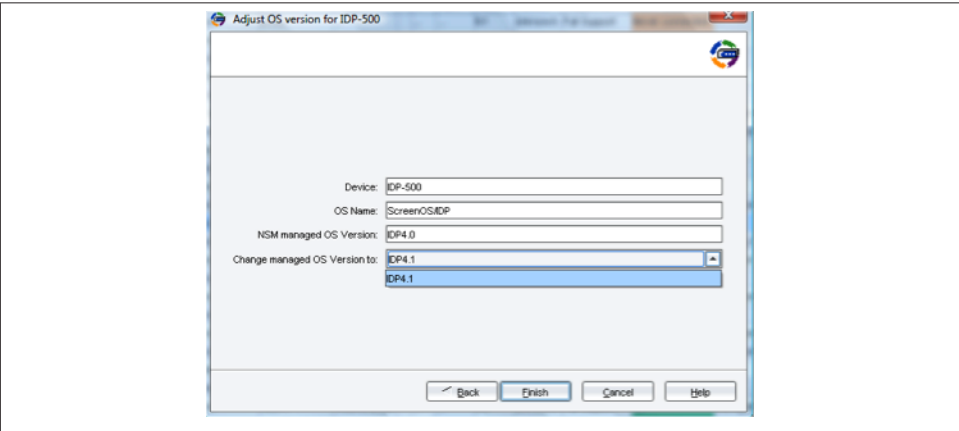
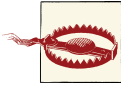


Figure 2-18. OS update dialog



NSM and device operating systems are in constant flux. The latest version of NSM is typically one or two device operating system versions behind. If you upgrade to the newest version of the device code, you will not be able to manage the device under NSM. To keep things in management by NSM, only upgrade to the versions that are supported by NSM, not what is available on the Juniper Networks website.

Updating attacks

On a day-to-day, week-to-week basis, and for no other reason than because someone has said so, the attack signatures that are referenced in the IDP policies have to be modified. If a predefined policy is used, these cannot be modified directly, but they can be copied to a custom policy and modified in that form—not the easiest method, but when things need to be done, the ugly way might be the only way possible. In the following example, an attack found in the weekly attack reports is to be deleted from the policy rather than added to the Exempt rulebase (this might be the case when the attack is obsolete).

The first step is to find the policy in the referenced attack groups and delete it. If the groups are minimal, and the IDPs are newly installed, this won't be a problem: simply look at the members of the attack group and locate the attack in question. In most cases, though, the list of attack groups that are used and the list of attacks per group will run to several pages, and it might take a while to locate the attack.

NSM has a solution to this time-consuming process, called the *Find Usage query* (Figure 2-19). If you can find the attack in the general listing, right-click it and select Find Usage, and NSM will list the custom attack groups where the attack appears. Very cool!

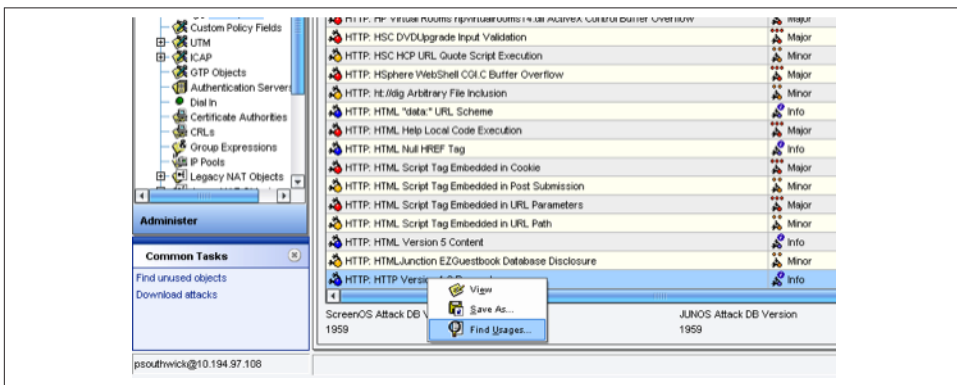


Figure 2-19. Find Usage query

In this case, the attack is used in a number of groups, as shown the results of the query in [Figure 2-20](#).

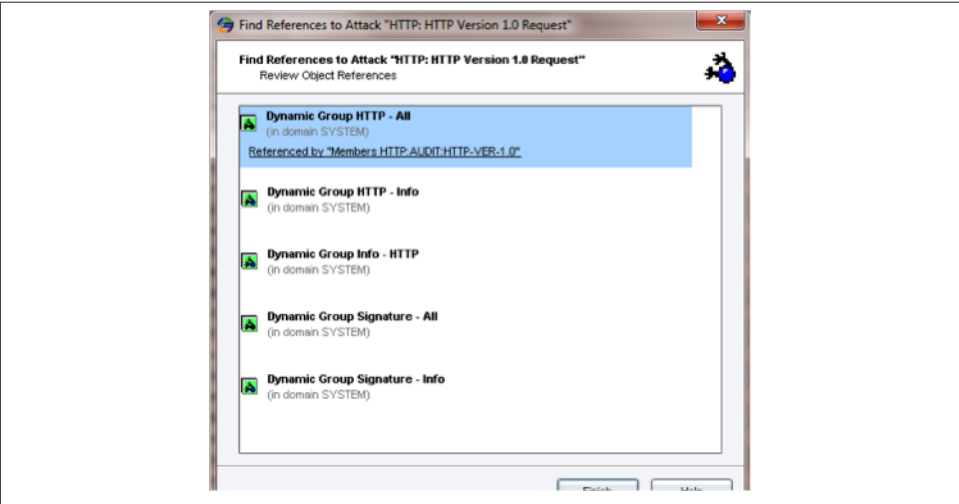


Figure 2-20. Find Usage query results

Once the attack groups are known, finding the attack in a group is easy with the use of the search mechanism (type the name of the attack with the group highlighted). Edit the group and remove the attack from the group as shown in [Figure 2-21](#).

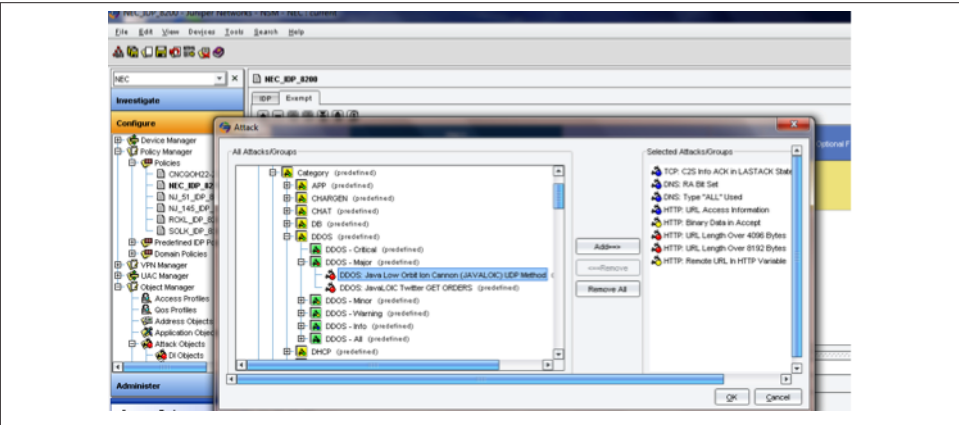


Figure 2-21. Removing an attack

Conclusion

This assignment allowed me and the other warriors involved to learn some very important lessons about maintenance, IDP, and the carrier environment.

Maintenance is as much a warrior's job as any design and implementation assignment. The general knowledge and expertise that make a network warrior can be applied to improve the processes and the operation of devices that have been fielded in the same manner that these tools are used for new network implementations. That old army adage about only 1 in 10 soldiers being on the front line does not mean the other 9 are any less important. As a matter of fact, without the other nine, the tenth soldier cannot do his job effectively. This assignment offered me the time and the opportunity to get to understand a totally different element of network engineering.

The IDP line of devices from Juniper Networks will one day be absorbed into the SRX product line: that handwriting is on the wall. The operation, setup, and maintenance of the standalone IDP appliance and its firewall manifestation are very close to one another. Anything learned on the IDP appliance can be moved to the firewall with little modification.

The upkeep of any IDP device is an ongoing effort; the recommended and predefined policies are easy to initially install, but the effort is in the tuning for performance. The population of the Exempt rulebase and the elimination of obsolete attack signatures is an ongoing activity. Each attack update causes some churn in the database that should be dealt with. We saw that the performance of the IDP appliance is greatly affected by the size and organization of the rulebases. This effect will only be increased when the SRX is the IDP platform.

Another lesson learned is that with a combination of CLI, NSM, and web interfaces, all of the maintenance tasks can be carried out fairly easily, and scripts can be created to automate these tasks (not my strong suit, but hey, we all have our specialties).

NSM is a great logging tool and has the capabilities to manage a network full of IDP devices. This tool will soon be replaced in the Juniper Networks arsenal with the Junos Space platform as it evolves. The tasks learned on NSM might not be portable to Space, but warriors should be able to adapt.

The final lesson had to do with the use of IDP on the customer side of the wireless carrier's network. Unlike with the use of IDP in an enterprise, where attacks are blocked and offending traffic is stopped in its tracks, here the IDP's role is to alert the carrier that a customer's traffic might be compromised. If a customer requests assistance, the carrier can step in and block the traffic (to a certain extent, given the offline mode of operation). The IDP deployment is more like a neighborhood watch program than a police program; all we can do is watch and listen.

By the way, the amount of traffic and the quantities of attack signature hits were an eye-opener for this warrior. I have worked on wireline carriers and ISPs, but the sheer amount of wireless traffic is daunting. When new services are added and new generations of services are included in the products, the traffic jumps significantly. Any security device working in this environment has to be *carrier grade* and must be able to handle the specified loads.

Luckily, the IDP8200 is up to the task, given a little care and feeding.

Data Center Security Design

In this chapter, a customer engagement is recreated that includes the design and testing of a data center security solution. The security is provided by Juniper Networks SRX5800 Series Services Gateways (SRXs) embedded in a Cisco infrastructure. To say the least, this engagement tested the technical and political skills of our small Juniper Networks warrior tribe: competing tribes had to work together to meet the demands of the customer, and bury the hatchet, so to speak, for the betterment of the project. It was a tough engagement, but in the end, it allowed Juniper Networks to get a foothold in a Cisco shop (something that has occurred more and more regularly in the last decade).

Many years ago, in a previous life, I designed equipment for a company that had two competing product lines. My product line was the cash cow, an old technology device that was feature rich and established. The other product was based on newer technology but was not as feature rich and definitely not established in the marketplace. When the company responded to requests for proposals, the two groups were encouraged to compete for the contracts. From these competitions, I learned the political side of technology—not always a nice place to be, but just as important as the technical side. We used incomplete results and misleading descriptions to make the other side look bad, while avoiding any hitches that might have cropped up in our design.

When I started down the path of a network warrior, I subscribed to the code that the customer's satisfaction was the only real goal of any engagement. If reaching that goal required the use of other vendors' equipment, then that was what happened—no obfuscation, no games. That is how our small tribe entered this engagement, and we were very surprised to find that not all warriors abide by the same code.

I have read repeatedly that multivendor networks (and data centers, for that matter) have greater reliability, greater endurance, and a lower total cost of ownership. And I believe that, drawing from the hands-on history of deploying them that this book partially documents. So can Juniper and Cisco live together? My tribe said yes and the other tribe said no. This was going to be interesting.

Discussion

The engagement was to design and test the connectivity for a new data center. The customer was a government agency, and a new data center design was needed to replace a distributed, outdated, and painfully slow private line solution. As part of the effort, multiple smaller data centers across the agency were to be consolidated into the new facility. The goals of the project were:

Security

The design should create an environment where the information in one department was secured from all other departments.

Connectivity

The design should allow full connectivity between all departments and their information, as well as all other departments—i.e., full any-to-any connectivity.

Survivability

No failure of a single piece of equipment or link should cause a loss of connectivity.

The departmental information was stored on virtualized servers in two data centers that were geographically separated (East and West). The virtualization allowed the backup and recovery of any data on any physical server in the data center. Synchronization and replication between data centers was another of the design goals of the engagement. The individual departments of the agency were segregated into *information silos*, in data center speak. The virtual servers in a silo could communicate freely, but all inter-silo communications had to be secured.

The user communities (clients) of the departments were scattered over a wide area. They accessed the data center network via private wide area networks, metropolitan area networks, or the Internet. All access to the data center network was protected by a set of access firewalls (actually a cluster of Juniper Networks SRXs, but that is a different deployment).

Two clusters of Juniper SRX 5800s enforced security for traffic in the data center. Each SRX was fitted out with two four-port 10 Gbps interface cards and a pair of services processing cards (SPCs). The SPCs do the heavy lifting for the firewall sessions in the

SRX, allowing the line cards to focus on passing traffic. The SRXs were installed in a multitier data center design, as shown in [Figure 3-1](#). All traffic to the servers from the user community had to pass through the firewalls. All traffic between the various departmental silos also had to pass through a firewall.

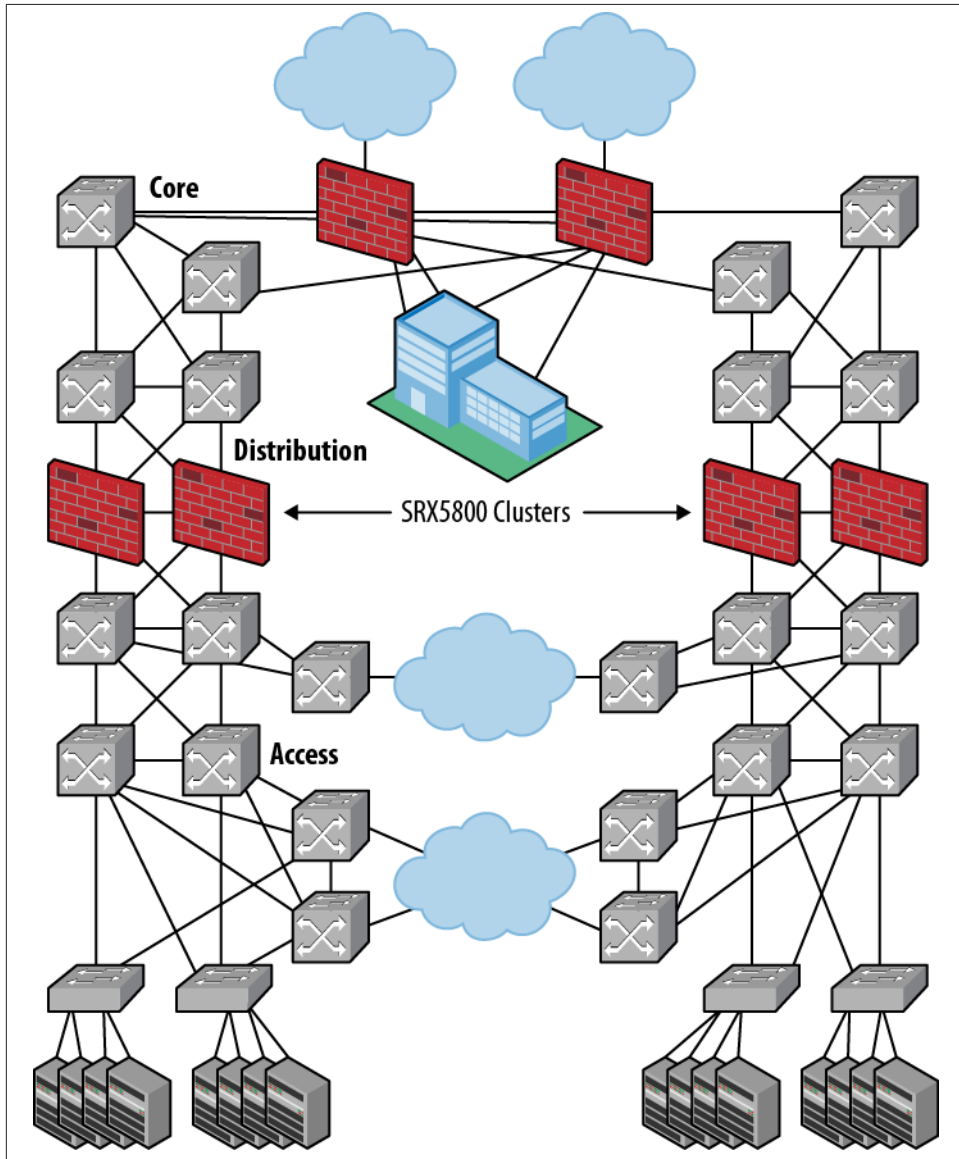


Figure 3-1. Data center overall design

For those who have not attended a Cisco class or attained a Cisco certification in the last couple of decades, our deployment mimics the classic Cisco data center design shown in [Figure 3-1](#) and contains the classic tiers:

Core

This is the top tier of routers in the design. The core tier connects the various users to the data center. The core is connected to the egress firewalls, which provide connectivity (secured) to the WANs, MANs, and Internet user communities. Because of the broad nature of the data center users, the security policies for the egress firewall between the users and the data center are somewhat open. Traffic suspected to be part of a spoofing or denial of service attack and unknown users are blocked; all other (known) users are allowed to access the data center. In the other direction, the data center has full access to any internal user and for most instances, the Internet. There is full-mesh connectivity between the routers and the egress firewall.

Distribution

The middle portion of the design comprises the distribution layer. It provides for full connectivity to the core tier (top of the diagram) from the access tier (bottom of the diagram). The distribution tier is broken into three sublayers: upper, middle, and lower. The upper distribution sublayer routers connect to the core; the firewalls form the middle sublayer, while the lower sublayer connects the East and West data centers to one another and to the access tier. Full connectivity is provided between all the sublayers to assure survivability.

Access

The lowest tier in the design is the access tier. These routers and switches provide connectivity to the server farms and to the synchronization and replication WAN between the data centers.

The tiers are duplicated on the East and the West side of the network. The interconnection points between the tiers are at the egress firewall, at the distribution tier and via the synchronization/replication WAN.

While the connectivity seems straightforward in the network diagram, there are a few aspects that the diagram does not show:

Active/passive operation

The network is mirrored between East and West. By means of route manipulation and server virtualization, the East side of the network is active while the West side is in a hot standby state. This arrangement reduces the chances of asymmetrical routing (going up the East side and returning the West side) while maintaining very high survivability. Depending on the failure type, sessions can be maintained during a failover.

Routing

All routing between the data centers and between the tiers is a quasistatic arrangement using the border gateway protocol (BGP) in its exterior peering (EBGP) mode of operation. This allows all routes between tiers and between the East and West stacks to be controlled. The use of EBGP also provides for redundant paths and failover between paths when a failure occurs. The use of EBGP and interface peering eliminates the need for an interior gateway protocol (IGP). This reduces both the complexity of the network and background traffic between routers. The engineered nature of the routing assures that the traffic is symmetrical in nature. External routing in the data center network is performed by a set of default routes generated by the egress firewalls. This assures that all traffic has a way to the users and the Internet without downloading the entire route table (ouch).

Symmetrical routing

With a network as survivable as this one, it is very possible to have traffic split between North, South, East, and West. However, this can lead to two problems. The first is the reassembly of sessions for the firewalls. While the SRX can reassemble a session within a cluster (out on *node1*, return on *node0*), it cannot reassemble traffic between clusters (out on East, return on West). In these cases, the traffic is blocked because it is out of sequence. The other limiting factor is the load balancers. Although not shown on the diagram, each access tier uses load balancers for traffic to the server farm. For the load balancer to operate correctly, it must see both sides of the session. The East and West load balancers are not interconnected and therefore cannot reassemble sessions. What this all means is that asymmetrical traffic (out on East, return on West) must be eliminated from the design.

Inter-silo traffic

All departmental information is stored on virtual partitions in the server farms. Access to this information is allowed to users who need it by the data center SRXs. When one server must interact with another server, the firewall might also see that traffic—if the servers are in the same department (silo) the access tier allows direct connectivity, but if the servers are in different departments, the firewall is brought into the picture. This is accomplished by the use of virtual LANs (VLANs) and the default gateways. All virtual services are assigned to a VLAN, and the gateway for the virtual server is the router in the upper sublayer of the distribution tier. The traffic must pass through the gateway (and through the firewall) to be routed to another silo (see [Figure 3-2](#)). This setup assures that all interdepartmental traffic is secured.

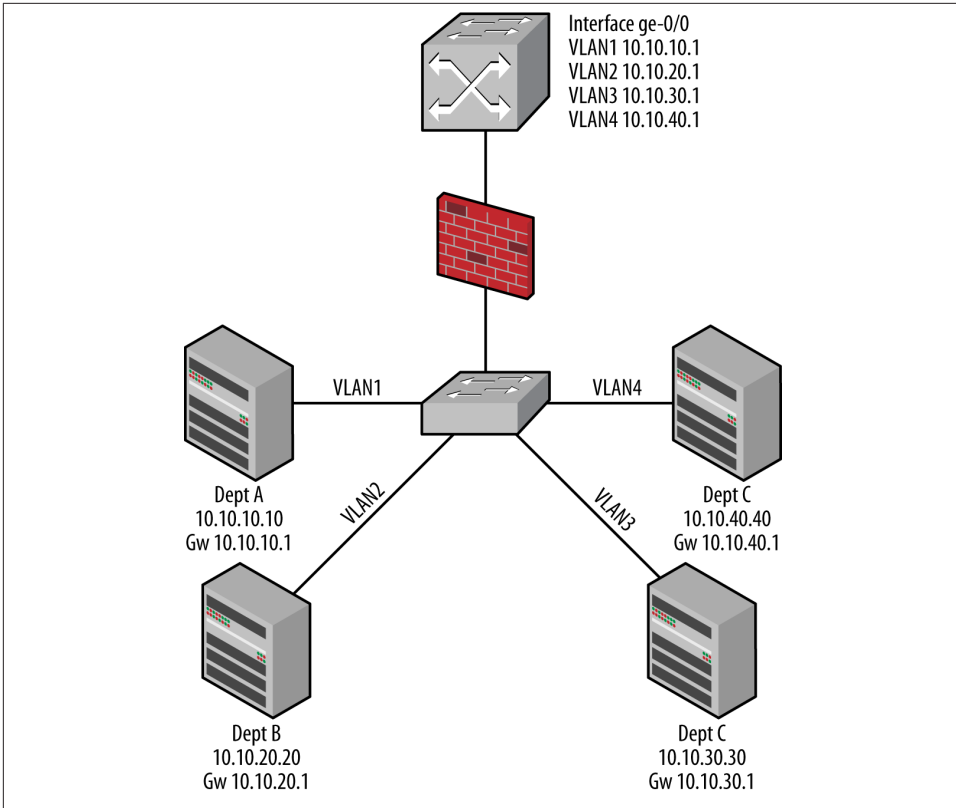


Figure 3-2. Silo connectivity

Design Trade-Offs

The overall design of the network was decided well before us warriors set foot on the customer's site. That happens. As a warrior, you must take what you're dealt and make it work, grumble as you may about the original architect.



If there had been the possibility of a true design war, this tribe's solution, using the **Juniper design guide**, the Juniper Networks data center fabric (QFabric), and the Juniper Networks virtualized security solution (vGW), would have saved money, space, and power!

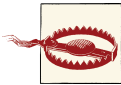
The design trade-off involved determining what role the SRX would play in the design. The SRX was a "forced" concession by the Cisco tribe to provide a multivendor solution to the data center, and as such, they felt that the role of the SRX should be minimized as much as possible.

Knowing the capabilities and strengths of the SRX, and especially the SRX5800, we knew that the proposed design could be reduced and improved by expanding the role of the SRX. The distribution tier of the design could be reduced from the proposed 14-device router/switch/firewall configuration to a pair of clustered SRXs (see [Figure 3-3](#)).

The SRX has the routing capabilities (EBGP) and the interface capabilities to handle all the requirements of the distribution tier. The use of a single device would reduce the complexity of the network and improve the survivability and manageability of the design, not to mention lowering the total cost of ownership.

Members of our tribe practically had to be physically restrained when the response to our proposal was received: routers route, switches switch, and firewalls filter, and the devices should not be confused, combined, or otherwise altered. So much for an open mind and keeping the customer focus! Clearly, we warriors do not all live by the same code.

In this engagement, the Cisco tribe had political ties to the decision makers that our group of warriors lacked. In the end, the proposed Cisco solution was adopted and the role of the SRX was simply one of filtering in transparent mode (a stupid bump in the wire!).



In this occupation, one cannot always take being a warrior too seriously —tomahawks and arrows are not supposed to be among the weapons we wield. Terminals and sniffers, yes; deadly force, no. When things go very wrong for nontechnical reasons, it's usually time to take a step back, have a cold beer, and just agree to disagree.

Decision

For those who have not read the fine print on the SRX, it offers what is called a transparent mode of operation. This is a holdover from the Netscreen operating system, where a firewall could be put into a Layer 2 mode and be transparent to routers and other devices on the network. It was meant to allow the device to be installed in an existing network and provide security, without having to readdress the other devices. The device entered the transparent mode when the last Layer 3 interface was deleted from the configuration. In the Netscreen version, there are even special zones for Layer 2 interfaces.

In the Junos version of transparent mode, there are a few restrictions. It is also only available on certain SRXs (SRX240s support bridged interfaces but not true transparent mode operation; refer to <http://bit.ly/VNFqOm> for a full description of the Layer 2 capabilities of the SRX).

As with the Netscreen devices, when no interfaces are configured as Layer 3 (*family inet*), the device is in transparent mode (the exception is the out-of-band *fxp0* interface).

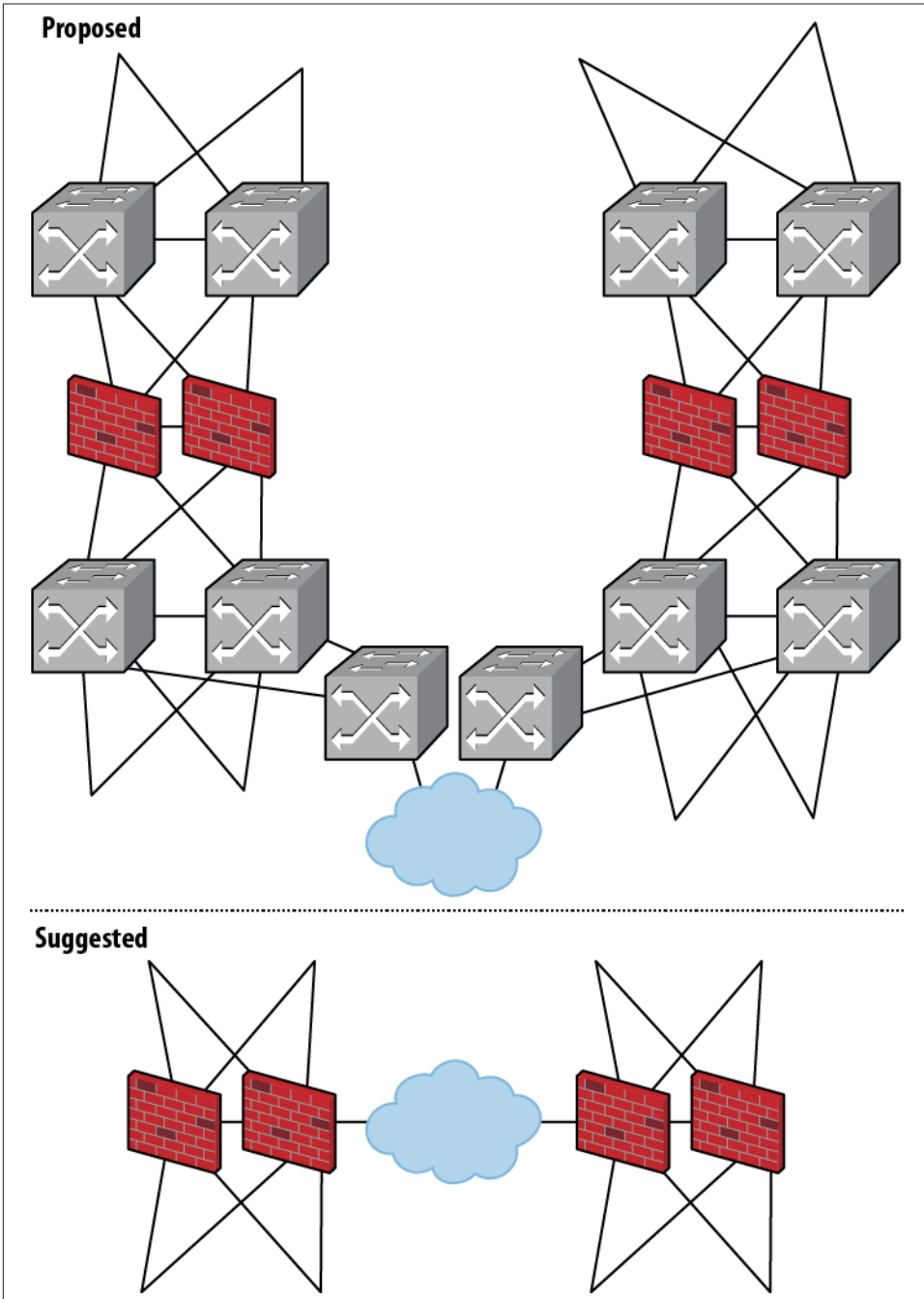


Figure 3-3. Distribution layer trade-off

The interfaces are grouped into bridge domains (broadcast domains from the old Ethernet days) and are assigned VLAN IDs. Because of the use of VLANs and no routing tables, the virtualization capabilities of the SRX are not needed. Essentially, the SRX becomes a VLAN switch with two interfaces per VLAN and security filters on each pair (see [Figure 3-4](#)).

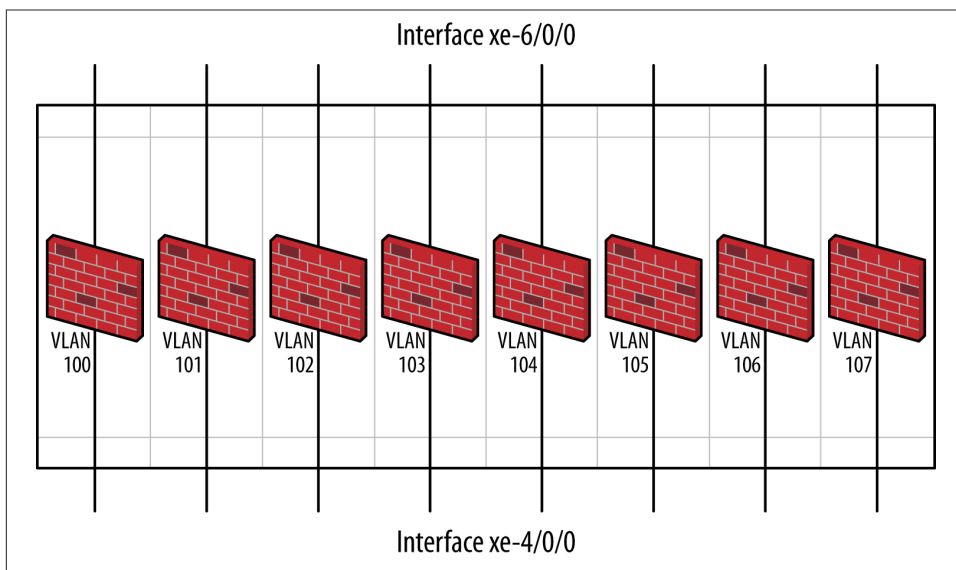


Figure 3-4. SRX in transparent mode

In [Figure 3-4](#), two physical interfaces are defined: *xe-4/0/0* and *xe-6/0/0*. These interfaces have multiple logical interfaces, each assigned a different VLAN identifier. Each logical interface pair (e.g., *xe-4/0/0.100* and *xe-6/0/0.100*) is assigned to a single bridge domain. Each only sees traffic from the two logical interfaces. If these interfaces are installed in two different security zones, then the policies written only need to allow or deny traffic between those interfaces. It creates a very simple and clean configuration, always a warrior's expression of good networking technique.

Configuration

The configuration of the SRXs for this engagement underwent a number of revisions. They were:

- Active/passive operation with redundant Ethernet (Reth) interfaces
- Active/active operation

- Active/active operation with redundant Ethernet interfaces

The reasons for the multiple versions of the configuration had to do with misunderstanding of the capabilities of the SRX and a change of code versions for the SRX. The initial requirement for the SRX was an active/passive operation for the cluster. As in the other portions of the network, all traffic passes over the active interfaces and processors, while the backup equipment, while hot, only sees traffic in the event of a failure.

The version of code that was available for the initial configuration did not support active/active clustering in the transparent mode of operation, so this configuration met the customer's requirements for both traffic handling and error recovery.

The problem with the initial configuration was that the other devices used in the distribution tier did not operate in a Layer 2 switched mode. As such, they could not recover when a failover occurred between the active and backup Ethernet interfaces. This caused us to rethink the configuration and arrive at one of active/active operation without the use of redundant Ethernet interfaces.

This configuration displayed connectivity problems that required all active interfaces to be installed on the same node of the cluster. While meeting the connectivity requirements of the customer, the design required additional line cards to be used, with the added costs associated with each node. Also, the added complexity of the solution would have led to maintenance and troubleshooting issues down the road.

The final configuration was a compromise between the two previous configurations. The code was updated to include the active/active capabilities for the transparent mode, while single-legged redundant Ethernet interfaces were used to provide the connectivity. This combination led to a working solution that met the requirements of the customer, while providing a manageable solution.

We'll look at each of these configurations in the following sections.

Take One Configuration: Clustering

Clustering an SRX is akin to stacking a switch. The two nodes of the cluster create one larger firewall with twice the number of card slots and two processors. The configuration for clustering has three main components: the chassis configuration, the group configurations, and the interface configurations.

The cabling for clustering is dependent of the device type. The cluster cabling for the SRX5800 is shown in [Figure 3-5](#). This diagram shows a control connection and twin data (fabric) connections. The use of dual control links is only possible if a second routing engine is installed in each node, considering the location of the nodes and the

possibility of a link failure, this added expense did not seem necessary. The control links are connected to the port 0 on SPCs that are in the same location in each node. Shown in Figure 3-6 are the SPCs in slot six (the eight physical slot in the chassis from the left starting to count at 1) in each node.

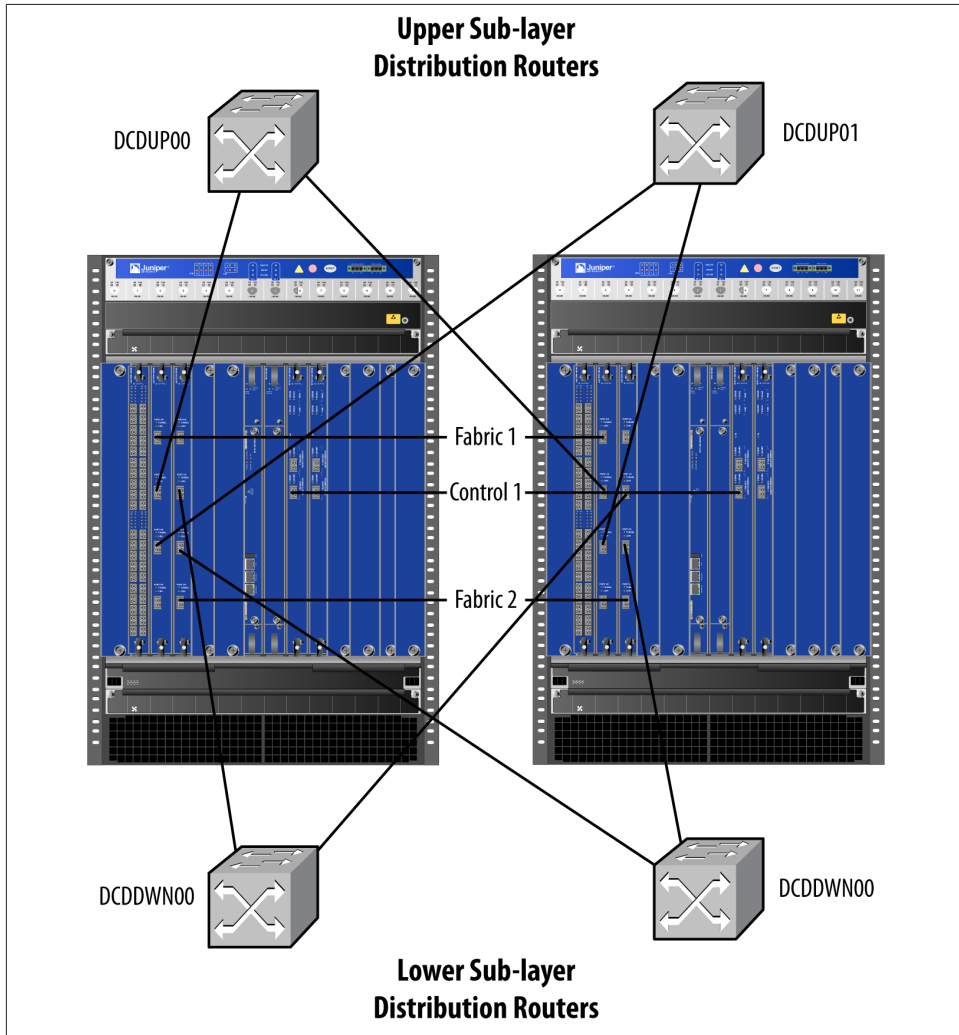
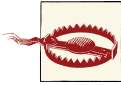


Figure 3-5. SRX5800 connectivity

The fabric links can use any *ge* or *xe* ports on the device. In the diagram, the ports from two different cards are used (*xe-2/0/0*, *xe-3/4/0*, *xe-14/0/0*, and *xe-15/4/0*) for the fabric links. The fabric links pass traffic between the two nodes, and they also pass some control-type traffic.

Confused yet on the numbering of the slots? When a pair of SRXs are clustered, the slot identifiers for the cluster start on node 0 and are extended to node 1. In an unclustered SRX5800, the slots are numbered 0 through 11 for input/output cards (IOCs), and the numbering skips the two middle slots, which contain the routing engine and the switch control cards. When the second node is added to the cluster, the numbering continues on that node at slot 12 and proceeds through slot 23. The first interface on the last card would be addressed as *ge-23/0/0*, while the first interface on the first card of the second node would be *ge-12/0/0*.



For interface numbering, pull the manuals and/or take a look at the silkscreening on the cards to see how the interfaces are numbered; no two are the same, and this warrior has been wrong 50% of the time.

Once the cabling is done, the configuration is needed. Except for the initial configuration steps, only the active node is configured. The standby routing engine is updated over the control channel.

The initial configuration sets the node identifiers and the control ports. Once this configuration is added to the active device (*node0* in our case), an operational command is given to put the nodes in clustering. From that point on, all configuration is performed on the active routing engine of a dual-chassis device. Here are the steps for the initial configuration:

1. On the node that will be the active node (*node0*), add the following configuration stanzas:

```
set chassis cluster control-link-recovery
set chassis cluster control-ports fpc 3 port 0
set chassis cluster control-ports fpc 9 port 0
```

2. The control link recovery stanza allows a node that has a failure to reenter the cluster without manual intervention. Once that information is committed, enter the following operational commands on both the nodes:

```
peter@node0> set chassis cluster cluster-id 1 node 0 reboot
peter@node1> set chassis cluster cluster-id 1 node 1 reboot
```

These commands reboot the SRXs and enter them into a cluster. The active routing engine (RE) looks for the standby RE on the control ports defined, and presto, a cluster is formed.

3. A construct of clustering is the redundancy group. All assets in a redundancy group switch from one node to the other at the same time. The switchover determination is made based on the node priority for the group and the errors that have occurred that change the priority. If a group's priority is set to 100 for the active and 50 for the standby RE, then an error worth 51 has to occur for a switchover to happen.

The error counts are assigned to interfaces. A processor failure wipes out the counter for that node. In our initial configuration, there are two redundancy groups: one contains the processors and the other the redundant Ethernet interfaces. The configuration for these redundancy groups is:

```
set chassis cluster redundancy-group 0 node 0 priority 254
set chassis cluster redundancy-group 0 node 1 priority 10
set chassis cluster redundancy-group 1 node 0 priority 254
set chassis cluster redundancy-group 1 node 1 priority 10
```

The failure-induced switchover is nonrevertive unless the *preempt* command is added to the configuration. In our case, the fewer switchovers the better, so a non-revertive failure is OK.

4. The next thing to configure is the fabric interfaces. These interfaces are used for internode traffic and some control information. In our setup, two fabric links are used. The configuration for the fabric links looks like this:

```
set interfaces fab0 fabric-options member-interfaces xe-2/0/2
set interfaces fab0 fabric-options member-interfaces xe-3/3/0
set interfaces fab1 fabric-options member-interfaces xe-14/0/0
set interfaces fab1 fabric-options member-interfaces xe-15/3/0
```

The *fab0* link is the *node0* side of the interfaces, while the *fab1* link is the *node1* side. To configure redundant *fab* links, additional interfaces are identified on the same *fab0* or *fab1* interface. The links must be of the same flavor for redundancy.

5. The final part of the boilerplate configuration is the group configurations for the two nodes of the cluster. For management and maintenance purposes, you have to differentiate between the two nodes. Since all the configuration is set in the active node, this poses a problem. This differentiation can be accomplished by the use of group configurations for each node. Each node recognizes its own group configuration, and all is well.

The items to be installed in the node group configuration are those that are different on each node and could include:

- Hostname
- *fxp0* interface address
- *fxp0* routing information
- Backup router information
- *snmp* node-specific information

The groups are named *node0* and *node1*, and the *apply groups* command is used to activate the configurations. The configuration for the groups is:

```
edit groups node0
set system host-name EDCFW-node0
set system backup-router 10.10.100.1
```

```

set system backup-router destination 0.0.0.0/0
set interfaces fxp0 unit 0 family inet address 10.10.100.2/24
set snmp description "Distribution-firewall-East-node0"
set snmp location "Bldg1A-rm123-rack43-0-43"
set snmp contact "Joh Dowed: 555-555-1234"
top
edit groups node1
set system host-name EDCFW-node1
set system backup-router 10.10.200.1
set system backup-router destination 0.0.0.0/0
set interfaces fxp0 unit 0 family inet address 10.10.200.2/24
set snmp description "Distribution-firewall-East-node1"
set snmp location "Bldg1A-rm123-rack44-0-43"
set snmp contact "Joh Dowed: 555-555-1234"
top
set apply-groups "${node}"

```

Once all these configurations are committed (after fixing any typos and bad addresses), the status of the cluster can be verified with the *cluster status* operational commands. There are a number of *chassis cluster* commands that show the status of the cluster. The simplest and most direct is:

```

{primary:node0}
lab@EDCFW-node0> show chassis cluster status
Cluster ID: 1
Node                Priority    Status    Preempt  Manual failover

Redundancy group: 0 , Failover count: 1
node0                255       primary   no       yes
node1                10       secondary no       yes

Redundancy group: 1 , Failover count: 1
node0                254       primary   no       no
node1                10       secondary no       no

```

A number of items are presented here that might get overlooked. The first is the prompt, the notification that this is the primary node. The status shows the priorities that were set for the redundancy groups and the failover status. Note that one manual failover has been performed, and the failover count indicates this. To see the reason for the failover and all sorts of other information, use this hidden command:

```

{primary:node0}
lab@EDCFW-node0> show chassis cluster information
node0:
-----
Redundancy mode:
  Configured mode: active-active
  Operational mode: active-active
Control link statistics:
  Control link 0:
    Heartbeat packets sent: 2788

```

```

        Heartbeat packets received: 2655
        Heartbeat packet errors: 0
        Duplicate heartbeat packets received: 0
    Control recovery packet count: 0
    Sequence number of last heartbeat packet sent: 2784
    Sequence number of last heartbeat packet received: 1714
Fabric link statistics:
    Probes sent: 2784
    Probes received: 1931
    Probe errors: 0
    Probes not processed: 1928
    Probes dropped due to control link down: 0
    Probes dropped due to fabric link down: 0
    Sequence number of last probe sent: 2784
    Sequence number of last probe received: 1714
Chassis cluster LED information:
    Current LED color: Green
    Last LED change reason: No failures
...

```

This display is reduced in content, but you get the picture; every piece of information concerning clustering and monitoring is shown with this command.

At this point, the system was up and operational, with management interfaces and the base configuration. The next portion of the configuration, the security stanza, did not change from the initial implementation. We decided to put all the “upper” interfaces in a single zone and each of the “lower” interfaces in a departmental zone. Each time the interfaces and the bridge groups changed, these had to be modified, but I’m presenting them once here for the initial interface configuration. For illustration purposes, I’m using four VLANs per interface rather than the multitude that the customer required:

```

edit security zones security-zone UPPER
set host-inbound-traffic system-services any-service
set host-inbound-traffic protocols all
set interfaces reth0.1
set interfaces reth0.2
set interfaces reth0.3
set interfaces reth0.4
set interfaces reth2.1
set interfaces reth2.2
set interfaces reth2.3
set interfaces reth2.4
top
edit security zones security-zone DEPT_A
set host-inbound-traffic system-services any-service
set host-inbound-traffic protocols all
set interfaces reth1.1
set interfaces reth3.1
set address-book address DEPT_A 10.10.10.0/24
top
edit security zones security-zone DEPT_B

```

```

set host-inbound-traffic system-services any-service
set host-inbound-traffic protocols all
set interfaces reth1.2
set interfaces reth3.2
set address-book address DEPT_B 10.10.20.0/24
top
edit security zones security-zone DEPT_C
set host-inbound-traffic system-services any-service
set host-inbound-traffic protocols all
set interfaces reth1.3
set interfaces reth3.3
set address-book address DEPT_C 10.10.30.0/24
top
edit security zones security-zone DEPT_D
set host-inbound-traffic system-services any-service
set host-inbound-traffic protocols all
set interfaces reth1.4
set interfaces reth3.4
set address-book address DEPT_D 10.10.40.0/24

```

The settings of the system services stanzas do not make a difference, because the routing engine should never be looking at inbound traffic. However, configuring them doesn't hurt, and it might help in troubleshooting down the deployment path.

The policies added were initially generic and were modified as requirements determined. Traffic from the UPPER zone was allowed to the server farm addresses, and traffic from the server farms was allowed to any addresses in the UPPER zone. The policies were:

```

edit security policies from-zone UPPER to-zone DEPT_A policy
  Permit_to_A
set match source-address any
set match destination-address DEPT_A
set match application any
set then permit
set then count
top
edit security policies from-zone UPPER to-zone DEPT_B policy
  Permit_to_B
set match source-address any
set match destination-address DEPT_B
set match application any
set then permit
set then count
top
edit security policies from-zone UPPER to-zone DEPT_C policy
  Permit_to_C
set match source-address any
set match destination-address DEPT_C
set match application any
set then permit
set then count

```

```

top
edit security policies from-zone UPPER to-zone DEPT_D policy
    Permit_to_D
set match source-address any
set match destination-address DEPT_D
set match application any
set then permit
set then count
top
edit security policies from-zone DEPT_A to-zone UPPER policy
    Permit_from_A
set match source-address DEPT_A
set match destination-address any
set match application any
set then permit
set then count
top
edit security policies from-zone DEPT_B to-zone UPPER policy
    Permit_from_B
set match source-address DEPT_B
set match destination-address any
set match application any
set then permit
set then count
top
edit security policies from-zone DEPT_C to-zone UPPER policy
    Permit_from_C
set match source-address DEPT_C
set match destination-address any
set match application any
set then permit
set then count
top
edit security policies from-zone DEPT_D to-zone UPPER policy
    Permit_from_D
set match source-address DEPT_D
set match destination-address any
set match application any
set then permit
set then count

```

You might note that there are no security policies for intrazone traffic. These would normally be necessary to allow traffic between two interfaces in the same zone. Considering the connectivity of this design, however, there is no need for traffic to ever be required to pass between the two interfaces of a zone. Intrazone traffic will always be switched at a lower or higher layer than the firewall.

With the security configuration opened up for the initial deployment, the interfaces are the only aspect that needs to be configured. This is the area of the configuration that changed over time. The first configuration used was with an active/passive redundant Ethernet configuration. The interface stanza was similar to:

```

edit interfaces reth0 unit 1
set description DEPT_A_East_up
set family bridge interface-mode trunk
set family bridge vlan-id-list 100
top
edit interfaces reth0 unit 2
set description DEPT_B_East_up
set family bridge interface-mode trunk
set family bridge vlan-id-list 101
top
edit interfaces reth0 unit 3
set description DEPT_C_East_up
set family bridge interface-mode trunk
set family bridge vlan-id-list 102
top
edit interfaces reth0 unit 4
set description DEPT_D_East_up
set family bridge interface-mode trunk
set family bridge vlan-id-list 103
top
edit interfaces reth1 unit 1
set description DEPT_A_East_dwn
set family bridge interface-mode trunk
set family bridge vlan-id-list 100
top
edit interfaces reth1 unit 2
set description DEPT_B_East_dwn
set family bridge interface-mode trunk
set family bridge vlan-id-list 101
top
edit interfaces reth1 unit 3
set description DEPT_C_East_dwn
set family bridge interface-mode trunk
set family bridge vlan-id-list 102
top
edit interfaces reth1 unit 4
set description DEPT_D_East_dwn
set family bridge interface-mode trunk
set family bridge vlan-id-list 103
top
edit interfaces reth2 unit 1
set description DEPT_A_West_up
set family bridge interface-mode trunk
set family bridge vlan-id-list 110
top
edit interfaces reth2 unit 2
set description DEPT_B_West_up
set family bridge interface-mode trunk
set family bridge vlan-id-list 111
top
edit interfaces reth2 unit 3
set description DEPT_C_West_up

```

```

set family bridge interface-mode trunk
set family bridge vlan-id-list 112
top
edit interfaces reth2 unit 4
set description DEPT_D_West_up
set family bridge interface-mode trunk
set family bridge vlan-id-list 113
top
edit interfaces reth3 unit 1
set description DEPT_A_West_dwn
set family bridge interface-mode trunk
set family bridge vlan-id-list 110
top
edit interfaces reth3 unit 2
set description DEPT_B_West_dwn
set family bridge interface-mode trunk
set family bridge vlan-id-list 111
top
edit interfaces reth3 unit 3
set description DEPT_C_West_dwn
set family bridge interface-mode trunk
set family bridge vlan-id-list 112
top
edit interfaces reth3 unit 4
set description DEPT_D_West_dwn
set family bridge interface-mode trunk
set family bridge vlan-id-list 113

```

This configuration installs the redundant Ethernet interfaces. Now the physical interface-to-Reth mapping must be added to the configuration.

The configuration makes all interfaces part of redundancy group 1 to assure that if one of the interfaces fails, they will all switch over to the other node. The chassis cluster configuration for redundancy group 1 is set to monitor all the interfaces with a weight of 255. That assures a switchover:

```

edit interfaces xe-2/1/0
set description "DCDUP00 2/1"
set gigether-options redundant-parent reth0
top
edit interfaces xe-14/1/0
set description "DCDUP00 2/2"
set gigether-options redundant-parent reth0
top
edit interfaces xe-2/2/0
set description "DCDUP01 3/1"
set gigether-options redundant-parent reth1
top
edit interfaces xe-14/2/0
set description "DCDUP01 3/2"
set gigether-options redundant-parent reth1
top

```



```

edit interfaces xe-3/1/0
set description "DCDDWN00 2/1"
set gigether-options redundant-parent reth2
top
edit interfaces xe-15/1/0
set description "DCDDWN00 2/2"
set gigether-options redundant-parent reth2
top
edit interfaces xe-3/2/0
set description "DCDDWN01 3/1"
set gigether-options redundant-parent reth3
top
edit interfaces xe-15/2/0
set description "DCDDWN01 3/2"
set gigether-options redundant-parent reth3

```

Once the interfaces are associated with the redundant Ethernet interfaces, the interfaces are installed in the bridge domains. One domain for each pair of interfaces (upper and lower):

```

edit bridge-domains bd100
set domain-type bridge
set vlan-id 100
set bridge-options interface reth0.1
set bridge-options interface reth1.1
top
edit bridge-domains bd101
set domain-type bridge
set vlan-id 101
set bridge-options interface reth0.2
set bridge-options interface reth1.2
top
edit bridge-domains bd102
set domain-type bridge
set vlan-id 102
set bridge-options interface reth0.2
set bridge-options interface reth1.2
top
edit bridge-domains bd103
set domain-type bridge
set vlan-id 103
set bridge-options interface reth2.2
set bridge-options interface reth3.2
top
edit bridge-domains bd110
set domain-type bridge
set vlan-id 110
set bridge-options interface reth2.1
set bridge-options interface reth3.1
top
edit bridge-domains bd111
set domain-type bridge

```

```

set vlan-id 111
set bridge-options interface reth2.2
set bridge-options interface reth3.2
top
edit bridge-domains bd112
set domain-type bridge
set vlan-id 112
set bridge-options interface reth2.2
set bridge-options interface reth3.2
top
edit bridge-domains bd113
set domain-type bridge
set vlan-id 113
set bridge-options interface reth2.2
set bridge-options interface reth3.2
top

```

While this configuration allowed full communications between the zones, it did not provide failover capabilities. The upper distribution router that was connected did not allow for an active/passive switchover with the same addressing on each port. The lower router was acting in a switch mode, so it had no problem with the Reths. To get around this issue, the Reth interfaces were eliminated from the configuration and straightforward 10 Gig Ethernet interfaces were used between the upper and lower distribution routers.

Take 2 Configuration: Active/Active without Reths

To provide active/active support without Reth interfaces, the XE interfaces were given the bridge family configuration and entered into the bridge groups. Rather than bore you with the full configuration, I'll just give the configuration for one interface and bridge group. You should be able to extrapolate the remaining interfaces and bridge groups:

```

delete interface xe-2/1/0
delete bridge-domains bd100
edit interfaces xe-2/1/0
set description "Connected to DCCUP00 port 1/2"
set vlan-tagging
set unit 1 description DEPT_A_UP
set unit 1 family bridge interface-mode trunk
set unit 1 family bridge vlan-id-list 100
set unit 1 description DEPT_B_UP
set unit 2 family bridge interface-mode trunk
set unit 2 family bridge vlan-id-list 101
set unit 1 description DEPT_C_UP
set unit 3 family bridge interface-mode trunk
set unit 3 family bridge vlan-id-list 102
set unit 1 description DEPT_D_UP
set unit 4 family bridge interface-mode trunk
set unit 4 family bridge vlan-id-list 103

```

```
top
edit bridge-domains bd100
set domain-type bridge
set vlan-id 100
set bridge-options interface xe-2/1/0.1
set bridge-options interface xe-3/1/0.1
top
```

To deal with the loss of the redundant interfaces, we had to double the number of physical interfaces, VLANs, and bridge domains. The security zones stayed the same, but each had four interfaces rather than the two shown in the security configuration.

Again, while this configuration worked, the traffic could not be split between the two nodes. All traffic flowed via a single node in the cluster. If an upstream failure caused the routing to be changed from the East side to the West side, all routes had to fail over. This was not an acceptable solution.

The issue was that the Junos 10.x code did not support true active/active operation for transparent mode. During testing, a prereleased version of the Junos 11.x code was used to provide this feature, and when this code was final, the configuration was generated.

Take 3 Configuration: Active/Active with One-Legged Reths

The final configuration was the result of a concerted effort between the members of the warrior tribe, JTAC, and Juniper Networks developmental engineering (DE). The concept of active/active operation was redefined during this period. In the past, active/active meant that multiple redundant Ethernet interfaces were configured on the cluster, and some had the active interface on *node0* and others on *node1*. By manipulating the redundancy groups, the active/standby status could be altered. If no Reths were present, the device was running in active/passive mode. But according to DE, in 11.x a device is considered to be running in active/active mode whenever an interface is active on *node1* of the cluster. So yes, the 10.x code was not going to perform as expected. The second piece of information from DE was that in order for an interface to operate properly from *node1*, it should be part of a Reth. Considering that Reths did not work in this environment, we came up with the concept of the *one-legged Reth*, setting a Reth interface per the normal configuration, but only assigning a single physical interface to the Reth.

To meet the connectivity demands of the design, the number of Reths was increased from the original four to eight. Each still carried multiple VLANs, and each was paired with another interface for the bridge group. We tried to tie four interfaces together in a bridge group, but the random nature of the bridge group caused the load balancers to start discarding traffic.

We ended up with a total of eight redundant Ethernet interfaces, each carrying 10 VLANs, creating a total of 40 VLANs and bridge groups. The configuration was a monster but looked very similar to what has been shown previously, so I will not repeat it again. Instead, the mapping between the physical interfaces, logical interfaces, VLANs,

and Reths is shown in [Table 3-1](#). There is an odd VLAN that was added to each interface for management purposes. These carry overhead traffic between the devices, but still need security configurations. All of these interfaces were initially installed in a single security zone with free access between all elements.

Table 3-1. Interface mapping

Physical interface	Redundant Ethernet interface	Logical interface	VLAN	Bridge domains
xe-2/1/0	Reth0	unit 1—unit 10	101—110, 10	bd100—bd109, bd10
xe-2/2/0	Reth1	unit 1—unit 10	111—119, 11	bd110—bd119, bd11
xe-3/1/0	Reth2	unit 1—unit 10	100—109, 10	bd100—bd109, bd10
xe-3/2/0	Reth3	unit 1—unit 10	111—119, 11	bd110—bd119, bd11
xe-14/1/0	Reth4	unit 1—unit 10	120—129, 12	bd120—bd129, bd12
xe-14/2/0	Reth5	unit 1—unit 10	130—139, 13	bd130—bd139, bd13
xe-15/1/0	Reth6	unit 1—unit 10	120—129, 12	bd120—bd129, bd12
xe-15/2/0	Reth7	unit 1—unit 10	130—139, 13	bd130—bd139, bd13

This configuration met all the requirements of redundancy, connectivity, diversity, and routing. All routing decisions are made at the routers, and the SRX is only a direct path between two routers.

Testing

During all of the configuration changeups, there was constant testing of the throughput and the latency through the firewall. The benchmark for testing was to mimic an Ethernet fiber cable connecting the two routers. We had to inform the other engineers that the fiber was not reassembling the packets and performing policy lookups twice for each test run. This did not make a difference to the “other” team; transparent means transparent, and it was not expected to affect the traffic in any manner.

A couple of issues with lost traffic under high loads (we were showing a .02% traffic loss at 90% utilization) were resolved by firmware changes to the IOCs.

The testing then moved from latency and throughput tests to security testing. This is where things got really interesting. If you remember from the initial discussion, the default gateway for the servers (and the load balancers, for that matter) is the upper distribution routers. The lower routers are actually Level 2 switches that pass the traffic up the stack. When traffic is from a user to a server, the normal operation of the firewall is seen: traffic originates at the upper tiers of the network and terminates in the lower portion. Each set of interfaces (bridge domain) was installed in a different zone, and policies were crated that allowed traffic between each zone.

When traffic passed from one server to another, the traffic passed up the stack to the upper distribution routers, then back down. Each pass was a new session for the firewall.

We started seeing strange connections and could not follow the traffic between the flows. What we determined was that the virtual servers were not using a unique MAC address, so the traffic for different flows was seen as coming from the same source. This made the Ethernet tables in the SRX very confused, and was the culprit responsible for our troubles. Once a unique MAC address was assigned to each virtual adapter on the servers, all things ran as expected. The MAC tables showed:

```
{primary:node0}
peter@EDCFW-node0> show bridge mac-table

MAC flags (S -static MAC, D -dynamic MAC,
           SE -Statistics enabled, NM -Non configured MAC)

Routing instance : default-switch
Bridging domain : bd1, VLAN : 10
  MAC          MAC          Logical
  address      flags        interface
  00:45:98:18:18:42  D,SE    reth5.1
  00:26:97:18:18:43  D,SE    reth4.1

MAC flags (S -static MAC, D -dynamic MAC,
           SE -Statistics enabled, NM -Non configured MAC)

Routing instance : default-switch
Bridging domain : bd11, VLAN : 100
  MAC          MAC          Logical
  address      flags        interface
  00:45:98:18:09:c3  D,SE    reth0.1
  68:ef:bd:a5:b7:30  D,SE    reth1.1
...
```

During the next phase of the testing—the failure testing—it became apparent that for all the planning and mapping, asymmetrical flows could exist in the network. That is, traffic could originate on the DCDDWN00 switch and terminate on the DCDDWN01 switches. Oops!

The resulting little piece of engineering required a rethinking of the zones and policies. 00 and 01 traffic was considered distinct, and we should not see both at the same time. What was happening at this point was that all the logical interfaces for the same department were going in the same zone. We redesignated the zones and created a new set of policies to include intrazone policies.

Once these changes were performed, all the further tests passed without a hitch. Our little “bump in the wire” ran just as expected.

Summary

This engagement was a challenge for this warrior tribe—more than once we considered folding up our tents and calling it a day. The politics and lack of a sense of fair play were

issues that were relatively new for us. Add to this a totally new deployment configuration, and you have the makings of a number of head-scratching sessions and late hours of conferences. Being a professional services engineer is often a challenge, and this engagement showed what we were made of. As the face of Juniper Networks, we took the high ground, presented the capabilities and faults of the SRX truthfully, and in the end, proved that this device and our tribe were able to meet the demands of a totally hostile environment. That says a lot for the equipment, the company's technology, and the tribe. I'm proud to be a member.

Layer 3 to Layer 2 Conversion

Service providers have deployed and enterprises have used IP networks for a couple of decades now, routing IP packets over a core network. The core network in these deployments consists of a series of IP routers. Recently, the trend has been to deploy a Layer 2 technology (Ethernet) in place of, or to augment, the Layer 3 networks. In this engagement, our tribe of network warriors assisted an enterprise (wishing to become a limited service provider) in deploying a Layer 2 virtual private network (L2VPN) service in one of the most inhospitable environments possible: Alaska.

This network warrior is a Vermonter, and we consider our state to be rugged and, for the most part, sparsely populated. I can now say we are mistaken. Alaska is wilderness; it has a similar population to Vermont (~730,000 versus ~630,000) but almost 70 times the land mass (663,000 sq. mi. versus 9,600 sq. mi.). The old Vermont saying, “Ya can’t get thar from here,” applies in Alaska as well, because for the most part there are no roads to get where you want to go. I heard an ad specify that one in five of the Alaskan population holds a pilot’s license, and now I understand why. A traveler in Alaska uses every available means to get from point A to point B: cars, planes, trains, quad bikes, 4x4s, and yes, sleds (although powered by gas, not dog food).

Deploying a network in such an environment likewise requires the use of every available transmission technology: fiber, terrestrial microwave, wire (DS1s and DS3s), and even satellite links. Links are rented, owned, and shared by service providers, universities, the state, and other companies. Combining these links into an IP network is not a gigantic challenge: IP is self-policing as far as link changes are concerned. Fragmentation happens at the low-speed links and reassembly happens at the end points. IP packets flow between the routers on a per-hop basis, stitching a path between end stations one router at a time. The paths are dynamic in nature and are determined by routing protocols and the “costs” of the links between them. The enterprise went into this project with a high confidence level because the IP network it had stitched together in the past had performed to expectations.

When an L2VPN is deployed, the network becomes an area-wide Ethernet bridge. Sites are interconnected by an Ethernet broadcast domain, not by IP hops. Routers are not needed in this environment—their only function is to interconnect the different broadcast domains that might be deployed. This may sound to the typical IT person like a simpler, easier, more efficient solution to interconnectivity than all the bother of IP networks. There are two different L2VPN technologies that are being deployed today. The first is a point-to-point technology that provides Ethernet links between sites. This virtual private wire service (VPWS) creates pseudowires between sites to transport Layer 2 frames. Switches or routers are used to terminate the pseudowires and determine where traffic can and should flow. VPWS connectivity is depicted in [Figure 4-1](#), where each user site is connected to the L2VPN service by an Ethernet switch.

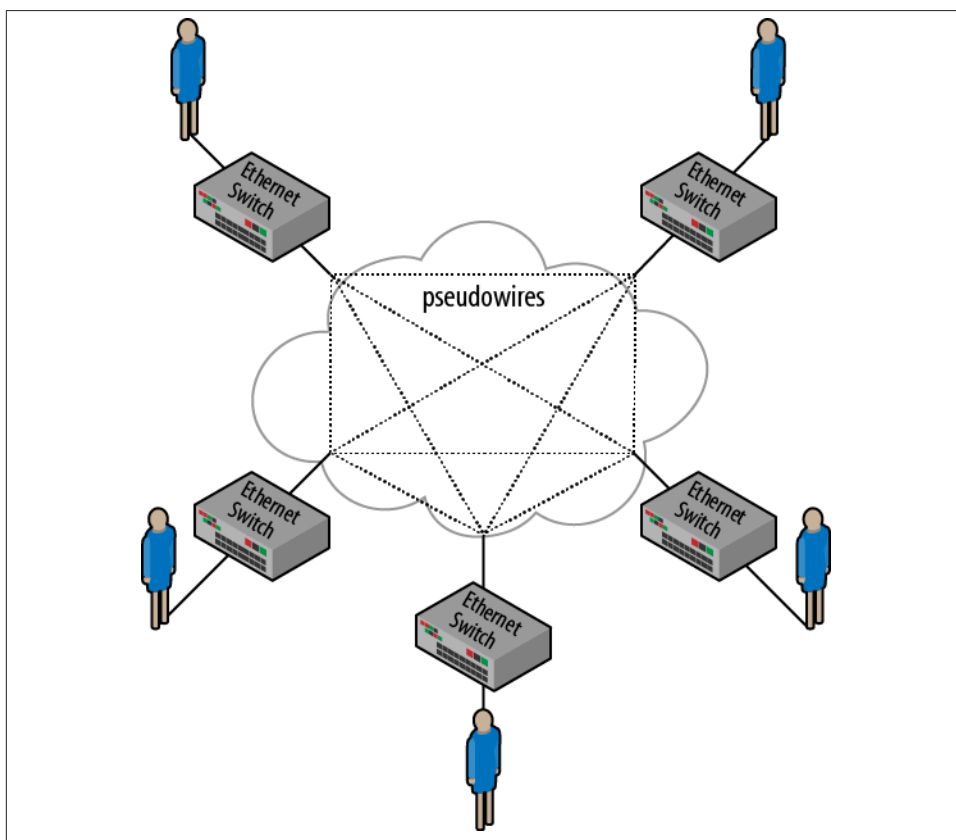


Figure 4-1. VPWS connectivity

The second technology is called *virtual private LAN service* (VPLS). While it's based on a similar technology to VPWS, VPLS offers any-to-any connectivity between sites in an

enterprise. The core network looks like a gigantic switch rather than a series of point-to-point links. For this engagement, the VPLS technology was chosen. Each site was to be interconnected to the cloud, and the cloud provided Ethernet switching (MAC learning bridging, using the old-school terminology), as shown in [Figure 4-2](#).

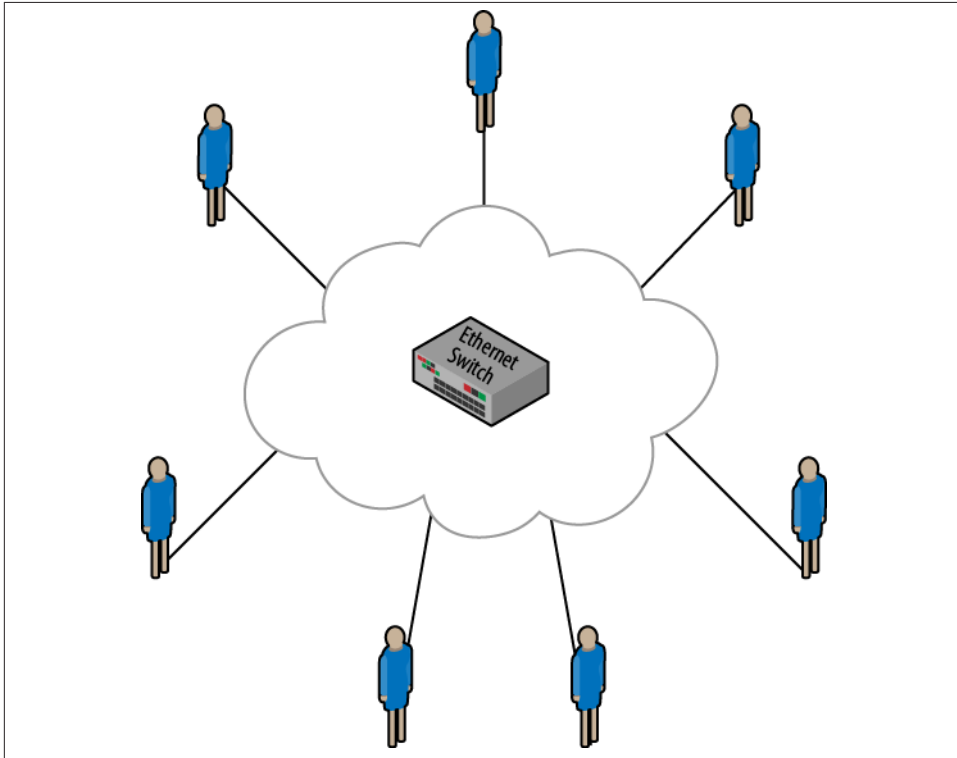


Figure 4-2. VPLS connectivity

L2VPN technologies are not new; they have been deployed in various environments and for different applications for a number of years. The issue with L2VPNs in general, and specifically VPLS L2VPN implementations, is that there is *no concept of fragmentation in the network*. In a classical IP network, routers do the fragmentation. In a classical bridged network, the theory is that this is a local area network and maximum transmission unit (MTU) changes should not be seen. However, when you add interconnecting links to the bridge network, you might have a problem. If the MTU is the same on all the links, all is well, but when there are links with differing MTUs, this is where network warriors start to earn their pay.

As traffic enters the VPLS network, the ingress device (PC or router) can fragment the traffic to the specified MTU of the local link. If the frame then encounters a link that

cannot handle that MTU, the frame is dropped (ouch), and no ICMP message is returned to indicate that an issue was experienced (ouch again). In most VPLS networks, there are no means for fragmenting an Ethernet frame and no routers to fragment the IP packets—remember, the network looks like a big Ethernet switch.

Problem

A bunch of Juniper Networks warriors were brought in to deploy an L2VPN based on VPLS technology in a network with multiple remote locations reached by various transmission means. The warrior tribe for this assignment was a trimmed-down group of professional service engineers, like me, and the client's networking engineers. Added to this mix were radio and satellite engineers who were called in for certain skirmishes. The client owns, rents, borrows, and shares transmission media from various providers. There are local exchange carrier links (T1s and DSL), long-distance provider links (T1s and DS3s), fiber channels (OC3 and OC12s), and radio links from any number of sources, including terrestrial microwave and satellite microwave (Ethernet and T1). Due to this arrangement, the client did not have direct control over the transmission system. However, they did own the routed network, which consisted of a mixture of Juniper Networks M-series and MX-series routers and EX-series switches. Because of the cost of transmission facilities and the geographic realities of Alaska, the network was a giant star, as shown in [Figure 4-3](#). Many of the remote locations had names that us folks in the lower 48 find hard to pronounce (Emmonak, Kotlik, or Unalakleet), yet some were familiar from a historical or romantic perspective (Nome, Fairbanks, and Barrow).

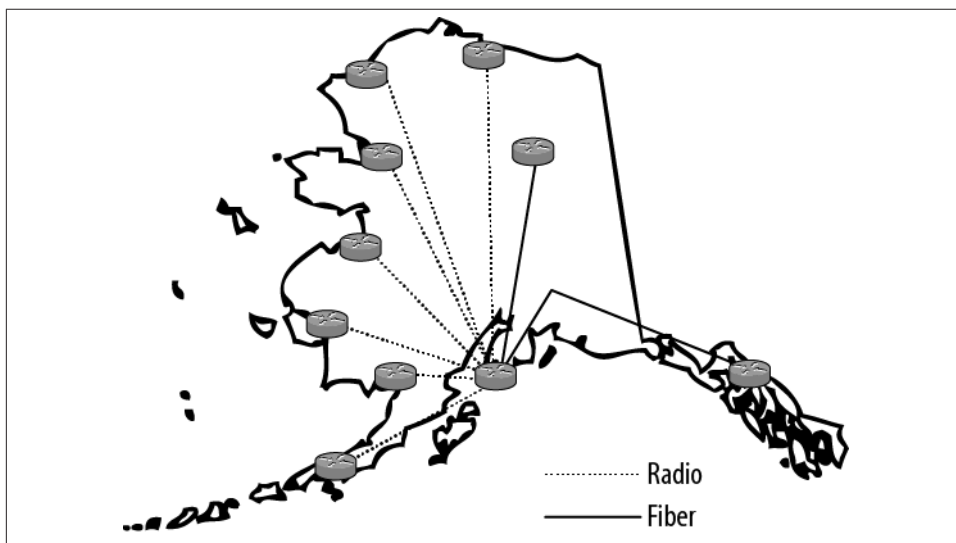


Figure 4-3. Network topology

The problem that we were facing was that the transmission systems had different maximum frame sizes, and we had no way of adjusting the frames automatically. An experienced reader would pipe in at this point and say, “Hey Peter, it’s all Ethernet, the standard has been around forever and everybody supports it.” And you would be right, except that we were transmitting not only Ethernet, but Ethernet encapsulated in multiprotocol label switching (MPLS).

The problem definition has to start with some networking basics.



Sorry for the Networking 101 stuff, folks, but this is where a warrior has to go when the head scratching begins and it’s time to start counting bytes.

When a ping is generated from a PC connected to an Ethernet segment, [Figure 4-4](#) is what’s seen at the command prompt.

```
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\Peter>ping 4.2.2.2

Pinging 4.2.2.2 with 32 bytes of data:
Reply from 4.2.2.2: bytes=32 time=60ms TTL=53
Reply from 4.2.2.2: bytes=32 time=56ms TTL=53
Reply from 4.2.2.2: bytes=32 time=59ms TTL=53
Reply from 4.2.2.2: bytes=32 time=55ms TTL=53

Ping statistics for 4.2.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 55ms, Maximum = 60ms, Average = 57ms

C:\Users\Peter>
```

Figure 4-4. PC-based ping

The basic ping is sending 32 bytes of data (abcdefg...) and receiving the same from the destination (in this case, 4.2.2.2). The basic frame format for a ping on an Ethernet is:

1. An ICMP header (8 bytes) is added to the data to form a message.
2. An IP header (20 bytes) is added to the ICMP header to form a packet.
3. The IP packet is encapsulated in an Ethernet frame that has a header (14 bytes) and a trailer (4 bytes).
4. Other Ethernet stuff (preamble bits) is added to the frame and the frame is transmitted on the wire.

So, our 32 bytes of data are surrounded by (8 + 20 + 14 + 4 = 46) 46 bytes of protocol overhead, plus the preamble bits. If you discount the preamble, the frame that is transmitted on the wire is 78 bytes long and looks like [Figure 4-5](#).

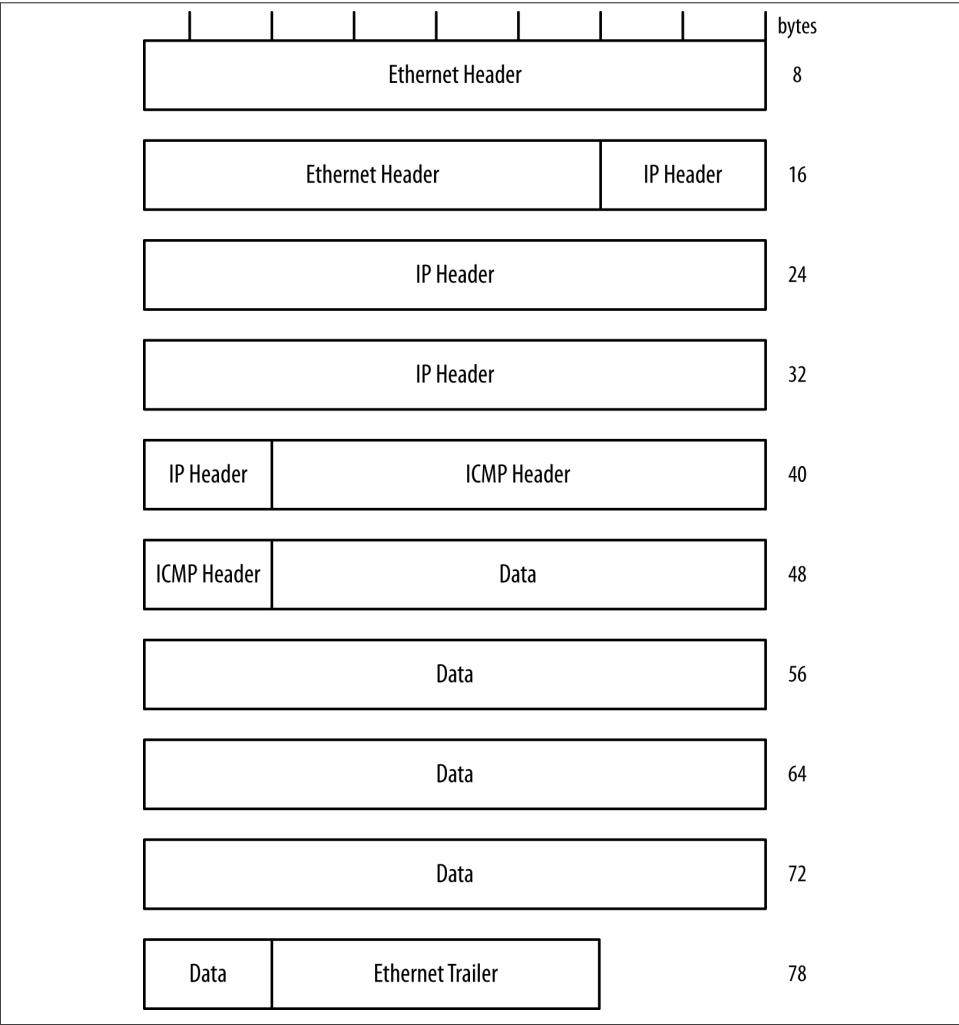


Figure 4-5. Ping packet structure

If the network were a series of interconnected links pushing Ethernet frames, we'd be able to end the Networking 101 review at this point. But sadly, that is not the case—we have two more pieces to add to this networking warrior puzzle.

Q-in-Q Framing

Because the client had many different departments that needed connectivity, and they also thought that once this beast was up and running they might be able to sell service to other companies in the area, they decided to use virtual LAN (VLAN) tagging to differentiate the departments and to differentiate between companies. This double tagging for Ethernet frames is called *Q-in-Q tagging* and is defined in IEEE standard 802.1Q. The dual VLAN tags (4 octets each) are installed in the Ethernet header and expand the header from 14 octets to 22 octets. The Q-in-Q-enhanced Ethernet header now looks like [Figure 4-6](#).

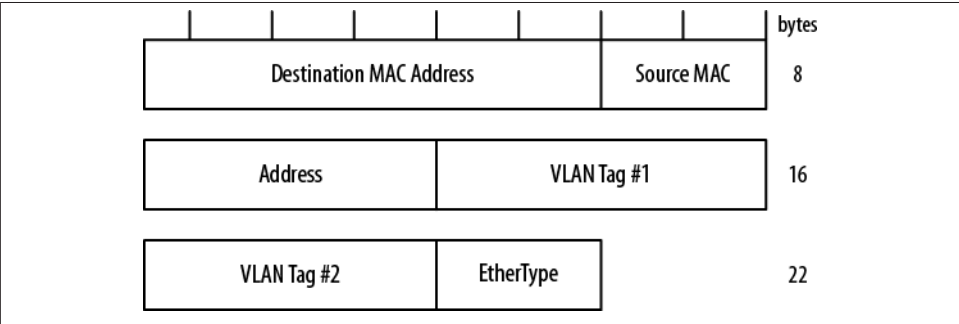


Figure 4-6. Q-in-Q header

The first and second tags each contain a type field that says that a tag is being carried. The final EtherType field defines the actual payload that is being carried (an IP packet). The addition of the Q-in-Q headers increases the ping packet size from 78 octets to 86 octets. This increase happens not at the PC, but at the first switch that is encountered. This is an important detail because of where fragmentation is occurring in the system. The PC is fragmenting the initial message to fit into packets that can go on the wire using a default MTU (1,518 bytes) for that fragmentation decision. If a switch then adds information to that frame, increasing the total size by 8 bytes, the 1,518-byte maximum-sized frame is now 1,526 bytes.

VPLS Overhead

The second piece of this puzzle is that the Ethernet frames (really Q-in-Q frames) are not being placed on the wire for transport to the destination, but are being carried over an MPLS infrastructure. As stated previously, the client deployed a VPLS infrastructure to transport the Ethernet traffic between locations. The use of VPLS again adds to the overhead of the information being carried.

When we consider VPLS, the overhead calculation for our initial ping starts as it did initially:

1. An ICMP header (8 bytes) is added to the data to form a message.
2. An IP header (20 bytes) is added to the ICMP header to form a packet.
3. The IP packet is encapsulated in an Ethernet frame that has a header (14 bytes) and a trailer (4 bytes).

The Ethernet frame then encounters the first VPLS provider edge router. At this point:

1. The Ethernet frame is stripped off the trailer (−4 bytes).
2. A VPLS (customer) label is added to the Ethernet frame (4 bytes).
3. An MPLS (transport) label is added to the VPLS label (4 bytes).
4. The MPLS frame is encapsulated in the transport medium framing, which in this case is Ethernet (14 bytes header and 4 bytes trailer).

For those who have not battled with VPLS or MPLS before, the concepts of labels and label stacking might have just raised some serious flags (like, “Whoa there, where did that crap come from?” types of flags). MPLS is a switching technology very similar to frame relay (FR) and asynchronous transfer mode (ATM) from the old schools. The payload is given a header that identifies a path through the network. The header is the label. The difference between FR/ATM and MPLS is that MPLS exists on the same IP infrastructure as normal IP traffic. The payload and the MPLS headers are sent between routers over the same links as normal IP traffic. In our example, the Ethernet header identifies the payload as MPLS rather than IP—it’s a pretty slick technology.

An MPLS network supports multiple service types (Layer 3 VPNs and L2VPNs being two of them). Another header (label) is added to differentiate between these services and the customers on these services. The labels exist above the Ethernet header and below the payload headers in the protocol stack. The OSI purists call MPLS a Layer 2.5 protocol or a *shim protocol* because it does not line up neatly in the OSI stack of definitions.



From a warrior’s point of view any weapon that gets the job done is just as good as any other, so it’s a great thing that Juniper Networks is known to stick to the standards, unlike other vendors who tend to make their own standards. With Juniper Networks gear, interoperability and stability between (most) vendors is a given.

In our implementation, VPLS uses two labels for passing traffic over the network: the transport label identifies the path that the traffic is to use between the edge devices, and

the customer label identifies the traffic as VPLS and belonging to a specific customer. For us, the mapping of VLAN tags and VPLS labels provides that one-to-one correspondence between customers and VPLS labels. In a default Juniper Networks MPLS network, the transport label is “popped” (extracted) by the second-to-last router in the MPLS path. This process, called *penultimate-hop popping*, is used to reduce the processing on the last (ultimate) router.

Figure 4-7 follows a frame in this environment that arrives at the edge router (called a *provider edge* or PE router). Two labels, transport and customer, are added (pushed) onto the packet. The transport label will be swapped at each internal router (called a *provider* or P router) until the frame hits the second-to-last (penultimate) router. At the penultimate router, the transport label is popped and the frame with the customer label is delivered to the destination PE router. The destination PE pops the customer label, maps the traffic to the correct outgoing interface, and sends the frame on its way. The label identifiers that are commonly found in Juniper Networks routers are between 250000 and 300000.

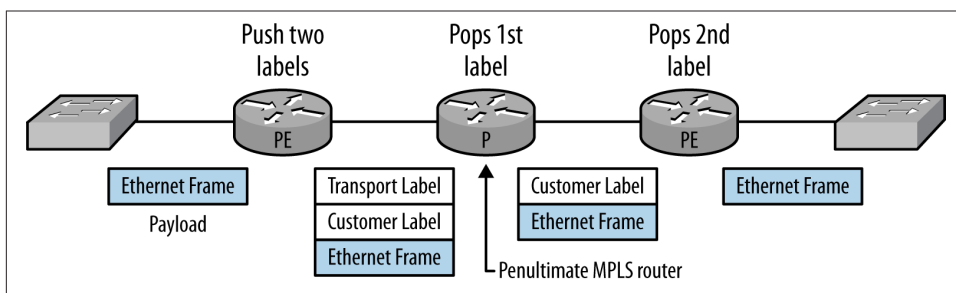


Figure 4-7. VPLS label processing

There is one additional note to be added here: the MPLS labels cannot transport traffic between routers without a true data link and physical layer protocol (this is a difference from FR and ATM networks). Our MPLS label, etc. are wrapped in whatever data link protocol is running between the routers. In our example, this happens to be Ethernet.

Given this last piece of information, our 32-byte ping data now has a whopping 72-byte header attached:

- 8 – ICMP header
- +20 – IP header
- +4 – 802.1Q tag#1
- +4 – 802.1Q tag#2
- +14 – Ethernet header (original)
- 4 – Ethernet trailer (original, stripped)
- +4 – MPLS label, customer

+4 – MPLS label, transport
+14 – Ethernet header (new)
+4 – Ethernet trailer (new)
= 72 total overhead bytes

The additional overhead makes the packet on the wire 104 bytes long!

During testing, the warriors captured traffic that showed all these levels of overhead. The ping was a 1,000-byte ping from a Juniper Networks router, not the 32-byte ping that we have been using for the preceding analysis, but the overhead showed the full protocol stack:

```
Ethernet II,
  Destination: JuniperN_db:ba:00 (00:24:dc:db:ba:00)
  Source: JuniperN_03:be:01 (00:1d:b5:03:be:01)
  Type: MPLS label switched packet (0x8847)
MultiProtocol Label Switching Header,
  Label: 262152, Exp: 0, S: 1, TTL: 255
MultiProtocol Label Switching Header,
  Label: 301152, Exp: 0, S: 0, TTL: 255
Ethernet II,
  Destination: JuniperN_db:9b:80 (00:24:dc:db:9b:80)
  Source: JuniperN_db:9b:81 (00:24:dc:db:9b:81)
  Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN,
  PRI: 0, CFI: 0, ID: 488
  Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN,
  PRI: 0, CFI: 0, ID: 516
  Type: IP (0x0800)
Internet Protocol,
  Src: 1.1.2.1 (1.1.2.1),
  Dst: 1.1.2.2 (1.1.2.2)
  Version: 4 Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 1028
  Identification: 0xbad3 (47827)
  Flags: 0x00
  Fragment offset: 0
  Time to live: 64
  Protocol: ICMP (0x01)
Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0 ()
  Checksum: 0xbdd8 [correct]
  Identifier: 0xef03
  Sequence number: 9488 (0x2510)
  Data (1000 bytes)
```



In today's environment of Ethernet switches, it is all but impossible to find a shared-bandwidth hub. These old devices allowed a protocol sniffer (packet analyzer) to monitor traffic between any two devices. Each port saw all the traffic on the hub. To gather the same traffic for a sniffer, Ethernet switches have to be configured to allow a monitor port. For Juniper Networks EX switches, the configuration for this function is called *port mirroring*. The configuration is:

```
[edit ethernet-switching-options]
  analyzer vpls {
    input {
      ingress {
        interface ge-0/0/23.0;
        interface ge-0/0/21.0;
      }
    }
    output {
      interface {
        ge-0/0/22.0;
      }
    }
  }
```

In this configuration, the ports *ge-0/0/23* and *ge-0/0/21* are the traffic-carrying ports. The traffic is mirrored to port *ge-0/0/22*, where the sniffer is attached. There are all sorts of restrictions on port mirroring on the devices, so check out the [Juniper techpubs](#) for all the details.

All of this warrior protocol analysis stuff showed that there was a lot of overhead on our network. While this is not an issue in itself, there are many more complex protocol stacks in use in the industry today. In this case, the overhead added by the network after the packet was put on the wire caused our initial 78 bytes of traffic from the PC to grow to 102 bytes. This 24-octet growth could take a full-size Ethernet frame (1,518 bytes) and make it bigger than the default MTU of a link. If the device terminating that link is unable to fragment the packet, it will discard this traffic.

This is the crux of our problem. The devices in the network do not fragment the traffic once the PC has transmitted it. All of the devices on the network are supposedly working at the Ethernet layer and do not support fragmentation. If a network link supports less than the required ($1,518 + 24 = 1,542$) MTU, then VPLS over that link will fail.

RFC 4448 defines the encapsulation of Ethernet frames in an MPLS/VPLS network. The applicable paragraph from the RFC states the issue:

6. PSN MTU Requirements

The MPLS PSN MUST be configured with an MTU that is large enough to transport a maximum-sized Ethernet frame that has been encapsulated with a control word, a pseudowire demultiplexer, and a tunnel encapsulation. With MPLS used as the tunneling

protocol, for example, this is likely to be 8 or more bytes greater than the largest frame size. The methodology described in [FRAG] MAY be used to fragment encapsulated frames that exceed the PSN MTU. However, if [FRAG] is not used and if the ingress router determines that an encapsulated Layer 2 PDU exceeds the MTU of the PSN tunnel through which it must be sent, the PDU MUST be dropped.

The failures are interesting to note. A normal ping will work, and an extended ping will work up to a limit (1,472 seems to be the limit), while larger pings will simply time out. An FTP over the link will work most of the time but will have a very slow response time. HTTP pages do not load. What this means to the casual observer is that the link is good, VPLS connections are good, MAC addresses are being learned, and pings are OK over the link. Only when traffic is actually tried between the edge devices is the link determined to be defective (often by the customer).

Solutions

The first obvious solution to this problem was that all links must support MTUs greater than 1,542. If the client was in an East Coast city or a Midwest area, the MTU would be a design specification on the order form. A setting would be made on the fiber multiplexer, and all would be happy. But considering that we were not in any of those areas, and the transmission facilities were in some cases 20-year-old technologies, the option to “just change the MTU” was not possible. Other alternatives had to be explored. The warriors had to put their heads together.

RFC 4623

RFC 4623, Pseudowire Emulation Edge-to-Edge (PWE3) Fragmentation and Reassembly, defines a mechanism for fragmenting traffic on a VPLS network. The RFC is interesting in that it states that other means should be used instead of fragmentation by the network devices to solve this issue. The list includes:

1. Proper configuration and management of MTU sizes between the customer edge (CE) router and provider edge (PE) router and across the VPLS/MPLS network.
2. Adaptive measures that operate with the originating host to reduce the packet sizes at the source. Path MTU and TCP MSS are two measures that are in common use.
3. The ability to recognize an oversized packet and fragment it at the PE router. The PE may be able to fragment an IP version 4 (IPv4) packet before it enters a VPLS/MPLS path.

Number one is the obvious answer that we already discussed. Number two is interesting in that it does work, but only for very specific applications: some standard web browsers (IE and Firefox) do not use these methods, so this option is not useful to us.

Number three is interesting, and we explored the possibility of using this as a solution. This issue here is that the solution is not scalable. We had to terminate each of the VLANs, create a routed interface for them, and adjust the default gateway of the devices on the customer's LANs. These restrictions obviated the benefits of VPLS.

We looked into the fragmentation capabilities defined in the RFC, approached Juniper Networks for their take on the options, and were told that the Junos OS did not yet support VPLS/MPLS fragmentation (it may be by the time you are reading this).

After a few more warrior sessions, two solutions to the problem were suggested.

Customer MTU restrictions

The first solution was setting the MTU at the customer locations to a value less than 1,518 bytes. If the sending PC or the last router in the network set its MTU to a lower value, these devices would fragment the larger packets and solve our problem.

By reducing the MTU at the customer edge from 1,518 bytes (the default) to 1,494 bytes, we could ensure that the maximum frame size with all the additional overhead would remain below the 1,518-byte default that the transmission links can handle.

This solution worked for our client for their internal divisions, because they owned the infrastructure and managed the PCs and routers that connect to this network.

However, the solution did not work for the client's long-range plans. They wanted to be able to recoup some of the cost of the network by selling interconnectivity services to other local companies. In these cases, they would not have control over the PCs and internal devices. An option for the proposed customers was to offer a "managed" router to interconnect with the customer and to set the MTU on the router's outgoing interface. The solution, while not elegant, gave the client a means to move forward on the project.

Move the MTU

The other solution was the deployment strategy for the MPLS VPLS service. The use of Q-in-Q VLAN tagging provides a separation between the client's traffic from different departments and possible third-party traffic. The links that have the lower MTUs are the spoke links to the remote locations. Rather than deploying a router at the end of these spoke links and pushing MPLS to the remote locations, we thought, why not leave the links as Ethernet, deploy a switch to the remote locations, and install MPLS/VPLS at the core locations?

The solution is shown in [Figure 4-8](#), and it moves the MPLS labels to the core of the network rather than the spokes of the network. The use of Q-in-Q on the links drops the requirement for the MTU from 1,542 to 1,526 bytes, an MTU that we found most vendors could support.

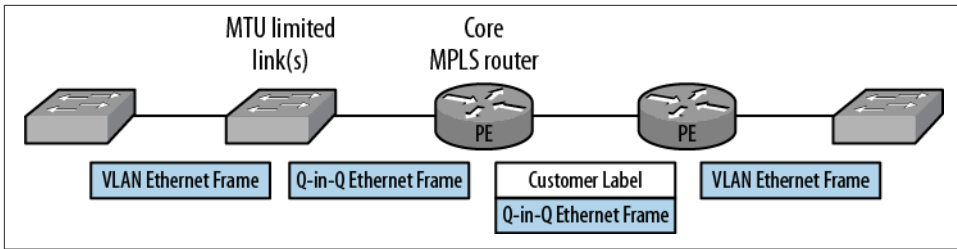


Figure 4-8. VPLS redesign

Our initial deployment used a combination of solutions. Where the client had routers that connected the traffic to the VPLS/MPLS network, the router's outgoing interface MTU was adjusted to be 1,494 bytes. Where the client only had Ethernet equipment at the remote location, the redesign was used to limit the overhead at these locations. Finally, when the service is resold, a managed router will be used at the customer-facing interface to provide fragmentation.

At some point Juniper Networks will include fragmentation in the VPLS code, but this customer did not have the time to wait for that eventuality.

Configurations

Once the design issues were resolved and the design could be rolled out, the actual configuration of the devices took place. The client allowed the devices to be staged prior to deployment into the nether regions of Alaska (the warriors were spared the pleasure of doing battle with the native warriors of that great state, although from what I hear, they would have kicked our butts all the way back to the lower 48).



While not trying to make this sound like a travelogue, while in Alaska, I was introduced to many of the different native peoples of the region. The native lore includes tales of many impressive warriors. We hear many stories of the Wild West and those Native Americans, but very few of the Wilder Far Northwest and these fierce Native Americans.

We initially deployed 20 devices in the staging area:

- The core routers that were to be our primary Internet access devices and the router reflectors for the rest of the network
- The primary distribution routers that served the local users and supported the firewalls to the Internet
- The remote distribution routers where transmission facilities could handle the MTU of a full VPLS frame

- The remote distribution switches where the transmission facilities were less than ideal
- The remote managed switches for the customers

The network as staged looked like **Figure 4-9**.

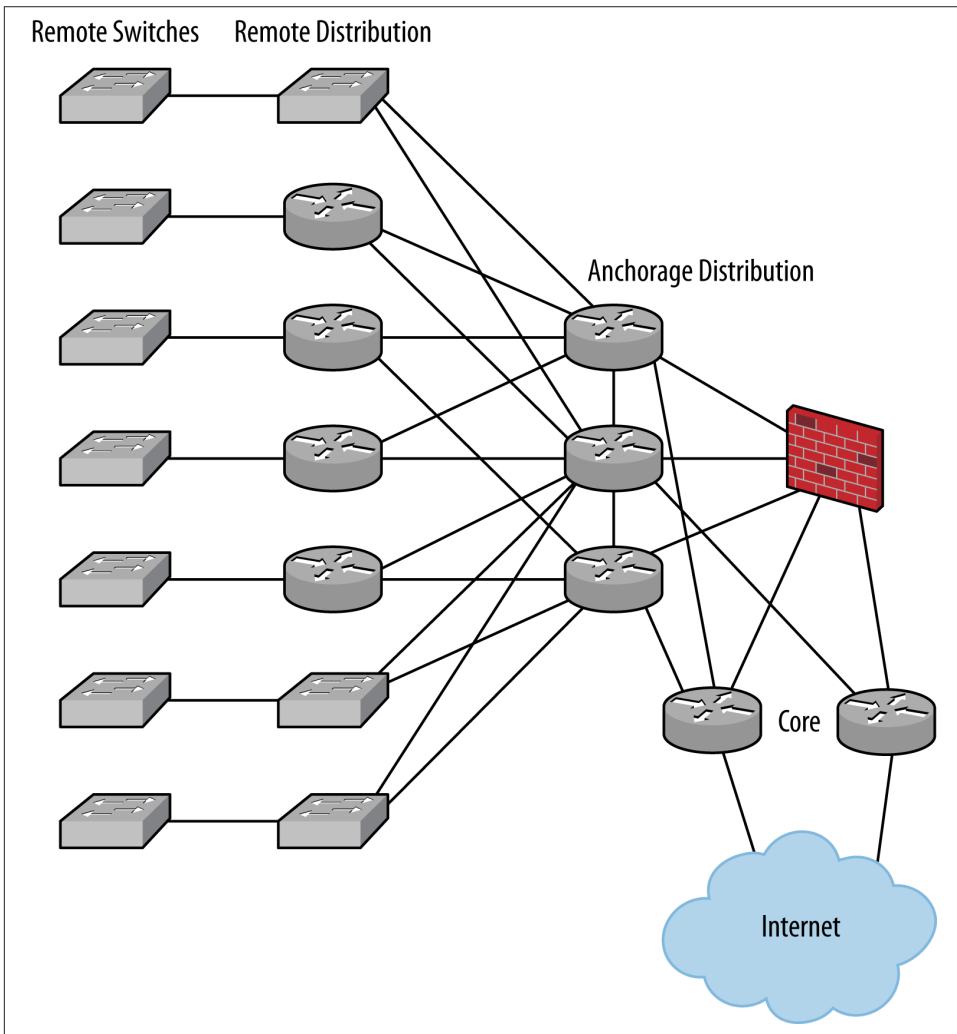


Figure 4-9. Staged network

Presented here are the configurations from one of each type of the devices. The configurations cover:

The management aspects of the network

These were all in-band as there were no facilities for out-of-band management (except for dial-up modems).

Router security

These configurations protected the routers from hackers and local threats.

BGP connectivity

A dual route reflector topology was used for BGP and also offered Internet access.

VPLS overlay

The VPLS overlay includes the label distribution protocol (LDP) and MPLS protocols where possible and Ethernet Q-in-Q where necessary.



The addresses used here have been sanitized for all the configurations. This client had a rather large legacy IP address space for its internal network. The addressing used that public address space for all aspects of the network. In my representation of the configurations, the address space has been replaced by the ubiquitous 10.0.0.0. I am aware that address translation would have to take place for the configurations to actually work in the real world.

Management

The first aspect of the configuration was common across all of the Juniper Networks platforms and provided a common means to access and monitor the devices. The management traffic was carried in-band, on the same transmission facilities as the working traffic. Where possible, the management traffic was carried on a separate VLAN. The management traffic was not carried on a VPLS instance.

The management configuration was composed of multiple parts—syslog settings, firewall settings, prefix lists, and access settings, reviewed here one at a time.

lo0.0

The loopback interface was the management portal to the devices. All devices had a loopback interface defined, rather than the more traditional *fxp0* management port. The source addresses for the other management functions (*syslog* and *snmp*) used the loopback port for a source address. The loopback interface was protected by a firewall filter that limited the traffic to the interface. A common configuration for the loopback interface was:

```

lo0 {
    unit 0 {
        family inet {
            filter {
                input LOCKDOWN;
            }
            address 10.102.191.21/32;
        }
    }
}

```

The firewall filter referenced in the loopback configuration provided access to the device from a select set of management resources. The firewall configuration used prefix lists to manage these resources. The firewall filter and the prefix list configuration were:

```

policy-options {
    prefix-list SECURE-SSH {
        10.102.74.0/26;
        10.102.18.192/27;
        192.168.141.0/24;
    }
    prefix-list SECURE-RADIUS {
        192.168.141.0/24;
    }
    prefix-list SECURE-DNS {
        192.168.141.0/24;
    }
    prefix-list SECURE-SYSLOG {
        192.168.141.0/24;
    }
    prefix-list SECURE-SNMP {
        192.168.141.0/24;
    }
    prefix-list SECURE-SNMPTRAP {
        192.168.141.0/24;
    }
    prefix-list SECURE-FTP {
        192.168.141.0/24;
    }
    prefix-list SECURE-NTP {
        10.102.72.1/32;
        192.168.141.0/24;
    }
    prefix-list SECURE-DHCP {
        10.102.84.1/32;
        192.168.7.1/32;
        192.168.141.2/32;
        192.168.141.3/32;
    }
    prefix-list SECURE-ICMP {
        10.102.1.100/32;
        192.168.141.0/24;
    }
}

```



```

    }
    prefix-list SECURE-BGP {
        10.102.0.1/32;
        10.102.0.2/32;
    }
}

```

The use of prefix lists provided a better means of controlling the access points of the network. The prefix lists could be updated more easily than the actual firewall terms. The firewall configurations remained generic and referenced the prefix lists. This also allowed portions of the network to be managed by different engineering groups, while keeping a standardized protection in place. The components of the firewall filter were:

```

firewall {
    family inet {
        filter LOCKDOWN {
            term ICMP-NO-POLICER {
                from {
                    prefix-list {
                        SECURE-ICMP;
                    }
                    protocol icmp;
                    icmp-type [ echo-reply echo-request time-exceeded
                                unreachable ];
                }
                then accept;
            }
            term ICMP {
                from {
                    protocol icmp;
                    icmp-type [ echo-reply echo-request time-exceeded
                                unreachable ];
                }
                then {
                    policer DOS-PROTECTION;
                    accept;
                }
            }
            term TRACEROUTE {
                from {
                    packet-length [ 82 40 ];
                    protocol udp;
                    port 33435-33523;
                }
                then {
                    policer DOS-PROTECTION;
                    accept;
                }
            }
        }
    }
}

```

The first couple of terms limited what ICMP traffic could get to the routing engine and the amount of ICMP traffic. Note that the policer referenced looked only at the ICMP traffic. There is an overall traffic limiter built into Junos that blocks storms of traffic to the RE. This rate limiter is applied by default and affects all traffic:

```
term SSH {
  from {
    prefix-list {
      SECURE-SSH;
    }
    protocol tcp;
    port ssh;
  }
  then accept;
}
term RADIUS {
  from {
    prefix-list {
      SECURE-RADIUS;
    }
    protocol [ tcp udp ];
    port radius;
  }
  then accept;
}
term DNS {
  from {
    prefix-list {
      SECURE-DNS;
    }
    protocol udp;
    port domain;
  }
  then accept;
}
term SYSLOG {
  from {
    prefix-list {
      SECURE-SYSLOG;
    }
    protocol udp;
    port syslog;
  }
  then accept;
}
term SNMP {
  from {
    prefix-list {
      SECURE-SNMP;
    }
    protocol [ udp tcp ];
    port snmp;
  }
}
```

```

    }
    then accept;
}
term SNMPTRAP {
  from {
    prefix-list {
      SECURE-SNMPTRAP;
    }
    protocol [ udp tcp ];
    port snmptrap;
  }
  then accept;
}
term FTP {
  from {
    prefix-list {
      SECURE-FTP;
    }
    protocol tcp;
    port [ ftp ftp-data ];
  }
  then accept;
}
term NTP {
  from {
    prefix-list {
      SECURE-NTP;
    }
    protocol udp;
    port ntp;
  }
  then accept;
}
}

```

The next set of terms allowed management traffic from secured sources to the routing engine. Because of these protections, more services were allowed on the device (*ftp*, for instance). There was a trade-off between services and security here, but because a fire-wall filter was used to protect the services, this was considered a good design decision:

```

term OSPF {
  from {
    protocol [ ospf igmp ];
  }
  then accept;
}
term BGP {
  from {
    prefix-list {
      SECURE-BGP;
    }
    protocol tcp;
    port bgp;
  }
}

```



```

        secret "$9$/.rItpBleWdVVLxjeonljkhnlhfks;la33/CuBEM87";
        source-address 10.102.191.21;
    }
login {
    class FULL-ACCESS {
        idle-timeout 30;
        login-alarms;
        permissions all;
    }
    class READ-ACCESS {
        idle-timeout 30;
        login-alarms;
        permissions [ firewall interface network routing system
            view view-configuration ];
        allow-commands "show log|clear interfaces statistics";
        deny-commands "(ssh)|(telnet)";
    }
    user fullaccess {
        authentication {
            encrypted-password "$1$aedfkiko3dsm_ka /";
        }
    }
    user localadmin {
        class super-user;
        authentication {
            encrypted-password "$1$cyMbsb/764YRg1";
        }
    }
    user readonly {
        class READ-ACCESS;
    }
}
}

```

Additional components were used to secure the device: RADIUS, restricted services, a login banner, event logging, and archiving. The authentication order specified that a RADIUS server would provide the initial authentication verification. If this system failed, a local database was present to support local authentication. The login users *fullaccess* and *readonly* were RADIUS user aliases (these aliases are mapped to real usernames in the RADIUS server). The login user *localadmin* is a local user that allows access to the device in the event that RADIUS is not responding correctly. (If RADIUS is down, the *fullaccess* user will also work, but if RADIUS is misbehaving rather than failing to respond, the *fullaccess* user might still not be able to access the device.) The final piece of the access puzzle is the root login. This allows local access to the full device capabilities:

```

system {
    services {
        ssh {
            root-login deny;
            protocol-version v2;
        }
    }
}

```

```

        connection-limit 10;
        rate-limit 10;
    ftp;
}
}

```

The services that were allowed to be supported on the devices were *ssh* and *ftp*. These are both protected by the LOCKDOWN filter and the prefix lists associated with authorized source addresses. Although *ftp* is considered a nonsecure protocol, the configuration as shown provides a sufficient level of protection for the services offered (gathering troubleshooting files from the device).

While no login banner will actually stop a hacker from breaking into a system, they are a required component if the owners of the system wish to press charges against the hacker.



For some reason, locking the door to your house is not a legal deterrent; you must post a sign that says that only permitted personnel are allowed in the house, and all others are subject to prosecution. I must be missing something—if a hacker is caught in a system that was locked down, why isn't that breaking and entering just like breaking a window and barging into a private home?

There are banners that are the standard, and there are banners that stand above the standard. This client actually went so far as to grab ASCII art and add it to the standard security banner. The results looked really strange in the configuration, but pretty spectacular from the login screen. The configuration for this banner was:

```

login {
    message "\n      db      `7MMF' `YMM'      .g8""bgd .g8""8q.
`7MMM.      ,MMF'\n\n      ;MM:      MM .M'      .dP'      `M .dP'
`YM. MMmb   dPMM \n\n      ,V^MM.      MM .d"      dM'      `dM'
`MM M YM    ,M MM \n\n      ,M `MM      MMMMM.      MM      MM
MM M Mb M' MM \n\n AbmmmqMA      MM VMA mmmmm MM.      MM.
,MP M YM.P' MM \n\n A'      VML MM `MM.      `Mb.      ,` `Mb.
,dP' M `YM' MM \n\n.AMA. .AMMA..JMML. MMb.      `bmmmd'
`"bmmmd"' .JML. `` .JMML.
\n\n***** Warning Notice *****\n\n
This system is restricted solely to AK-COM authorized users for\n
legitimate business purposes only. The actual or attempted\n
access, use, or modification of this system is strictly\n
prohibited by AK-COM. Unauthorized users are subject to Company\n
disciplinary proceedings and/or criminal and civil penalties under\n
state, federal, or other applicable domestic and foreign laws. The\n
use of this system may be monitored and recorded for administrative\n
and security reasons. Anyone accessing this system expressly consents\n
to such monitoring and is advised that if monitoring reveals possible\n
evidence of criminal activity,

```

```

AK-COM may provide the evidence of\n such activity to law enforcement officials.
All users must comply\n with AK-COM company policies regarding the
protection of AK-COM\n information assets.
\n\n ***** Warning Notice *****\n\n";
}

```

When a user logged into the devices, here's what she saw:

```

      db      `7MMF' `YMM'      .g8""bgd .g8""8q. `7MMM.      ,MMF'
;MM:      MM .M'      .dP'      `M .dP'      `YM. MMMb dPMM
,V^MM.      MM .d"      dM'      `dM'      `MM M YM ,M MM
,M `MM      MMMMM.      MM      MM      MM M Mb M' MM
AbmmmqMA      MM VMA mmmmm MM.      MM.      ,MP M YM.P' MM
A'      VML MM `MM.      `Mb.      ,`Mb.      ,dP' M `YM' MM
.AMA. .AMMA..JMML.      MMb.      `bmmmd' `bmmmd" .JML. ` .JMML.
***** Warning Notice *****
This system is restricted solely to AK-COM authorized users for
legitimate business purposes only. The actual or attempted
unauthorized access, use, or modification of this system is strictly
prohibited by AK-COM. Unauthorized users are subject to Company
disciplinary proceedings and/or criminal and civil penalties under
state, federal, or other applicable domestic and foreign laws. The
use of this system may be monitored and recorded for administrative
and security reasons. Anyone accessing this system expressly consents
to such monitoring and is advised that if monitoring reveals possible
evidence of criminal activity, AK-COM may provide the evidence of
such activity to law enforcement officials. All users must comply
with AK-COM company policies regarding the protection of AK-COM
information assets.
***** Warning Notice *****

```

I asked who the ASCII artist was and was told that <http://patorjk.com/software/taag/> converted text to any number of ASCII art styles—it's nice to see someone taking time to play now and then!

The next part of securing the devices was to record the events happening on the devices. The syslog stanza has an equal number of on-device files and remote device files, so when an event was observed, the logging system was unlikely to miss it. The syslog system included default syslog files and also those specific to the issues seen by the client; the use of NTP assured that the logs from all the devices would be synced in time.



Did you know that Alaska has its own time zone, but some of the Aleutian Islands are actually in the next time zone?

The syslog and NTP stanzas looked like this:

```

system {
  syslog {
    user * {

```

```

        any emergency;
    }
    host syslog.ak-com.net {
        any any;
        facility-override local7;
    }
    file auth.log {
        authorization any;
        archive size 1m files 1;
    }
    file firewall.log {
        firewall any;
        archive size 5m files 3;
    }
    file interfaces.log {
        any info;
        match .*SNMP_TRAP_LINK.*;
        archive size 1m files 3;
    }
    source-address 10.102.72.1;
}
ntp {
    boot-server 192.168.141.2;
    server 192.168.141.2;
    server 192.168.141.3;
    source-address 10.102.72.1;
}
}

```

The final part of securing the devices was recording the configurations to a secure server. The system's archival capabilities allowed the configurations to be saved on the devices, but now when a commit is performed they are also sent to a server. The configuration for this capability is:

```

system {
    archival {
        configuration {
            transfer-on-commit ;
            archive-sites {
                scp://admin@archive.ak-com.com:config/MX/password
                admin123;file://config/MX/config;
            }
        }
    }
}

```

To communicate with the company's network operations center, each device had an SNMP configuration that allowed an SNMP agent to poll the devices for bandwidth measurements and receive traps from the devices in the event of failures. The SNMP stanza was very basic:


```

snmp {
    name Kodiak-1;
    location "Kodiak Island";
    contact "AK-CON NOC 1-800-123-4567";
    community "eert5c$" {
        authorization read-only;
    }
    community "Gfesb765#5" {
        authorization read-only;
    }
    trap-options {
        source-address lo0;
    }
    trap-group 1 {
        version v2;
        categories {
            authentication;
            chassis;
            link;
            remote-operations;
            routing;
            startup;
            rmon-alarm;
            vrrp-events;
            configuration;
            services;
        }
        targets {
            192.168.141.55;
        }
    }
}

```

Protocols

Logically the next configuration portion should be the interfaces, but due to the wide range of devices in use, the interface section is instead shown in the device-specific configurations. The next portion of the configuration to cover is device generic—the protocols running on the network's routers. The network used OSPF area 0 as an IGP, and the limited numbers of routers allowed the network to operate with little convergence delay. BGP was used throughout to support the services. Internal and external BGP peerings were supported in the network. MPLS was supported with LDP on all the routers. This made most of the protocols section very generic. The only differences were the interfaces that were assigned to the routing protocols. In the following examples, a generic set of interfaces (*ge-0/0/1* and *ge-0/0/2*) is used. One part of the configuration that I took particular note of was that traceoptions were set up for each protocol and deactivated—this type of configuration foresight allows quick troubleshooting and keeps a standard operating procedure for tracing issues.

MPLS

The MPLS configuration identified specific interfaces in the area and used authentication to prohibit rogue routers from being connected to the network. Here's a configuration example:

```
protocols {
  mpls {
    inactive: traceoptions {
      file mpls.log size 5m files 2;
      flag error;
      flag state;
    }
    interface ge-0/0/0.0;
    interface ge-0/0/1.0;
  }
}
```

Also associated with the MPLS protocol was that all interfaces carrying MPLS traffic had to have that protocol enabled in the logical interfaces.

BGP

The next protocol in the stack was BGP. In this engagement, BGP was the workhorse protocol for deploying services throughout the network. Managed Internet (*family inet unicast*), L3VPN (*family inet-vpn*), and VPLS (*family l2vpn-signaling*) were all deployed with this protocol. To minimize the configuration tasks for BGP, two route reflectors were used in the core of the network. The use of route reflectors allowed each of the remote devices to have a generic BGP configuration for the internal group. For those with external connections, customer-specific groups were created along with the routing policies.

The BGP routing policies were kept to a minimum. Route filtering was performed at the edges to the network, keeping the internal BGP operation free and open (L3VPN and VPLS policies were also kept to a minimum by the use of the route target commands, as we'll see later):

```
bgp {
  inactive: traceoptions {
    file bgp.log size 1m files 2;
    flag state;
  }
  log-updown;
  group iBGP {
    type internal;
    local-address 10.102.72.1;
    neighbor 10.102.0.1 {
      description "Peer To Anchorage-1";
      family inet {
        unicast;
      }
    }
  }
}
```

```

        family inet-vpn {
            unicast;
        }
        family l2vpn {
            signaling;
        }
        authentication-key "$9$ga-Ysdb G6eer-baZD";
    }
neighbor 10.102.0.2 {
    description "Peer To Anchorage-2";
    family inet {
        unicast;
    }
    family inet-vpn {
        unicast;
    }
    family l2vpn {
        signaling;
    }
    authentication-key "$9$ -YoGUH.eedfa-baZD";
}
}
}

```

An example of an external BGP group with route filtering policies is shown next. In this case, the client used a group configuration to contain all the configuration for the customer. The group configuration allows easy setup and easy cleanup when customers come and go. It contains the BGP configuration for the customer, the interface to the customer, the policies for the customer's prefixes, and a policer to regulate the amount of traffic the customer has contracted for.

The autonomous system (AS) numbers for the configuration are 1234 for our client (yes, it is made up) and 65432 for the customer. The local AS number was set as part of the routing options:

```

routing-options {
    autonomous-system {
        1234;
    }
}

```

Without the above stanza, the internal BGP stanza would not commit. Once that was added, the external BGP configuration group was added:

```

groups {
    PETRO {
        interfaces {
            fe-0/2/0 {
                no-traps;
                speed 100m;
                mtu 4500;
                link-mode full-duplex;
            }
        }
    }
}

```



```

        then {
            community add FROM-PETRO;
            next-hop self;
            accept;
        }
    }
    term REJECT {
        then reject;
    }
}
policy-statement 1234-TO-65432-DEFAULT {
    term DEFAULT {
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then accept;
    }
    term REJECT-ALL {
        then reject;
    }
}
community FROM-PETRO members 65432:10;
}
}
firewall {
    policer 20M {
        if-exceeding {
            bandwidth-limit 20m;
            burst-size-limit 100k;
        }
        then discard;
    }
}
}

```

This generic configuration provides the customer with only a default BGP route—if a full Internet routing table were requested, the 1234-to-65432 policy would be altered to allow the full routing table to be transmitted to the remote router. The import policies protect the client from private addresses and addresses that are too long. These are not important for this client as they only have a single prefix to export, but it is another protection that we installed for the client to prohibit problems from customers. In this case, the import policy included the *next-hop self* statement. This prevents the loss of prefixes due to OSPF errors for interfaces and/or inclusion of static routes. Finally, each incoming prefix was tagged with a community string that identified the origin of the prefix. These were used at the route reflectors to filter prefixes.

OSPF

The next protocol in the stanza is the IGP OSPF. Area 0 was used for all the routers, as the network was not large enough to bother with multiple areas. Authentication was used to prevent rogue routers. On some routers an export policy was used to advertise static routes that were needed to reach “different” parts of the network, but we tried to keep these to a bare minimum:

```
ospf {
  inactive: traceoptions {
    file ospf.log size 1m files 2;
    flag state;
    flag error;
  }
  area 0.0.0.0 {
    interface ge-0/0/0.0 {
      authentication {
        md5 1 key "$9$eJ.KXNwssfsgCA1Iws24JD";
      }
    }
    interface ge-0/0/1.0 {
      authentication {
        md5 1 key "$9$eJ.KsdfgsgsgCA1Iws24JD";
      }
    }
  }
}
```

LDP was the MPLS label-switched path (LSP) control protocol of choice for the customer. The star topology had no need for traffic engineering, and convergence times were such that failure recovery was very quick. Running LDP is an easy way to create LSPs across the network. With the use of route reflectors, resource reservation protocol (RSVP) static routes or false LSPs are not needed:

```
ldp {
  inactive: traceoptions {
    file ldp.log size 5m files 2;
    flag error;
    flag state;
  }
  interface ge-0/0/0.0;
  interface ge-0/0/1.0;
}
```

Core Router Configurations

With the protocols installed, all the generic portions of the configurations were covered. The next set of configuration options was the device-specific configurations; these were added or merged into the generic configurations on the devices.

The device-specific configurations start with the core routers. These had two position-specific elements: the route reflector configuration, which replaced the generic internal BGP configuration, and the Internet access BGP configurations, which provided Internet service to the entire network. At these locations, OSPF was modified to provide a default route to the network as well. This first portion displays the Internet access BGP configurations:

```
protocol {
  bgp {
    traceoptions {
      file bgp.log size 1m files 2;
      flag state;
    }
    log-updown;
    group EBGp-Internet {
      type external;
      export 1234-T0-INTERNET
      import INTERNET-T0-1234
      neighbor 172.16.90.93 {
        description "Peer To L3 (AS 77)";
        authentication-key "$9$kihgkolkn00^jikfa23d/9";
        peer-as 77;
      }
      neighbor 172.16.83.226 {
        description "Peer To AK-COM (AS 70)";
        peer-as 70;
      }
    }
  }
}
```

The referenced import policies accept all the routes and attach a service-specific community string. The export policies provide the prefixes that are accepted from customers and the local prefix aggregations. The policies and the related aggregates are:

```
routing-options {
  aggregate {
    route 192.168.136.0/22;
    route 10.102.0.0/20;
    route 10.102.16.0/20;
    route 10.102.32.0/20;
    route 10.102.48.0/20;
    route 10.102.64.0/20;
    route 10.102.80.0/20;
    route 10.102.96.0/20;
    route 10.102.112.0/20;
  }
}
policy-options {
  prefix-list ORIGINATED-ROUTES {
    10.102.32.0/20;
    10.102.64.0/20;
    10.102.80.0/20;
```

```

    10.102.96.0/20;
    192.168.136.0/22;
}
prefix-list ORIGINATED-ROUTES-DEPREF {
    10.102.0.0/20;
    10.102.16.0/20;
    10.102.48.0/20;
    10.102.112.0/20;
}
policy-statement 1234-TO-INTERNET {
    term ORIGINATED-ROUTES {
        from {
            protocol aggregate;
            prefix-list ORIGINATED-ROUTES;
        }
        then accept;
    }
    term ORIGINATED-ROUTES-DEPREF {
        from {
            protocol aggregate;
            prefix-list ORIGINATED-ROUTES-DEPREF;
        }
        then {
            as-path-prepend 1234;
            accept;
        }
    }
    term FROM-PETRO {
        from {
            protocol bgp;
            community FROM-PETRO;
        }
        then {
            as-path-prepend 1234;
            accept;
        }
    }
    term REJECT {
        then reject;
    }
}
policy-statement INTERNET-TO-1234 {
    term FROM-INTERNET-REJECT {
        from {
            route-filter 10.102.0.0/20 upto /32;
            route-filter 10.102.16.0/20 upto /32;
            route-filter 10.102.32.0/20 upto /32;
            route-filter 10.102.48.0/20 upto /32;
            route-filter 10.102.64.0/20 upto /32;
            route-filter 10.102.80.0/20 upto /32;
            route-filter 10.102.96.0/20 upto /32;
            route-filter 10.102.112.0/20 upto /32;

```



```

        route-filter 192.168.136.0/22 upto /32;
        route-filter 0.0.0.0/0 exact;
    }
    then reject;
}
term FROM-INTERNET {
    from {
        protocol bgp;
    }
    then {
        community + FROM-INTERNET;
        accept;
    }
}
term REJECT {
    then {
        reject;
    }
}
}
community FROM-PETRO members 65432:10;
community FROM-INTERNET members 7077:10;
}

```

A couple of notes about this configuration:

- To provide a level of load sharing for incoming traffic, some of the aggregates were prepended with the local AS for this core router. The other core router swapped the group contents.
- The use of prepending to similar-level ISPs provided a means for the ISPs to load-share incoming traffic to our client. In the outgoing direction, the client's traffic went to the closest ISP (core #1 or core #2).
- The import policy rejected any prefix that was originated by the client, to protect the client from loops.
- This is a single customer prefix being exported; the configuration was repeated for all the customers that were being provided ISP service.

The route reflector BGP configuration listed the remote BGP sites in a single group. Each remote site was adjusted for the services offered at that site. A couple of generic export policies were used for each site, and incoming BGP routes were accepted whole-sale:

```

protocols {
    bgp {
        deactivated: traceoptions {
            file bgp.log size 1m files 2;
            flag state;
        }
    }
}

```

```

log-updown;
group iBGP-32328 {
    type internal;
    local-address 10.102.0.2;
    authentication-key "$9$t4Khu01EcrML7.gsT-Vk.Pf3eKM8-baZD";
    cluster 10.102.0.2;
    neighbor 10.102.0.1 {
        description "Peer To CoreRtr-1";
        family inet {
            unicast;
        }
        family inet-vpn {
            unicast;
        }
        family l2vpn {
            signaling;
        }
        authentication-key "$9$.Pff3eKM8-baZD";
    }
    neighbor 10.102.0.3 {
        description "Peer To DistRtr-1";
        export DEFAULT_ONLY
        family inet {
            unicast;
        }
        family inet-vpn {
            unicast;
        }
        family l2vpn {
            signaling;
        }
        authentication-key "$9$.Pff3eKM8-baZD";
    }
    neighbor 10.102.0.4 {
        description "Peer To DistRtr-2";
        family inet-vpn {
            unicast;
        }
        family l2vpn {
            signaling;
        }
        authentication-key "$9-Vk.Pff3eKM8-baZD";
    }
    neighbor 10.102.72.1 {
        description "Peer To DistRtr-3";
        family l2vpn {
            signaling;
        }
        authentication-key "$9Vk.Pff3eKM8-baZD";
    }
}

```

```

    }
  }
}

```

In this configuration example, three of the remote distribution routers are shown. For *DistRtr-1*, all services (Internet access, L3VPNs, and VPLS) are offered. The Internet access delivers a default route only, so that policy is included at the neighbor level.



We had debates about the restriction of the full routing table and neighbor-specific export policies. On the one hand, when a new service is offered, more than the remote locations need to be touched. But on the other hand, having the bandwidth and local memory being used up for 700,000 routes that are never used seems to be a waste. We ended up with the more conservative approach. As services are added, the route reflectors need to be updated as well as the remote locations.

The policy referenced is the same used in the distribution routers:

```

policy-options {
  policy-statement DEFAULT-ONLY {
    term DEFAULT {
      from {
        route-filter 0.0.0.0/0 exact;
      }
      then accept;
    }
    term REJECT-ALL {
      then reject;
    }
  }
}

```

The default route was added to the routing table as a static route, which gave the client more control over the default than having it arrive from the ISPs. It was also exported to OSPF for use in the normal routing of the network. Outgoing load balancing is performed by the use of the static routes and a load-balancing policy:

```

routing-options {
  static {
    route 0.0.0.0/0 next-hop {
      172.16.90.93;
      172.16.83.226;
    }
    forwarding-table {
      export LOAD-BALANCE;
    }
  }
}
policy-options {

```

```

    policy-statement {
        policy-statement LOAD-BALANCE {
            then {
                load-balance per-packet;
            }
        }
    }
}

```

This policy allows both ISP routes to be installed in the forwarding table, and per-flow load balancing is performed on the outgoing streams.

Distribution Switch Configurations

In the locations that supported low MTU levels, the customer-facing device was a switch rather than the normal distribution router, allowing for the delivery of the services with a link that supported only a lower MTU. The switches were all Juniper Networks EX4200s in a single or dual virtual chassis arrangement. The normal management configurations were used in these devices, with the change being in the customer-facing VLAN configurations. The use of 802.1ad dual VLAN tagging allowed the differentiation between customers and allowed the customers to add VLAN tags to their traffic. The configurations for these switches were:

```

interfaces {
    ge-0/0/0 {
        description "To DistRtr-1";
        mtu 4500;
        ether-options {
            no-auto-negotiation;
            link-mode full-duplex;
            speed {
                100m;
            }
        }
    }
    unit 0 {
        bandwidth 5m;
        family ethernet-switching {
            port-mode trunk;
            vlan {
                members all;
            }
        }
    }
}
ge-0/0/2 {
    description "VPLS-Customer-1";
    mtu 4500;
    ether-options {
        no-auto-negotiation;
        link-mode full-duplex;
        speed {
            100m;
        }
    }
}

```

```

    }
  }
  unit 0 {
    bandwidth 5m;
    no-traps;
    family ethernet-switching;
  }
}
vlan {
  unit 201 {
    family inet {
      filter {
        input LOCKDOWN;
      }
      address 172.29.64.94/30;
    }
  }
}
}
routing-options {
  static {
    route 0.0.0.0/0 next-hop 172.16.64.93;
  }
}
ethernet-switching-options {
  dot1q-tunneling {
    ether-type 0x8100;
  }
}
}
vpls {
  CUSTOMER-VPLS {
    vlan-id 88;
    interface {
      ge-0/0/2.0;
    }
    dot1q-tunneling {
      customer-vlans 1-4094;
    }
  }
  MGMT {
    vlan-id 201;
    l3-interface vlan.201;
  }
}
}

```

Two interfaces are defined, one connecting back to the core of the network and the other to the customer. In this example, the only customer is a VPLS customer. The management traffic for the switch uses a VLAN ID of 201. All VPLS customer traffic arrived at the core with a service provider tag of 88.

The static route was for management traffic. The normal protection firewalls and prefix lists were used too, but were attached to the management VLAN rather than the loop-back interface.

Distribution Router Configurations

The distribution routers were responsible for offering services to the customers. The existing configurations offered L3VPN service. We added the VPLS services to these devices. In the process of the update, we also cleaned up the configurations and modularized the L3VPN customers. All service configurations were created in a group, allowing the administrators to easily add, copy, and delete customers to/from the configurations. The routing instances used the *vrf-target* statement to eliminate the need for route policies for each customer. This is fine for pure customer installations, but when a customer wants an extranet, or filtering of sites, the policies are necessary. For the existing customers, this was not the case, so the simpler approach was used.

Here's a generic L3VPN group configuration:

```
L3VPN {
  interfaces {
    ge-0/0/2 {
      unit 0 {
        family inet {
          address 172.16.80.49/30;
        }
      }
    }
  }
  policy-options {
    policy-statement DEFAULT {
      from {
        route-filter 0.0.0.0/0 exact;
      }
      then accept;
    }
  }
  routing-instances {
    L3VPN {
      instance-type vrf;
      interface ge-0/0/2.0;
      route-distinguisher 1234:100;
      vrf-target target:1234:100;
      routing-options {
        protocols {
          ospf {
            export DEFAULT;
            area 0.0.0.10 {
              interface ge-0/0/2.0;
            }
          }
        }
      }
    }
  }
}
```

```

    }
  }
}
set apply-groups L3VPN

```

Here, the customer-facing interface is *ge-0/0/2.0*. If the remote device were a distribution switch, the interface would be a logical interface with a VLAN identified (for example, *ge-0/0/2.88*, VLAN ID 88). In this example, the customer is using OSPF as a routing protocol. The prefixes that are received from the customer are readvertised to the rest of the locations, and the default route is advertised back to this customer site.

A VPLS customer group configuration is very similar to the L3VPN. An example of a VPLS configuration is:

```

groups {
  VPLS {
    routing-instance {
      VPLS-cust {
        instance-type vpls;
        interface ge-0/3/0.88;
        route-distinguisher 1234:200;
        vrf-target target:1234:200;
        protocols {
          vpls {
            site-range 10;
            mac-table-size {
              4096;
            }
            no-tunnel-services;
            site VPLS-1 {
              automatic-site-id;
            }
          }
        }
      }
    }
  }
}

policy-options {
  community FROM-VPLS members target:1234:200;
}

interfaces {
  ge-0/3/0 {
    unit 88 {
      description "Ethernet to VPLS";
      encapsulation vlan-vpls;
      bandwidth 100m;
      traps;
      vlan-id 88;
      family vpls;
    }
  }
}

```

```

    }
}
apply-groups VPLS

```

We used a couple of VPLS features here that reduced the bookkeeping for the administrators. The first was automatic site numbering. Each site was given a unique site identifier, and the system figured out the numbering. We also used the *vrf-target* statement again to reduce the need for route policies.

The L3VPN and VPLS configuration groups were installed at each customer-facing router. The use of the distribution switch did not alter the router's configuration.

Rate Control

Those customers that were served over the low-bandwidth satellite circuits had an additional configuration in place. The satellite modems were sensitive to being overrun by the network, and when that happened, the traffic loss was unpredictable and often caused outages on the links. To solve this issue, a combination of policers and traffic shapers were used to control the rates of traffic accessing the modems.

The policers were configured facing the customer for incoming traffic. The filters were:

```

interface {
    ge-0/0/0 {
        unit 0 {
            family inet {
                filter input {
                    10m-POLICER;
                }
            }
        }
    }
}
firewall {
    family {
        inet {
            filter {
                10m_POLICER {
                    term 1 {
                        then {
                            count 10m;
                            policer 10m-POLICER;
                        }
                    }
                }
            }
        }
    }
}
policer {
    10m-POLICER {

```



```

        if-exceeding {
            bandwidth-limit 10m;
            burst-size-limit 1m;
        }
        then {
            discard;
        }
    }
}

```

For the shapers that faced the satellite modems, the configurations were very simple:

```

class-of-service {
    interfaces {
        ge-0/0/0 {
            shaping-rate 10;
        }
    }
}

```

The combination of the shapers, policers, and flow control on the satellite modems eliminated the buffer overflows and stopped outages for these finicky links.

CPE Switch Configuration

In some cases, the customer requested that the client provide a managed device onsite. These devices (typically Juniper Networks EX4200s) needed to be managed by the client's administrative staff. Since the customer was given full use of all the VLANs, the use of a VLAN for management was not supported. The solution to this issue was to move the Q-in-Q to the CPE device. This allowed the use of service provider VLAN tags as well as management VLAN tags on the circuit. The configuration of these devices became the same as that of the distribution switch shown above.

Conclusion

When network warriors enter an engagement, they often arrive at the scene with preconceived notions of what they are about to do for the client. The 12-hour flight from Vermont to Alaska provided me with many opportunities to create preconceived notions. I thought we were going into a simple Layer 2 virtual private network implementation, trading out one set of customer-facing protocols for another. The reality of the engagement was a deep dive into the interworkings of the protocols, their frame structures and operations. We had to dig out the reference materials and reacquaint ourselves with the basics of LANs and data networking. A common theme in this book is that on

every engagement, warriors learn. We learn, and we take the results and solve client issues. The solution to this client's issue included not only creating a set of configurations, but also redesigning the components of the network to allow it to operate in the limiting environment.

A side lesson of this engagement was that the most basic of networking components can necessitate a full redesign of the service. In this case, the MTU limitation of the satellite links caused the redefinition of the edge of the VPLS network. As networking professionals, warriors have to look at all the little details to assure a successful implementation. It is said that the devil is in the details, and this was a great case in point.

Internet Access Redress

Over my years as a network warrior, I have been involved with many tribes of warriors, and I have seen projects in almost every stage of implementation. In some cases, the warriors are called in after the fact, like custodians, to “clean up on aisle 4.” In other cases, we are brought in to draft the initial design, create the bill of materials, and take the project to a handoff to the client’s operations team. Most engagements are somewhere in between these two extremes. Such was the case with this engagement for a state government.

The call came for network warriors to engage in a retrofit of the Internet access for the government offices. When we arrived on site, we walked into an environment that reminded me of a dime-store novel. The main plot was to offer a better Internet experience. The subplots included jealousy, fiefdom protection, getting “the man” (whoever that was), and greed. The only subplot left out was sex, and that was probably happening too, but I didn’t want to go there.

If the reader has gathered anything from the various engagements described in this book, it should be that all engagements involve some level of suspense, political intrigue, and/or nontechnical elements. This one just happened to have them all—but hey, if it was easy, they wouldn’t call us *warriors*.

It took a while to find the players and assemble a tribe that was going to work together along the same plotlines (“let’s keep it simple here and focus on the Internet experience”). The tribe consisted of a Juniper sales engineer, who had been around this client from the beginning and knew the roots of their personnel problems but was also an incredible technical resource; select members of the client’s security team; and a couple of members of the client’s networking team. This might seem like a large team, but the task was formidable—this was not your average, simple Internet border router (IBR) replacement.

Objective

The objective of the engagement was to solve a number of problems being seen in the current Internet egress design. The problems were the result of the current system being created piecemeal without a central design. Over the 10+ years of its existence, many hands were involved in building the system, and the fingerprints of those individuals could be seen in all its facets. The system was a combination of Cisco, Checkpoint, Websense, Nokia, Extreme, and a host of other minor devices. The more pressing issues of the old system were:

Many firewalls

The design had over a dozen different firewalls, managed by different groups for different user groups. They were operating in an active/hot-standby relationship, often with a manual switchover requirement.

Many management systems

Each firewall, intrusion detection protection (IDP) system, virtual private network (VPN) appliance, and content filtering system had a different management platform, methodology, or administrator. A change to the system required the coordination of all these folks.

Many administrative domains

There were people focused on networking, network security, and application security, as well as many application (data center) groups. These folks all reported to different C-level bosses and had different views on what a good experience was.

Many functions

The existing design was built over the course of a decade and contained every possible feature, function, and fad that had become available during that timespan. Few of the systems had been retired. Most that had been retired were still in place, just powered down.

Static routing

Due to the complexity of the connectivity, the various active/passive paths, and Layer 2 interconnects, all routing in the system was statically configured.

Manual failover

Due to the architecture of the existing firewalls and the static routing, all recovery from equipment and link failures was a manual process, often requiring an engineer to report to the equipment rooms to configure and patch.

Single points of administration/knowledge

Cross-training between the devices was rare; different people working on the system had created knowledge silos, making them indispensable to its overall operation.

Low bandwidth

The system used routers and firewalls that were stressed both in terms of bandwidth and processor power. The result was an average throughput of around 100 Mbps.

Poor scalability

Due to the above issues, the existing design had no potential for growth without adding to the complexity and problems of the design.

The initial objective of the engagement was the creation of an Internet egress design that matched the salient capabilities of the existing design. These capabilities included:

A stateful firewall

The system had to provide firewall capabilities that followed the full session of traffic between users. It also had to provide logs of these sessions and be able to control access between numerous groups of users/networks.

IDP

The system had to provide an IDP mechanism to block malicious traffic.

Content filtering

This was a government network; all content had to pass strict content filtering rules for usage and subjects (no eBay shopping at work here!).

Logging and monitoring

The systems had to provide a clear view of what traffic was being allowed and what traffic was being denied. State governments, like the federal government, have strict auditing rules. The changes to the security devices must also be logged.

Added to the existing requirements were a number of requirements typical of any new installation of modern equipment:

Resilience

The design had to automatically fail over between primary and secondary elements. Failover should be transparent to the users.

High bandwidth

The design would use 10 Gbps links to multiple Internet service providers (ISPs). Both links should carry traffic in a load-balancing fashion.

Scalability

The design should allow for growth in the areas of users, bandwidth, and networks.

Manageability

The design should be managed from as few platforms as necessary.

If the tribe had been present at the start of this redesign battle, the initial design might have been different (but not by much). Instead, when we arrived on the field of battle, the design had been set and the equipment was in place. The decision had been made that the new Internet egress would be centered on a cluster of Juniper Networks SRX5800

Series Services Gateways for the Data Center. These monster-eating firewalls are capable of handling up to 150 Gbps of firewalled traffic. They take a big hit, but still will support over 25 Gbps of IDP traffic or IPSec traffic. Appropriately provisioned, the system will allow 20 million concurrent sessions (bidirectional traffic between two devices) and support the establishment of those sessions at a rate of a little over 350,000 sessions per second. All these sessions are monitored by the use of up to 80,000 security policies.

The SRX is also a high-speed router that can handle multiple copies of a full Internet routing table (up to 1,000,000 routes in total), 2,000 BGP peers, and 128 different instances of BGP in operation. The SRX5800 can also be a virtual platform supporting up to 2,000 security zones and virtual routing instances.

Although the installation was not going to tax all of these capabilities, it was going to employ most of these features.

Design

As highlighted, the existing design was a hodgepodge of devices, capabilities, and connectivity that looked as though it had been thrown into a blender and run on puree for 30 seconds. What a mess.

The root of the network was a Cisco-designed three-layer architecture (core, aggregation, and access) with the Internet egress attached at the core layer ([Figure 5-1](#)).

The egress switches held the system together. They interconnected the wide area routers, the Internet border routers, and the firewalls in a tangle of VLANs, Layer 3 routes, and trunk ports.

The traffic flow was not initially obvious to this warrior, and I've seen a lot of different flows. The traffic arrived (follow along on [Figure 5-1](#)) at the IBR, passed through the egress switch (and content filters), and was sent to the active Internet firewall for filtering. The traffic that passed through the firewall was then sent on to the second level of egress switches. At the second level of egress, the traffic was inspected by the IDP system and then forwarded to the core of the campus.

Trusted traffic (from the state WAN) was handled in a similar fashion, but through another firewall. Extranet traffic was also processed by this system and had its own firewall; the traffic had to pass through the trusted firewall prior to getting to the core network.

All VPN traffic was processed through this system as well. The VPN concentrators are not shown on the diagram, but they included two different sets (primary and secondary) of IPSec concentrators, and three different servers, each managed manually, performed the web filtering. The design was the picture of confusion and a nightmare to manage.

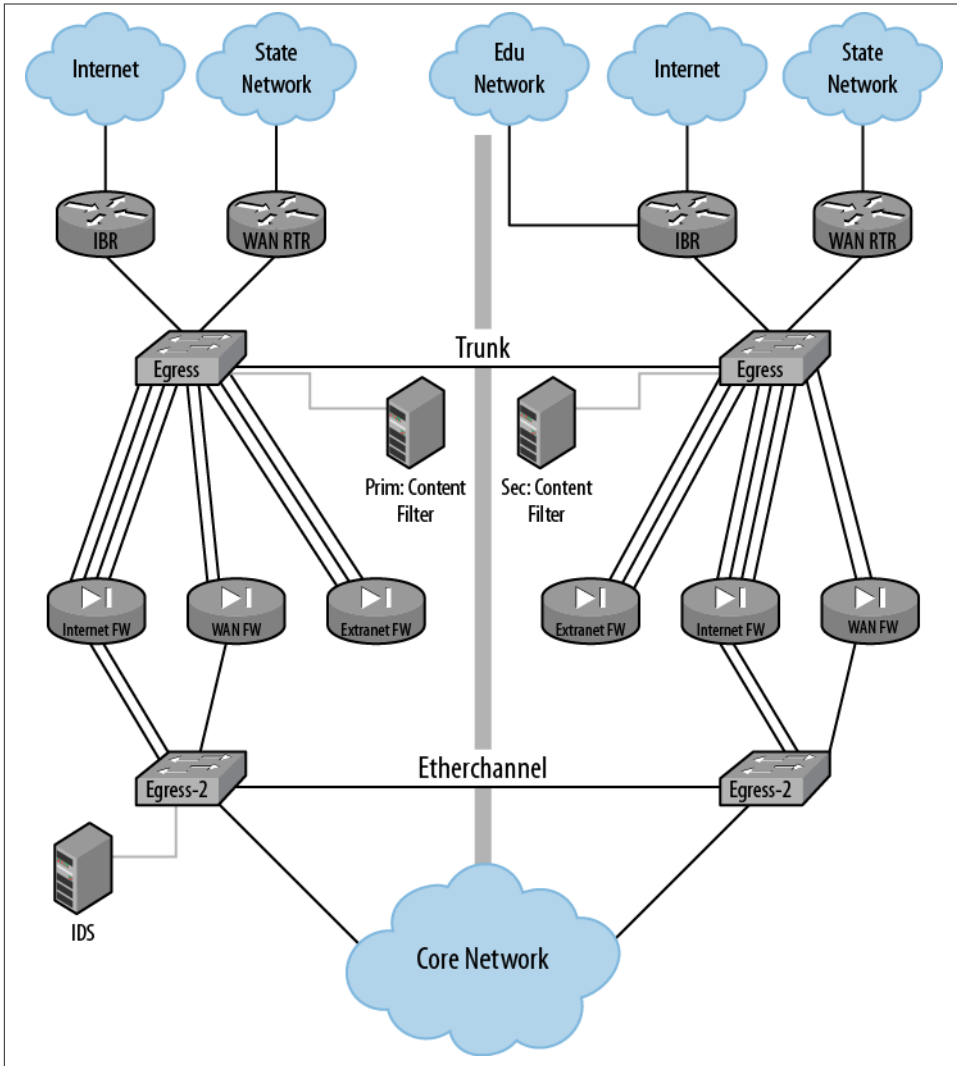


Figure 5-1. Old egress design

As an example of the complexity of managing this system, adding a site-to-site IPSec VPN extranet user to the system required the following steps to be performed:

1. Configuring the two VPN concentrators that handled the VPN security associations
2. Setting firewall rules in the extranet firewalls (both active and secondary)
3. Setting static routes in the extranet firewalls (both active and secondary)

4. Setting firewall rules in the egress firewalls (both active and secondary)
5. Setting static routes in the extranet firewalls (both active and secondary)
6. Setting VLANs in the egress first-tier switches (both active and secondary)
7. Adding static routes to the two-tier egress switches to be redistributed to the core

And remember that few of these steps were performed by the same team, let alone the same person on the same management platform. Most of the steps were done at the command line, or on a web interface directly connected to the device. Coordination between the firewall team and the networking team was required, and the content management team's and IDP team's devices needed to be updated to allow these new addresses in the core as well. It just kept getting more and more difficult to understand how this system actually functioned on a day-to-day basis.

The bright side of the engagement was that just about everybody was as confused and dismayed by the system as we were. Any new system had to be better than what was already in place, and agreement can be a fine thing to focus on.

As stated previously, the new design was based on a single SRX cluster. The firewall would be replacing the existing firewalls, routers, IDP appliances, and VPN appliances. The two areas where the SRX could not match the existing capabilities were:

Content filtering

Although the SRX Series Services Gateways for the Branch offer URL filtering and content filtering, the larger SRX5800s do not support this feature. The client was familiar with Websense as a content and URL filtering system. The decision was made to stay with that company.

Remote access VPNs

Again, a feature that the smaller branch SRXs offer with the Pulse client and a dynamic IPsec VPN is not supported on the data center SRXs. The initial design was to add a Juniper Networks SA6500 SSL VPN appliance for remote access. The majority of remote users are law enforcement and other mobile users around the state. The thought of changing these mobile users (very few of whom are technically savvy—no offense meant to law enforcement officers, but we're all specialized warriors; you don't want us playing with weapons, and we don't want you playing with computers) seemed out of the question. The decision was made that for this engagement, the existing IPsec clients and concentrators would remain in place for the remote access users.

The proposed design was as shown in **Figure 5-2** and was a picture of simplicity. The existing mayhem was replaced by a single system and management platform that contained all the features and functions of a dozen separate appliances. It offered better throughput, better reliability, improved scalability, and a response time to changes counted in the minutes rather than days, weeks, or even months.

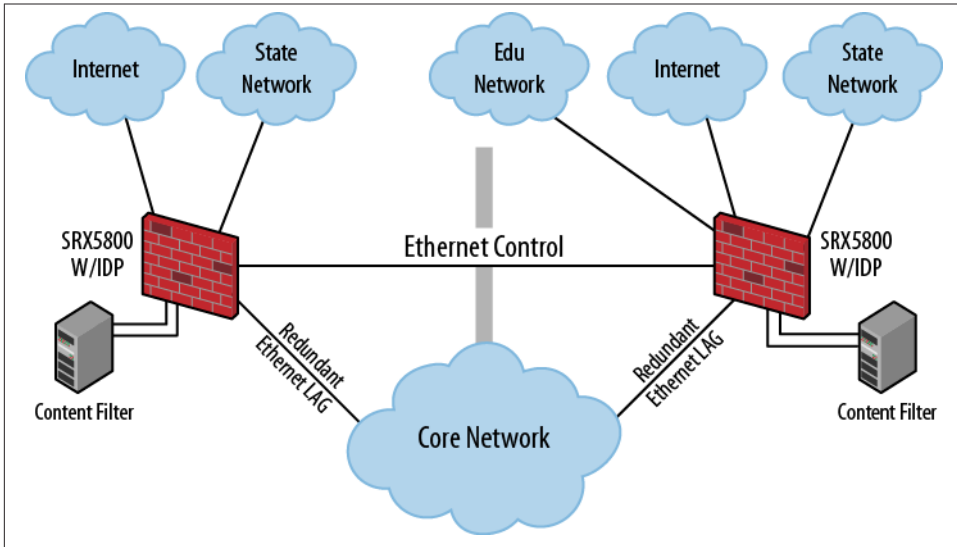


Figure 5-2. Initial design

The design had many benefits; it met the technical requirements for the Internet egress, an orderly migration was possible, and it gave the client the flexibility to offer new services to its customers. The problem with the design, as we found out quite soon, was that it crossed many administrative boundaries, causing turf wars within the state networking and security organizations. Thankfully, we didn't have to deal with that, as we are network warriors, not interdepartmental people managers.

Trade-offs

Prior to our arrival on the scene, the biggest trade-off decision had been made: the SRX5800 was the choice for the egress firewall. What remained to be done was hammer out the details of the implementation between the various teams. The list of major items to be considered was:

- Dynamic routing versus static routing
- IBR router and firewall versus firewall
- Integrated IDP versus standalone appliances
- Filter-based forwarding versus virtual firewalls
- Active/passive versus active/active firewalls

Routing

The old design used a combination of BGP and static routing for moving traffic into and out of the egress devices. This system, while stable, required constant manual intervention and tuning to keep up with changes in the requirements. It also lacked automatic failover capabilities. The new design could be integrated directly with the core routing system's OSPF area. Initially, the networking group balked at having the firewall perform the routing functions (more on that in [“IBR integration” \(page 144\)](#)). The decision was finally made that the firewall would operate with a dynamic routing protocol. The major reason was the automatic failover in the case of a failure of any components in the egress design. This decision also lowered the management burden for the design. Once the networking groups realized the routing capabilities of the SRX5800, it was not hard to gain agreement on this decision.

IBR integration

The existing design segregated the routing and firewall functions. The networking group ran the routers, and the security group operated the firewalls. The addition of a new feature or capability required the coordination of both groups (and an act of the state legislature!). Initially, the networking group was adamant that these two functions would be kept separate. If the decision had just been for the Internet access there might have been a different outcome, but there were to be two other routers included in the egress system: one to the state's wide area network (WAN) and the other to the education network (again a statewide entity, EDU). Each of these networks was to use the new system for Internet access. These routers were managed by yet another group, and both were aged and ready for retirement. The other groups were more than happy to relinquish control of Internet access and the management of these routers. Once these networks were on board, the networking group decided that it would be best to have all Internet access managed from a single point rather than multiple points. It was decided that the SRX5800 was to be the IBR, the WAN routers, and the firewall for all Internet egress.

The SRX5800 handled the BGP peerings to the Internet service providers as well as the WAN and the EDU networks. It also handled the OSPF adjacencies to the core network. This decision simplified the survivability design and again reduced the management requirements of the system.

The ultimate design actually created an IBR as a routing instance on the SRX5800. This virtual IBR received all ISP and EDU routes and sent a default route to the default routing instance. This allowed a clean separation of BGP and OSPF route tables and also allowed superb control of route selections for survivability and load sharing (more on that later).

IDP

The SRX5800 has an IDP capability (based on the same rule sets as the IDP standalone appliances from Juniper Networks) that is integrated into the security policies: IDP policies can be assigned to any traffic flowing through the device. The device suffers a performance hit when IDP is active, but this allows incredible flexibility in the IDP design. Our initial plan was to integrate the IDP into the SRX5800, but testing showed that traffic could be disrupted when mistakes were made with the IDP policies. While this was an outside chance, the application security group that was responsible for the IDP deployment was yet another administrative domain—this fell outside the responsibility domain of the egress group. This administrative issue led to the decision to remove the IDP features from the SRX and instead install standalone IDP appliances.

The comment was once made that if there existed the remotest possibility that those “rocket scientists” could disrupt the firewall’s operation, this would occur at the worst possible moment. We were always taught to think positive, and here was a case of positive thinking.

Filter-based forwarding

Filter-based forwarding refers to the capability of routing traffic based on some means other than the destination address of the IP packet. In this engagement, we needed to extract HTTP traffic and send it to the external content/URL filter servers. The initial tests showed that there were limitations on the use of filter-based forwarding on the SRX5800s. While these limitations were scheduled to be removed in a later version of Junos, they were present in the version that we were testing, so we had to look at different implementations.

During a design review, one of the other members of the tribe came upon the idea of splitting the firewall and installing the content/URL servers between the pieces, as shown in [Figure 5-3](#). That way, all servers could see all traffic, and they could pass on (or look at) any traffic that they pleased. Such a simple design—brilliant.

The split design also allowed a separation of other features (network address translation and BGP routing), reducing the complexities of the overall design.



It is amazing how a limitation in a device can lead to a better design for the client. Once again, this shows that there are always things to learn as a warrior, new tricks to be employed in other battles.

Clustering

The SRX5800 offers a survivable configuration named *clustering*. In a clustered SRX, the primary device performs the administrative functions while the secondary routing

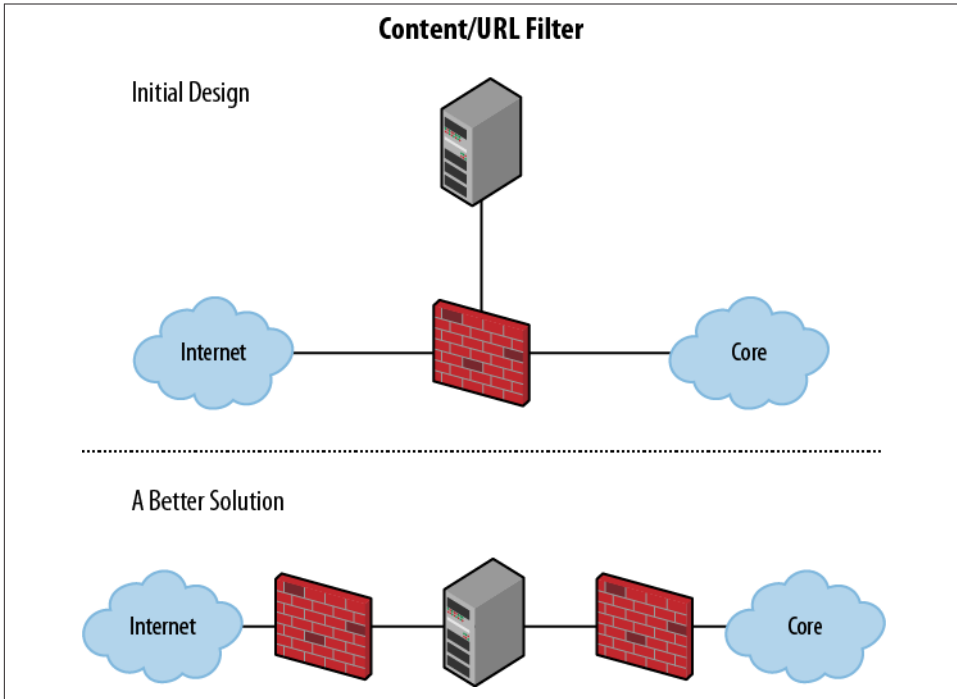


Figure 5-3. Content/URL filtering

engine is in *hot standby mode*, ready to take over in the case of a failure of the primary. The two devices that are clustered look like a single extended chassis (separated by miles, in our case). Port 1 exists in one data center and port 2 exists in the other data center, but they're both part of the same chassis. Very cool technology.

The technology has two operational modes, *active/passive* and *active/active*, as shown in [Figure 5-4](#). In an *active/passive* deployment, redundant Ethernet interfaces (one port on each chassis) are all active on one chassis, and in standby on the other. In an *active/active* deployment the active interface can be on either chassis. That is, redundant Ethernet (Reth) port 1 can be active on chassis 1 and passive on chassis 2, while redundant Ethernet port 2 can be active on chassis 2 and passive on chassis 1. While the *active/active* mode offers a better opportunity to load-balance traffic between the two chassis, it increases the complexity of the traffic flow and the configurations. Considering that, for the most part, the failure recovery was going to be covered by OSPF at Layer 3, the added capabilities of an *active/active* arrangement were not needed in this case. The design was kept simple with the *active/passive* design.

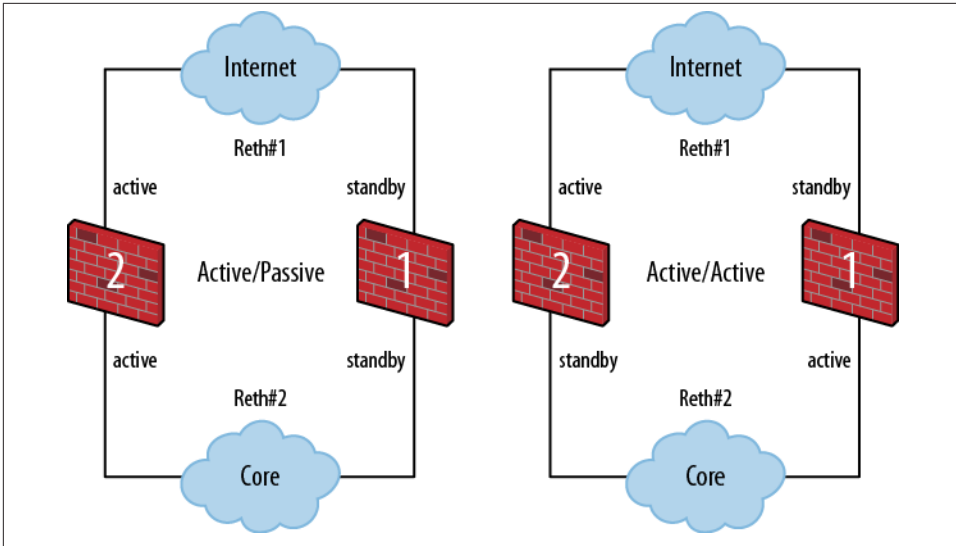


Figure 5-4. Clustering options

Configuration

Once the main trade-off decisions were made, it was time to get down to the details of the design: the configuration and prototype testing. For this engagement we had the luxury of having the SRX5800s racked, powered, and connected to a few test devices, so we could try different configurations and verify the operation of the configurations prior to going live.



Regardless of the number of cut-overs I've done and my confidence in my configurations, I always take the time and make the effort to verify all the configurations and features. You never know when something that you have done a hundred times will backfire on you because you did not double check.

After many false starts and “not quite” architectures, the following configurations provided the capabilities that were required. They are presented here in the logical sequence for configuring the firewall.

Clustering

The first configuration chore is the creation of the SRX cluster. This firewall is running an active/passive configuration with a single redundant Ethernet interface. The first part of the configuration defines the node-specific groups that allow the management of the

two SRXs as individuals. The group configurations are called in the main configuration and are active on the node being accessed. The groups contain the hostnames, the management addresses, and the default routing information for each node (in case the routing daemon, *ripd*, is not operational). The configuration for the node groups was:

```
groups {
  node0 {
    system {
      host-name egress-east;
      backup-router 10.10.10.97 destination 0.0.0.0/0;
      services {
        outbound-ssh {
          client nsm {
            device-id D5tdd;
            secret "$9$p9EFuIcLeWhsgdrsYFn/9u01Rh";
            services netconf;
            10.10.10.99 port 7804;
          }
        }
      }
    }
    interfaces {
      fxp0 {
        unit 0 {
          family inet {
            address 10.10.10.98/29;
          }
        }
      }
    }
  }
  node1 {
    system {
      host-name egress-west;
      backup-router 10.10.10.105 destination 0.0.0.0/0;
      services {
        outbound-ssh {
          client nsm {
            device-id 7F9A8A;
            secret "$9$jhfxfdrf/WLXxwYjik";
            services netconf;
            10.10.10.99 port 7804;
          }
        }
      }
    }
    interfaces {
      fxp0 {
        unit 0 {
          family inet {
            address 10.10.10.106/29;
          }
        }
      }
    }
  }
}
```



```

        fabric-options {
            member-interfaces {
                xe-11/0/0;
            }
        }
    }
    fab1 {
        fabric-options {
            member-interfaces {
                xe-23/0/0;
            }
        }
    }
}

```

The final portion of the base configuration is the system and SNMP stanzas. These hold nothing spectacular, so I will not bore you with the full configuration. However, one part of the system stanza that was new to me was the archival capabilities. The state used a secure server for the archiving of the configurations. Archiving was performed on each commit and sent to a server with the filename *srxegress.gz*. A time/date stamp was added to the transfer. The archival configuration was:

```

system {
    archival {
        configuration {
            transfer-on-commit;
            archive-sites {
                "scp://SRX@10.10.10.50/srx/config/srxegress.gz
                password Juniper123";
            }
        }
    }
}

```

Security

The security configuration for the egress firewall presented a couple of challenges: the IDP and content filtering requirements forced a virtual router implementation, and the network address translation (NAT) requirements called for some warrior imagination due to limits on the earlier version of Junos in use. The configurations presented here are in a logical order: first we added the virtual routers, the zones, the interfaces, and the policies. Once these were defined, we added the NAT and the logging.

Routing instances

Before I get into defining the interfaces and the security zones on the firewall, I should explain the virtual structure of the firewall. As mentioned in the trade-off section, the firewall was divided in two, and the content/URL filters were inserted between the two halves. What that also allowed was a separation of responsibilities between the two

virtual firewalls. The upper firewall (both east and west portions) looked like a single firewall interfacing with the external networks. The lower, virtual firewall was logically divided between east and west in a survivable arrangement (primary/secondary) and connected to the state's core network. This arrangement is depicted in [Figure 5-5](#), which shows three firewall partitions. The upper and lower separation is caused by routing instances (the Internet routing instance and the default routing instance), while the East and West divisions are the primary/secondary relationship of the default routing instance. The East and West divisions form a means of traffic separation (for example, the routes on the west side are preferred over the routes on the east side).

The interconnectivity between the two routing instances was provided by a set of aggregate Ethernet interfaces. These interfaces were connected through the content/URL filter servers (not shown on the diagram as they were pass-through devices). In the following sections, the configurations of the virtual routers and routing are examined.

Interfaces, zones, and policies

The primary security construct of the SRX firewall is the definition of the security zones and the interfaces that are assigned to those zones, as no traffic will be allowed to pass through the SRX without their definition. There is a set of rules for assigning zones in firewalls that are divided by routing instances. From the SRX training material:

- A strict hierarchical linkage exists between zones and interfaces.
- Logical interfaces are assigned to a zone.
- A logical interface cannot be assigned to multiple zones.
- Logical interfaces can also be assigned to a routing instance.
- A logical interface cannot be assigned to multiple routing instances.
- All logical interfaces in a zone must belong to the same routing instance.

For this engagement, there were two different zone arrangements. The Internet virtual router (VR) contained only two security zones: one for the education network and one for the other interfaces. The theory here was that routing would restrict traffic between the two ISPs, and the policies in the default routing instance would regulate what traffic was allowed to and from the Internet. We found it amusing that the folks at the state considered the education network to be the same threat level as the Internet.

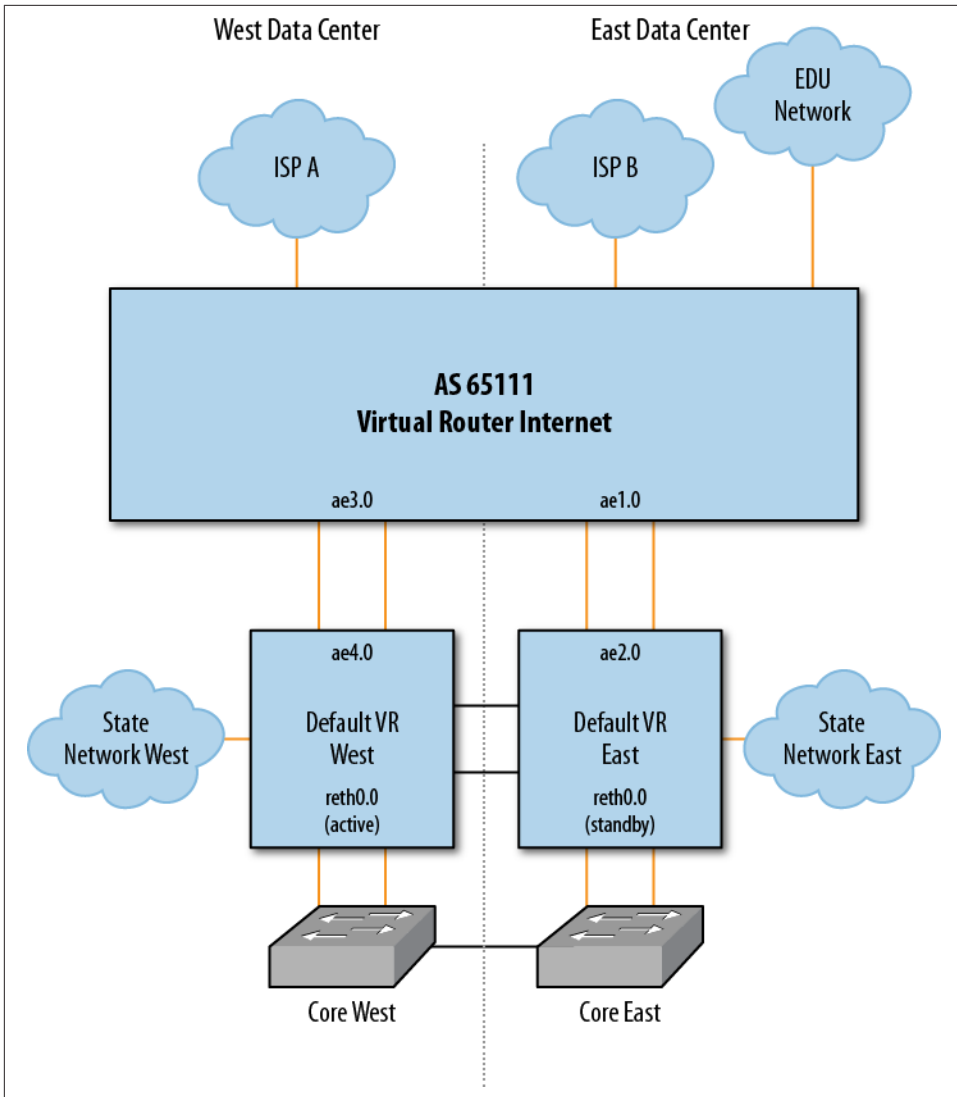


Figure 5-5. Virtual firewalls

The configuration for the Internet virtual router and the security policy for that zone looked like this:

```
security {
  zones {
    security-zone VR_internet {
      screen Internet-screen;
      host-inbound-traffic {
        system-services {
```

```

        ping;
        traceroute;
    }
    protocols {
        bgp;
    }
}
interfaces {
    ge-10/1/0.0; ## ISP A
    ae1.0;
    ae3.0;
    ge-22/1/0.0; ## ISP B
}
}

security-zone EDU {
    screen Internet-screen;
    host-inbound-traffic {
        system-services {
            ping;
            traceroute;
        }
        protocols {
            bgp;
        }
    }
    interfaces {
        ge-10/0/9.0; ## EDU
    }
}

policies {
    from-zone VR_internet to-zone VR_internet {
        policy 764915 {
            match {
                source-address any;
                destination-address any;
                application any;
            }
            then {
                permit;
                log {
                    session-close;
                }
            }
        }
    }
    from-zone EDU to-zone VR_internet {
        policy 764933 {
            match {
                source-address any;
                destination-address any;
                application any;
            }
        }
    }
}

```


The policies are wide open; the EDU network was not under the same administration as the other state users, so full access was allowed to the Internet.



An alert reader might question the policy names. When a firewall is imported to the Juniper Network and Security Manager (NSM), the firewall policy names are rewritten to an index rather than the originally defined names.

Counting was added to all the policies to allow troubleshooting of connectivity. Since all traffic to the Internet would pass through other policies, logging was not necessary.

The big difference between the Internet VR and the default VR for zones was the addition of the OSPF protocol and additional host inbound traffic services. Also, as this was where the actual security policies were created, there were numerous address book entries. The complete address books have been eliminated from this display:

```
security {
  security-zone Internet {
    address-book {
      address net_200.0.0.0/8 200.0.0.0/8;
      address net_201.0.0.0/8 201.0.0.0/8;
      address net_202.0.0.0/8 201.0.0.0/8;
      address-set Bad-European-guys {
        address net_200.0.0.0/8;
        address net_201.0.0.0/8;
        address net_202.0.0.0/8;
        ...
      }
    }
    host-inbound-traffic {
      system-services {
        ping;
        traceroute;
      }
      protocols {
        bgp;
      }
    }
    interfaces {
      ae0.0;
      ae2.0;
    }
  }
  security-zone inside {
    address-book {
      address net_10.100.100.42/32 10.100.100.42/32;
      address net_170.2.2.61 170.2.2.61/32;
      address-set mass-mailing-ips {
        address net_10.100.100.42/32;
```

```

        address net_10.100.100.32/32;
        ...
    }
    host-inbound-traffic {
        system-services {
            ssh;
            ping;
            traceroute;
            http;
        }
        protocols {
            ospf;
        }
    }
    interfaces {
        reth0.0;
    }
}
}
}
}

```

The policies associated with these zones represent the full security policy for the state; I won't spend time presenting them in detail here, but each policy is logged and counted. A similar security zone (as shown here) was established for the state networks, and security policies were installed to provide full connectivity between all the zones.

When we designed the security policies for the firewall, the existing firewall configurations from the old design were used as a template. These configurations were converted to the SRX format and added wholesale to the SRX. Once the firewalls were installed and operational, a policy cleanup was performed to eliminate the policies that were not used or that did not make sense for the traffic patterns. This method of migration guaranteed that no security holes were introduced into the system.

With the zones added, the structure of the firewall resembled [Figure 5-6](#).

Numerous zones are omitted from this diagram, but from a design perspective, they did not alter the configuration.

NAT

I have stated a couple of times in this chapter that the division of the firewall into two instances allowed a separation of features between the instances. A prime example of this feature split is that all NAT was performed in the Internet virtual router. Of all the portions of security configurations, NAT is often the most painful: it affects policies, address books, and troubleshooting. Putting it in the Internet VR assured that all the default VR policies and addresses were for private addressing, thus simplifying the overall design of the system.

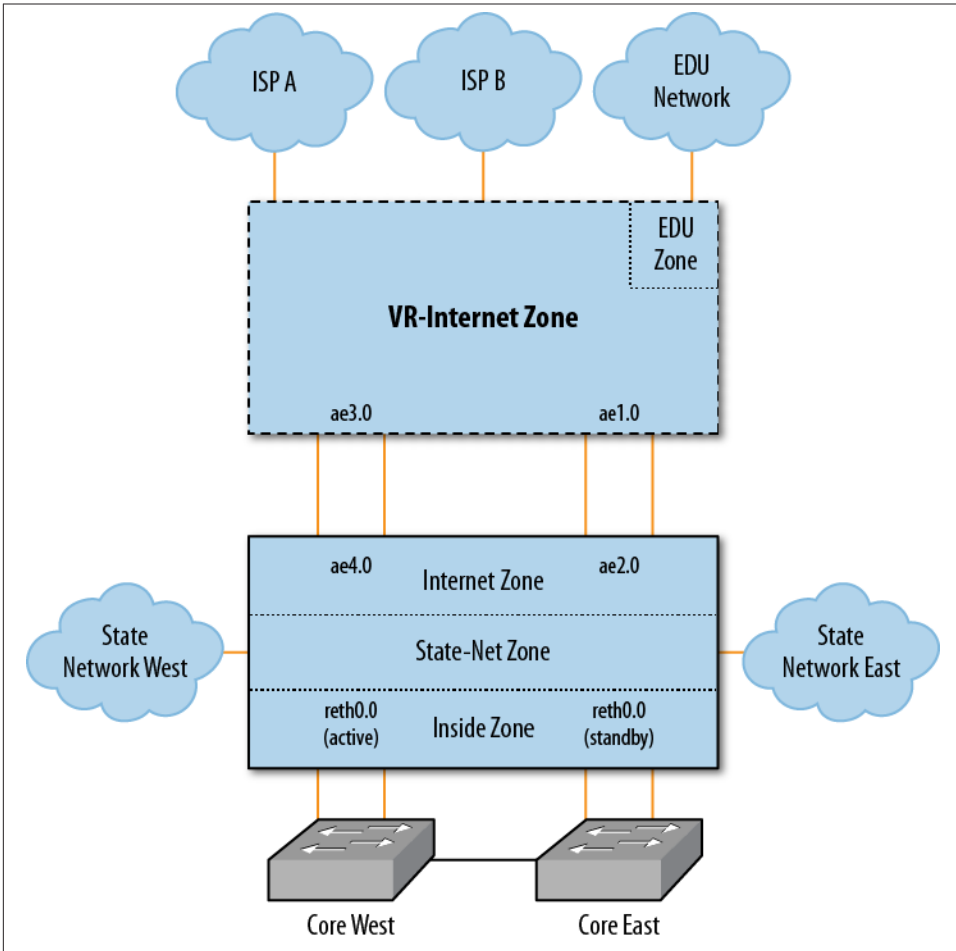


Figure 5-6. Firewall zones

The state uses private addressing for most components, although some public addressing is used in the state WAN networks and other areas. There are requirements for static NAT and source NAT. Here's an example of the rules for NAT:

```
security {
  net {
    source {
      pool 1 {
        address {
          1.1.1.2/32;
        }
      }
    }
    address-persistent;
  }
}
```


The *address-persistent* command assures that the same source address is used for full sessions for outgoing traffic. This was necessary to ensure that IPsec tunnels remained operational. The static NAT entries have a *proxy-arp* setting for the outgoing interfaces. Here's an example:

The biggest challenge of the NAT configuration was staying within the restrictions of the older versions of Junos. This engagement went operational with 10.1 code. In that code, the number of rules in a rule set and the number of rule sets were limited. For instance, the static NAT rules had to be broken into three 256-rule rule sets. Each rule set had a different context (virtual router, interface, or zone). In later versions of the Junos OS, these restrictions have been eliminated.

Security logging

Before we leave the security configuration, the security logging that was used should be reviewed. Initially, three syslog servers were to be used to gather security logs (remember the multiple administrators with responsibilities for portions of the egress firewall?). When the three log servers were installed and made operational, we found that the active routing engine on the SRX was being bogged down—the culprit was the logging. Two of the log servers were eliminated, and forwarding functions were added to the servers to accomplish the same results. Here's an example of the security logging configuration:

```
security {  
  log {  
    format sd-syslog;  
    source-address 10.1.1.2;  
    stream STRM {  
      severity info;  
      format sd-syslog;  
      category all;  
      host {  
        10.1.1.81;  
      }  
    }  
  }  
}
```

Routing

Like the security configuration, the routing configurations required a little imagination and service provider–type policies. The routing has three parts: the BGP coordination with the ISPs and the state WAN, the OSPF coordination with the core network, and the use of default routes. One of the requirements that caused a couple of changes to the configuration was the requirement for load sharing between the ISPs. Initially, we implemented equal cost multipath (ECM) load balancing on the virtual router. However, we found a code conflict between NAT and ECM that disabled the ECM load balancing. So, we went to plan B. To help you keep a clear picture of where each routing protocol was in use and the interrelationships between these protocols, [Figure 5-7](#) provides a diagram of these relationships.

BGP

The initial BGP deployment strategy was to only have BGP operational in the Internet VR. A default route would be sent to the default routing instance, and that would be redistributed to the OSPF protocol. ECM in the Internet VR would provide load balancing. As described, the initial plan was not operational. Plan B was to use multipath routing on the virtual router. This did not provide per-session load balancing, but rather per-prefix load sharing. The problem with this solution was that asymmetrical routing (out on ISP A and in on ISP B) would cause sessions to be dropped. This was not an

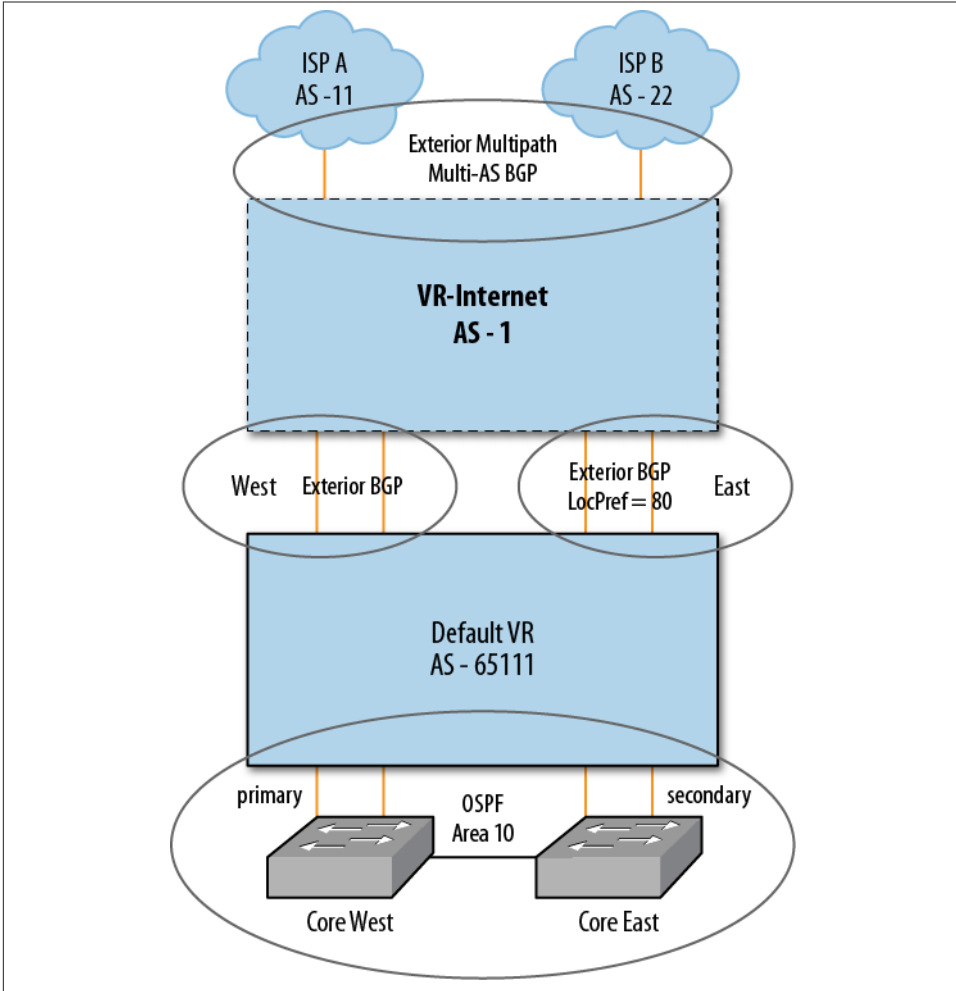


Figure 5-7. Egress routing plan

issue for incoming traffic, but for outgoing traffic, we needed to use a little sleight of hand. The source address pool for ISP A was different than the pool for ISP B. These routes were sweetened and soured accordingly so that return traffic preferred the ISP that originated the traffic. A diagram showing this relationship is presented in [Figure 5-8](#).

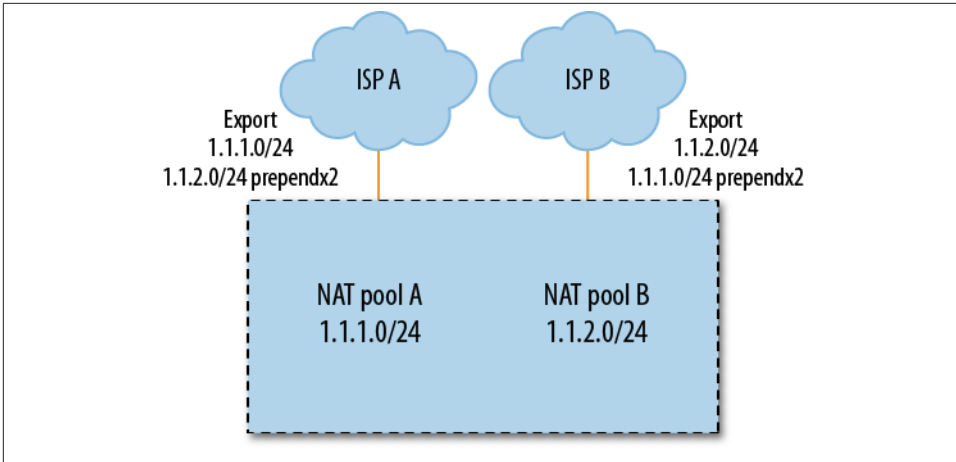


Figure 5-8. Symmetrical routing via selective NAT

The remainder of the routes were advertised equally to both ISPs. The complete BGP configuration for the ISPs was part of the routing instance configuration. For the display, the state's AS is given as 1 and the ISP's as 11 and 12. These obviously were not the registered AS numbers, but are just placeholders for this book. The configurations were:

```

routing-instances {
  VR_Internet {
    instance-type virtual-router;
    interface ge-10/1/0.0;
    interface ge-22/1/0.0;
    interface ae1.0;
    interface ae3.0;
    routing-options {
      static {
        route 1.1.1.0/24 reject;
        route 1.1.2.0/24 reject;
      }
      autonomous-system 1;
    }
  }
  protocols {
    bgp {
      log-updown;
      group to-isp {
        type external;
        multipath multiple-as;
        neighbor 2.2.2.33 {
          local-address 2.2.2.34;
          authentication-key "$9$7eVsgF/Yg/t";
          export isp_west_out;
          remove-private;
          peer-as 11;
        }
      }
    }
  }
}

```

```
neighbor 3.3.3.185 {  
    local-address 3.3.3.186;  
    authentication-key "$9$p/BEyvfhd fghcoG";  
    export isp_east_out;  
    remove-private;  
    peer-as 12;  
}  
  
}  
  
}  
  
}
```

A couple of notes on this configuration snippet:

Static routes

These were the NAT pools that were used on the firewall. The static routes found in the normal network were redistributed from the default routing instance.

multipath multiple-as

This was a great tool for load sharing—instead of load sharing between peerings that share a common router ID, it allowed sharing between routes that come from different ISPs.

remove-private

This was a protective command to prevent the inadvertent advertisement of the internal private AS number to the ISPs (see the next section).

The export policies that were called contain no surprises. Each side exported the full network routes (represented by 1.1.0.0/17 here) and a set of NAT routes. These were AS paths prepended with the local AS# to force the traffic back to the side that had assigned the NAT address. The export policies were:

```

policy-statement isp_west_out {
    term source_nat_west {
        from {
            route-filter 1.1.1.0/24 exact;
        }
        then accept;
    }
    term source_nat_east {
        from {
            route-filter 1.1.2.0/24 exact;
        }
        then {
            as-path-prepend "1 1";
            accept;
        }
    }
    term public_out {
        from {

```

```

        route-filter 1.1.0.0/17 exact;
        ...
    }
    then accept;
}
term deny_all_bgp {
    from protocol bgp;
    then reject;
}
}
policy-statement isp_east_out {
    term source_nat_north {
        from {
            route-filter 1.1.1.0/24 exact;
        }
        then {
            as-path-prepend "1 1";
            accept;
        }
    }
    term source_nat_south {
        from {
            route-filter 1.1.2.0/24 exact;
        }
        then accept;
    }
    term public_out {
        from {
            route-filter 1.1.0.0/17 exact;
            ...
        }
        then accept;
    }
    term deny_all_bgp {
        from protocol bgp;
        then reject;
    }
}
}

```

The second BGP arrangement was between the two routing instances. The VR-Internet and the “inside” routing instances were connected by a set of aggregate Ethernet links divided between east and west. This set of links was to be used in a primary/secondary relationship. All traffic would use one set of links all the time, and in the event of a link, interface, or device (content/URL filter server) failure, the traffic would divert to the standby link. To achieve this relationship, the east side routes were sourced with a lower route preference.

It is somewhat of a challenge to have both sides of a BGP peering on the same device. In this case, the two sides were separated by the routing instance. For the internal relationships, private AS (65111) numbers were used. This allowed an external BGP peering between the two sides of the same device. The configuration was:

```

protocols {
    bgp {
        log-updown;
        group internet-east {
            type external;
            import local-pref-80;
            neighbor 10.10.10.33 {
                local-address 10.10.10.34;
                export ospf-to-bgp;
                peer-as 1;
            }
        }
        group internet-west {
            type external;
            neighbor 10.10.10.17 {
                local-address 10.10.10.18;
                export ospf-to-bgp;
                peer-as 1;
            }
        }
    }
}

```

Again, these configurations are straightforward. The import policy for the east side was:

```

policy-options {
    policy-statement local-pref-80 {
        term local-pref {
            then {
                local-preference 80;
                accept;
            }
        }
    }
}

```

This forced all Internet-learned routes to use the west side aggregate Ethernet links as a primary route.

The export policies took the OSPF routes from the core, deleted the default route, and sent the remainder to the routing instance. While this was not necessary from a routing point of view, it allowed us to manipulate the return traffic. The export policy was:

```

policy-options {
    policy-statement ospf-to-bgp {
        term default {
            from {
                route-filter 0.0.0.0/0 exact;
            }
            then reject;
        }
        term ospf {
            from protocol ospf;
        }
    }
}

```

```

    }
    then accept;
}
}

```

The routing instance side of the configuration was very similar to the default routing instance, the only difference being the import and export policies:

```

routing-instance {
  VR_Internet {
    protocols {
      bgp {
        group inside-east {
          type external;
          import local-pref-80;
          multipath;
          neighbor 10.10.10.34 {
            local-address 10.10.10.33;
            export default_only;
            peer-as 65111;
          }
        }
        group inside-west {
          type external;
          multipath;
          neighbor 10.10.10.18 {
            local-address 10.10.10.17;
            export default_only;
            peer-as 65111;
          }
        }
      }
    }
  }
}

```

The default route received from the ISPs was advertised to the default routing instance (*export default_only*), and the routes received on the east side were given a lower preference (*import local-pref-80*) than the routes on the west side.



When I see examples of route preference manipulation, I often see the authors set the preferred side links to a route preference of 120 and the nonpreferred side to 80. I understand the math; what confuses me is why they adjust both sides. This doubles the work, and doubles the chances for an error. My mantra has always been KISS. Setting one side lower and leaving the other at the default reduces the chances of an error and is often easier to troubleshoot later when things go wrong.

OSPF

Compared to the BGP configuration, the OSPF configuration was a simple affair. The Reth interfaces provided the primary/secondary relationship, the core provided all the routes from the state's networks, and a redistribution policy blended BGP and OSPF together. The OSPF configuration was:

```
protocols {
  ospf {
    export bgp-to-ospf;
    reference-bandwidth 1g;
    area 0.0.0.10 {
      interface reth0.0
      interface lo0.0
    }
  }
}
```

The export policy for OSPF accepted the default route from BGP (originated by the ISPs) and added a metric of 1. The type 2 metric and the low cost assured that this route would be preferred over other default routes that were found floating around in the state's network:

```
policy-options {
  policy-statement bgp-to-ospf {
    term bgp {
      from {
        protocol bgp;
        route-filter 0.0.0.0/0 exact;
      }
    }
    then {
      external type 2
      metric 1;
      accept;
    }
  }
}
```



I'm constantly getting confused between type 1 and type 2 OSPF external routes. Type 1 external routes inherit the cost of the path, whereas type 2 external routes keep the same metric that was given to them. It's sort of like the first sons and second sons of old-timers: the first son inherits the estate and the second son gets whatever is left.

Default route

While the use of BGP and OSPF allowed the distribution of all the different routes in the network, the distribution of the default route simplified the routing in the network.

The default route (0/0) was received from both ISPs equally. The VR-Internet routing instance looked at the default route as an equal-cost route. The default route was the only route advertised downward (VR-Internet to default instance) within the firewall. In the default routing instance, the routes received on the east side were given a higher route preference than those received from the west peering. Also, although not shown in the configurations, the BGP protocol was given a lower route preference than external OSPF (145 versus 170). This setting assured that any default route received from OSPF would only be used by the firewall if both ISPs were lost. Finally, the BGP default route was redistributed into OSPF as a type 2 external route with a metric of 1. Again, the use of the type 2 external OSPF route with a very low metric (1) assured that this ISP-generated default route would be accepted and used in the state's network.

Often, dealing with default routes means a static mapping of the route, with its associated lack of failure recovery and route selection. Here, we used the dynamic nature of the routing protocols and route preferences to assure that traffic was handled correctly regardless of the failure scenario. Using [Figure 5-9](#) as a guide, the failure scenarios can be traced. If an ISP is lost, the remainder of the network is not affected. If the preferred west side link is lost, the east side route is present and takes over. If either chassis of the cluster fails, the other takes over. If one of the legs of the Reth fails, the other advertises the route. If one of the core switches fails, the other provides the route. All in all, it's a very survivable setup.

Out-of-band management network

Of all the networking elements that we had to design and install, the OOB network required the most thought, trials, and general angst. The root of the problem was a laundry list of issues that included the following:

- The management systems were not in an IP subnet that was separated from the general population of the state users. The NSM, syslog, administrative workstations, and NTP were not all in the same subnet.
- The chassis were physically in two different towns.
- There wasn't any equipment (routers, switches, links) designated solely for OOB.
- There are restrictions on what SRX services can use routing instances for outgoing traffic (for example, NTP does not work if the outgoing interface is in a routing instance).
- The same subnet cannot appear twice on the same router on different ports.
- In an SRX cluster, both chassis must talk to the NSM.

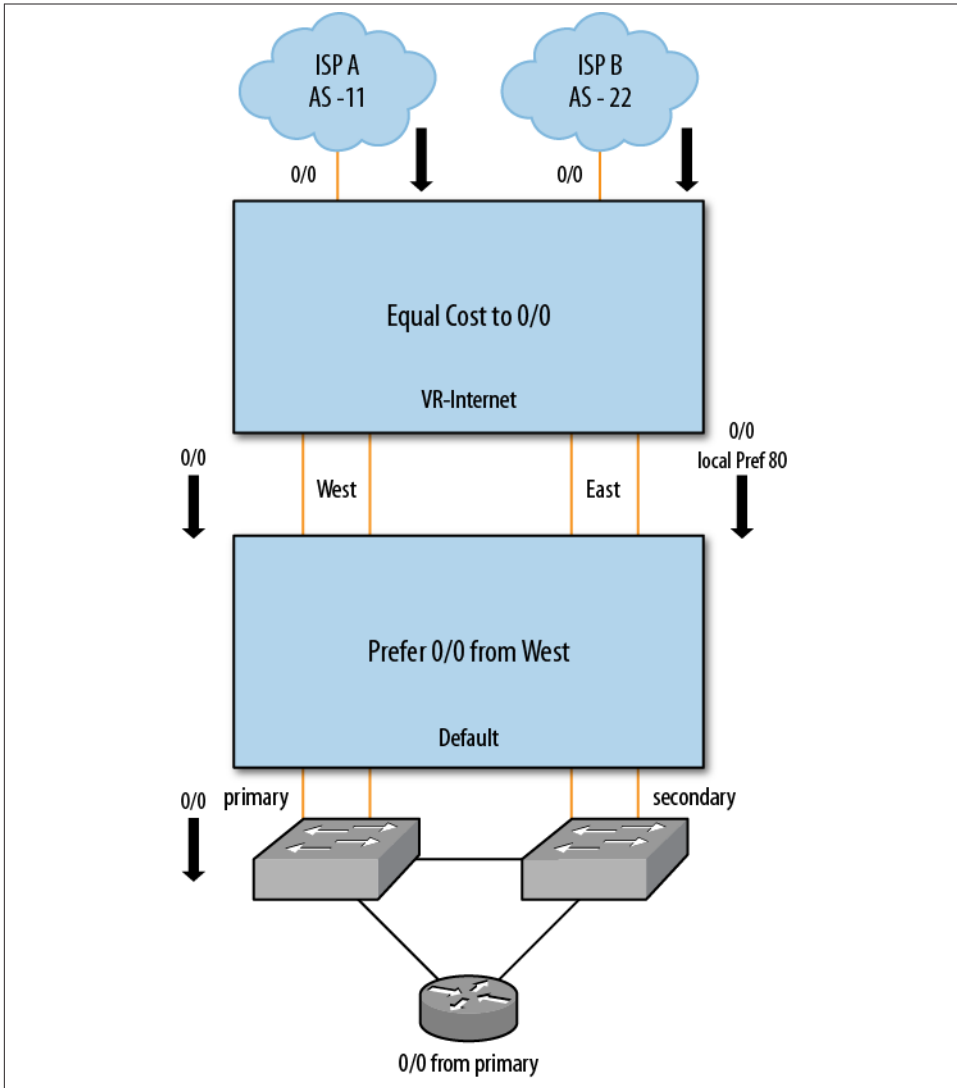


Figure 5-9. Default route treatment

We tried a combination of routing instances, node groups, and static routes to no avail. We ended up just setting the *fxp0* port addresses in the group configurations, and setting a retained static route to the management network (routes with the *retain* tag remain in the routing table of the standby chassis). The minimal configuration for the management system was:

```

routing-options {
    graceful-restart;
    static {
        route 10.10.10.96/29 {
            next-hop 10.10.10.105;
            retain;
        }
    }
}
interfaces {
    fxp0 {
        unit 0 {
            family inet {
                address 10.10.10.98/29;
            }
        }
    }
}

```

Implementation

This engagement has been presented as a completed project that, once configured, was stood up and made operational. Nothing could be further from the way the project actually deployed.

First, a little history will help. The state's network had been a complete Cisco network: routers, switches, firewalls, IDP appliances, and VPN concentrators. The intrusion of a Juniper Networks device into this design was met with skepticism, but also interest. Initially, the “new” firewalls (the SRX5800s) were to replace only the egress firewalls (reference [Figure 5-1](#)). Once the capabilities of the SRX had been explored, and it proved that it could handle the requirements of not only the egress firewall but also the other firewalls, the decision was made to expand the scope of the project.

The next initiation rite for the SRX was investigating the routing capabilities of the device. Once these had been demonstrated with a new ISP circuit and test Cisco routers, the scope was again expanded to include the IBR, the EDU router, and the WAN router. These swap-outs were not all performed at once, but in a number of iterations of configuration and maintenance windows.

The site-to-site IPSec VPNs were next to be migrated to the SRX. A number of tests and configurations were generated to validate that the capabilities of the existing VPN concentrators could be mimicked in the SRX. Once this battle was won, we again expanded the scope of the project and migrated the VPN users to the firewall.

The final migration was the retirement of the egress switches. These devices contained a labyrinth of VLANs and numerous L2 and L3 adjustments that were known to only a few of the state's networking engineers. The elimination of these devices also meant that a cleanup of the core routing had to be performed to assure that the routes offered by the firewall were used throughout the state's infrastructure.

All in all, these steps took over six months of determined effort from the entire tribe.



Prior to joining the professional services ranks, I was a Juniper instructor. The differences between the two occupations are just as striking as the similarities. Both allow techies to play with Juniper gear to their hearts' content. Both involve meeting great networking personnel from the world over. Both are learning experiences. The principal difference is the intensity level. For teaching, an instructor is “on” for the entire day, and it is very intense. At the end of the day, I would sit and vegetate. In contrast, professional service engagements are like a desk job. You test, configure, write up and present your findings, sit in meetings, and take notes. As is the case with many athletes, being a network warrior is 99% preparation and 1% total panic. The 1% is the cut-overs during maintenance windows where the other 99% of the job is put to the ultimate test... does it work?

Lessons Learned

There were many “1%” early mornings in this implementation. Rather than go through the gory details, here are the lessons learned from the implementation.

Feature interactions

The first lesson was that any device introduced into a network will result in interactions between features that have not been fully vetted prior to deployment. Note that all the issues identified below have since been resolved, and I do not believe that in the current code any of these remain a problem:

- Application layer gateways (ALGs) and intrusion detection and prevention initially caused the SRX to reset all the input/output cards in the chassis. We had to deactivate all ALGs to allow IDP to operate on the device.
- Egress filtering had limitations that did not allow its use for filter-based forwarding. All traffic filters had to be installed in the inbound direction on the interfaces (ingress filters); no egress filters were supported.

- Virtual private networks could not be installed from a virtual router. The private key exchange had to be originated from the default routing instance. This caused the IPsec VPNs to be sourced from the internal firewall instance rather than the Internet-facing firewall instance.
- Syslog and NTP could not be run from a virtual router routing instance. These management functions have to be run from the main routing instance. The source interface for the traffic must also reside in the default routing instance.
- Network address translation and equal cost multipath load balancing did not operate together on the same interfaces. This required us to rethink the load-sharing requirement for the customer. The use of BGP multipath prefix-based load balancing was the final implementation.

Network interactions

There are many things that can go wrong with the peering when working with an Internet service provider on a BGP link. In some cases the result is misrouted traffic, in others it is traffic that is lost (blackholed), and in some cases there is no outward response; the peering just is dropped. Here are a few of the issues that we experienced with the ISP BGP peerings:

- ISPs of the same size do not offer routes at the same “cost.” For BGP, the goodness of a prefix is based on a number of criteria. Each criterion is ranked. If two prefixes have the same ranking for the top criterion (is the next hop usable), the next-highest criterion (local preference) is used as a tiebreaker. If both next hops are good and the local preference is the same, then the AS path length is examined. This is the number of networks the prefix is away from the IBR.

Further, if both ISPs are the same size (number of customers) and cover the same geographic area (regional versus national or international), it would be expected that the AS path statements from each ISP for the same prefix would be equal. Not so here—one of the ISPs had a consistently shorter path to prefixes than the other, so we had to pad the AS path from that provider to compensate. This caused havoc with symmetrical routing and resulted in lost sessions.

- ISPs can accept but ignore routes with longer-than-expected subnet masks. This was an interesting problem to troubleshoot. In our outgoing prefix list, a longer (/26) route was included in the export policy by mistake. The ISP would only accept prefixes with a /24 or shorter mask. Traffic to this prefix was not getting to the users from the Internet. Using a looking glass, the /24 of the prefix was seen in the Internet with a path to our two ISPs. We were not trashing the traffic due to security policies

(the traffic did not show up in any security logs). We finally did a policy scrub and saw the long prefix. For some reason, the prefix was not rejected in BGP from the ISP, but accepted as an invalid route. We had to add terms in the export policies to guarantee that no long subnets were being advertised.



A **looking glass** is an Internet site that allows an administrator to see what is happening to a route in the Internet.

- ISPs can drop the peering upon receipt of an invalid AS path for a prefix. In this lesson, BGP provided no clue (the *traceoptions* file contained no BGP message showing the cause of the loss of peering). What would happen was that the peering session would go from *active* to *established* for a second or two, then back to *active*. Monitoring the status continuously using the *show bgp summary | match 1.1.1.1* command would show this blip of the peering being established. Scanning the advertised routes to this ISP showed that one of the routes received from the state's WAN network included a private AS path designator (e.g., *AS path: 34563 43721 65111 43122 I*). The inclusion of the *remove-private* command in the BGP peering stanza solved this issue.
- Not all of the issues were with the ISPs. OSPF interoperability between the SRX and the existing Cisco routers provided a number of lessons. The first lesson was that the default MTU of Cisco routers is not the same as that of Juniper routers. In our initial testing, the OSPF adjacencies would not come up. The logs showed an error in MTU negotiation, but both sides were Ethernet 100 Mbps FDX without an MTU set. Monitoring the traffic on the interface (*monitor traffic interface ge-0/0/0*) with detail and a length of 1,500 octets showed that the outgoing MTU was different than the incoming. Changing the local MTU to match that of the incoming side allowed the adjacencies to be completed.
- Another OSPF issue was the use of a 1 Gbps OSPF reference bandwidth to determine the OSPF cost for links. As soon as links above 100 Mbps (the default reference bandwidth) are used in a network, the reference bandwidth should be increased to 1 or 10 Gbps. Otherwise, all the LAG interfaces, 1G, and 10G interfaces will have the same cost as a 100 Mbps circuit (1). Once the decision is made to use the greater reference bandwidth, this change has to be made network-wide or the results are very unpredictable. It took us days of monitoring the OSPF database to find all the existing routers that need the reference bandwidth to be changed. OSPF database entries age very slowly, and it seems that rogue prefixes can appear and disappear by magic. This lesson was one of patience and attention to detail rather than the normal technical outcomes.

Administrative issues

Not all design issues were due to the operation of the hardware and software of the egress devices. Some of them were caused by the humans that operated and maintained the devices. Here are some of the lessons this warrior learned from this engagement (no, you cannot just hit them with a hammer, no matter how easily that would solve the issue!):

- Communicating with C-level officers can take a long time—often days or weeks. Our project was funded and manned by the chief networking officer (CNO) of the state. He reported to the chief information officer (CIO). The CIO also had a chief security officer (CSO) who reported to him. Those of you that have tried to work in a similar environment know where this is going already. The CNO was responsible for the project of providing security to the network. The CSO had the responsibility of defining what security means to the state. Any change in the egress security had to be approved by the CSO. We spent weeks waiting for approval for seemingly simple changes to the design. We redesigned some elements back and forth and back again. The final design attests to the mistrust between the two groups (IDP is in different appliances than the firewall).
- Nontechnical issues can waylay a project. The initial cut-over maintenance periods did not go smoothly. We experienced outages and had to get all hands on deck to resolve the issues. These initial events set a precedent in the state. Any time a maintenance window was requested, the objections came in from all corners. Some were frivolous (opening a new library that weekend), and some were serious (the elections are on Tuesday, or this is a pay week). For one of the phases of the project we had to postpone the cut-over four times, for a total of six weeks. It was a lesson in patience.
- Some users are more equal than others. I could see the benefit of keeping the governor's office happy, and maybe the highway patrol should be a priority since everyone had to drive back and forth to the site, but where does a state rep's access to Facebook fall into the general order of things? How about the... well, you know where this is going. We not only had technical issue to resolve, but we spent days chasing issues for the proverbial squeaky wheel.

Conclusion

Like it or not, facing all of these issues is part of being a network warrior, and this warrior would not wish for any changes. This is what makes it not a job, but a learning experience. Each and every day I learn something new—sometimes it's technical, sometimes social, sometimes personal. When this job stops teaching me, I'll retire, but I don't see that happening anytime soon.

Service Provider Engagement

It used to be that Juniper Networks was a service provider networking vendor. That's not the case any more, as this book makes evident, but the Junos OS was created for SPs, and the hardware was built for very large networks with lots of physical interface cards (PICs) and interfaces. In the early days, we warriors were SP guys. We lived in Network Ops. We got the right certifications, and the enterprise jobs were a means to fill the gaps between bigger service provider gigs.

Today, service provider warriors come in teams—maybe tribal nations, if we stay true to the warrior concept. And this is a necessity. From data centers to clouds to the large carriers, you need a total solution and you need a team of bright people with special skills in various areas. Service provider warriors are a special breed who can think in next hops of thought clumps to the final logical outcome, do algorithms in their heads, and speak using no less than a dozen acronyms per sentence.

For the following engagement, the tribe consisted of a seasoned professional services engineer (me), a talented Juniper Networks sales engineer, and a very bright engineer from the customer's office.

Company Profile

This engagement was an opportunity to assist an independent telephone company in upgrading its infrastructure. The company was installing Juniper MX edge routers and Juniper EX switches in support of its digital subscriber line (DSL) and voice customers. The task at hand was the deployment of two virtual private network technologies: virtual private LAN service (VPLS), and a Layer 3 virtual private network (L3VPN) overlaid on a border gateway protocol/multiprotocol label switching (BGP/MPLS) network.

The work environment of this service provider reminded me of my introduction to telephony many years ago, when I worked summers at the Champlain Telephone

Company. Working part-time while I was in college, I assisted line gangs, installers, and central office (CO) repairmen. This essentially meant trying to keep out of their way and learning as much as I could. How things have changed: from open-wire 16-party lines to Gigabit per second (Gbps) Ethernet in 30 short years.

Training the up-and-comers is good network warrior code. Try to take on some interns as you do your gigs and pass down the craft. I'm sure in books written 50 years from now, some of those interns grown into warriors will be recounting how quaint 100 GE was within data centers.

Anyway, the objectives for this engagement were multifold: my team and I were to provide boilerplate configurations for the MXs, VPLS, and L3VPN services; an out-of-band management network; redundancy; and directed path selection for the VPNs. They had chosen Juniper Networks for the simple reason that the hardware would deliver and scale, and now it was time to put it together.

Let's get cracking.

Physical Network Topology

The network layout shown in [Figure 6-1](#) has the remote locations of the MXs in central offices, and the EX switches are located in controlled environmental cabinets (CECs) or central offices. The digital subscriber line access multiplexers (DSLAMs) are located with the switches, MXs, CECs, or on customer premises. Access to the voice network and the Internet is provided from the central node of the network. All remote offices are connected by a bidirectional fiber ring as well as a direct fiber back to the central node.

The fiber ring supports 10 Gbps Ethernet traffic, while the direct fiber supports a pair of 1 Gigabit Ethernet connections. Each DSLAM is connected to the EXs by a pair of 1 Gigabit Ethernet connections. The total coverage of the 10 Gbps ring is between 100 and 200 miles. The optics deployed in the network are based on the distances spanned and the optical transmission equipment used. This design was based on a combination of geography, scaling, and survivability. The geography of the area limits the paths that fiber can economically take: the home runs to the central node follow the existing cable plant paths, while the ring follows an alternate path through valley and dale. The home run is the primary path for all the DSL and voice traffic, while the ring is the preferred path for the high-bandwidth traffic from the cell towers in the area. Combined, they offer a very reliable system.

The geographically close DSLAMs are connected to a single EX switch with redundant fiber Gigabit Ethernet links. The links are in a redundant trunk group (RTG) at the switch and operate in a primary secondary arrangement. The switches, in turn, are

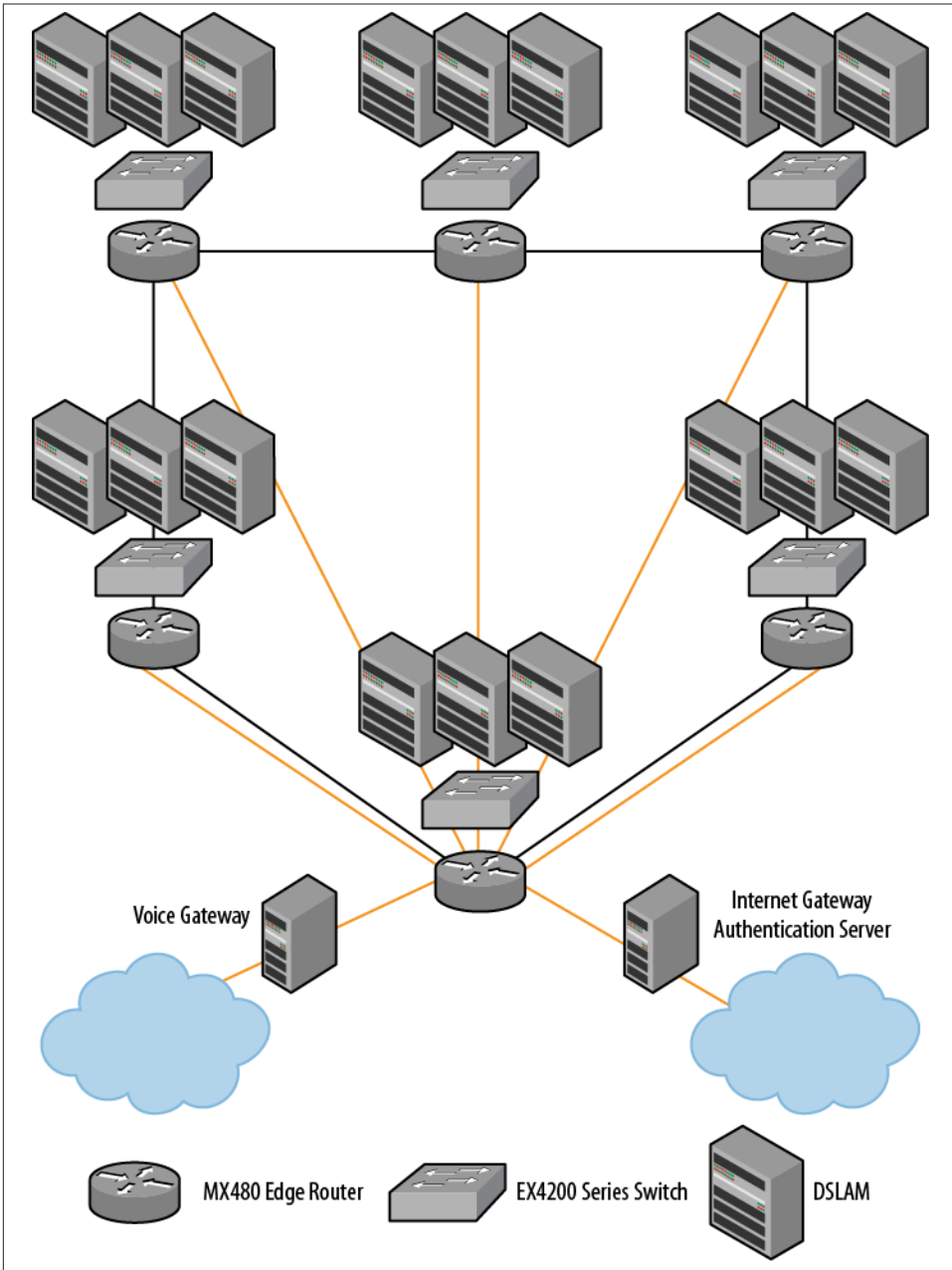


Figure 6-1. Service provider network topology

connected to the MXs in a similar manner, but this time aggregate Ethernet links are used rather than the redundant trunk groups. Each location is wired in a similar fashion, reducing the troubleshooting and configuration tasks. The use of RTGs in the EXs reduces the need for the spanning tree protocol (STP) and its inherent problems.



I have nothing against STP—I’ve used it in many applications—but when milliseconds count and a service provider’s primary services are concerned, I would much rather stick to a technology that is carrier class.

The aggregate links are used where RTGs are not supported. Employing the link aggregation control protocol (LACP) on the aggregate links allows additional channels to be added without dropping the entire circuit. Finally, as new locations are added to the topology, the 10 Gbps ring can be expanded while maintaining connectivity and redundancy.

Services

The services offered on the network included residential and commercial voice traffic, home and office data traffic, and the data feeds to cellular sites in the serving area. The teleco offered both copper-based and fiber-based DSL services in its serving area. Internet customers were offered rates up to 18 Mbps, dependent on the quality of the copper plant and the distance to the DSLAM.

Business customers and regional wireless providers were offered virtual private network services and virtual private line services over the network. The 10 Gbps ring was being constructed just for wireless backhaul services for Verizon and AT&T, who had cell towers in the area.

The teleco was also replacing the classical circuit-switched class 5 central office voice switch with a Voice over IP (VoIP) switching system. This allowed the voice traffic to ride on the same backbone as the data traffic.

Design Approach

The design approach for this project was actually created by the Juniper Networks sales team, prototyped, and then demonstrated in the Juniper labs. My team and I received the set of configurations from the demonstrations and created the working design. Our design had a number of guidelines common to most service provider build-outs these days:

KISS (Keep It Super Simple)

The design had to be straightforward. The majority of the technicians were telephone-oriented; “IP” is often used in a derogatory manner. Our deployment had to be easy to understand and uncomplicated. “Keep it simple, stupid” is another alternative.

Make it carrier class

Redundancy and survivability were key requirements. A single fiber cut or hardware malfunction cannot bring down the system.

Reduce, reuse, and recycle

The design had to be able to handle all the services and reuse existing links and equipment to the greatest extent possible. This went for the configurations as well as the hardware; ideally, the configurations for the remote sites would be almost interchangeable.

Keep it transparent

The design should not affect the end customers.

In keeping with these design guides, a layered approach was employed.

MX connectivity

The OSPF routing protocol was currently being used in the network, so reuse of this protocol in the backbone made sense. Stacked on top of this routing protocol is BGP in its interior manifestation. This allows the support of Layer 3 VPNs as well as VPLS services. It also allows the filtering that is necessary to associate paths with traffic types.

On top of these protocols, or rather below them in the protocol stack, is MPLS. This protocol supports the transport of both Ethernet and private IP traffic over a common infrastructure. Supplying the signaling for the MPLS paths is the resource reservation protocol (RSVP). This wonderfully flexible signaling protocol supports a number of path selection options for the network's traffic.

Of course, these protocols are only useful when physical connections are created between the MXs. The connectivity was provided by 10 Gbps fiber links deployed in a series of point-to-point arrangements that created a ring configuration. This gave each MX two connections to all other MXs, one east and one west. The 10 Gbps links were reserved for wireless carriers and other large bandwidth users and as a backup for the primary links. The primary links were a pair of 1 Gbps fibers combined into a link-aggregated path. The resulting connectivity looks like a peace sign gone haywire, a hub and spoke design also connected by a ring.

To provide the carrier-class reliability, the MXs were deployed with redundant routing engines and redundant power supplies. Interface pairs were terminated on one of two

interface cards. This arrangement covered most single points of failure for the MX hardware. As an added security, the power supplies were direct current (DC) and were fed by the CO's -48 VDC battery plant with generator backup. All in all, a very survivable package.

EX connectivity

The EXs were connected to the DSLAMs and the MXs with 1 Gbps fiber links. The links between the DSLAM and the switch operated in a primary/secondary arrangement without the use of the spanning tree protocol. The switch used redundant trunk group (RTG) technology that provides a 100 ms failover time between the two links, without the need of running any protocol on the DSLAM. As far as the DSLAM is concerned, both links are active and only one responds to traffic.



Traffic on the secondary links is received by the card and dropped. Layer 2 control traffic is actually active on the link—for instance, link layer discovery protocol (LLDP) traffic will still be active and can be used for management purposes.

The dual links between the EX and the MX were aggregated into a single 2 Gbps link.

Each EX was deployed in a two-switch virtual chassis. The redundant links (RTGs or LAGs) were terminated on different chassis, providing the hardware failure resiliency that was needed. Like the MXs, the EXs were powered by the CO's -48 VDC power supply.

In some cases, multiple EX virtual chassis were interconnected to form a larger virtual stack. This improved survivability for the traffic between the DSLAMs and the EXs.

Deployment

The last point to be made about this service provider design has to do with the initial deployment plans. When we came on board with this project, we were informed that the configurations were to be created and put in place in the devices and that the connectivity would be provided on the night of the cut-over. All testing and troubleshooting would be performed at that time. We had a short discussion about this topic, and considering that the second of the MXs was being delivered the day we arrived, we arranged for that device to be delivered to the hub site for a staging period prior to the actual field deployment.

This test period allowed us time to try different approaches and get the bugs worked out prior to the cut-over. Each of the other nodes was staged in the same manner. This reduced the cut-over time for each node and also allowed us to find any equipment anomalies.

On the nights of the cut-overs, the 2 Gbps links were moved to the MXs, the links were moved on the DSLAMs, and traffic was back online in short order.

Management network

The management of the devices was performed by an out-of-band management network. The *fxp0* and *vme0* ports of the remote Juniper devices were connected to an Ethernet-to-T1 converter. At the hub location, all of the T1s were terminated and the Ethernet links were connected to a router. This provided connectivity to the support staff and the management tools that were used to monitor the links (for example, the multirouter traffic grapher, or MRTG) and provide log monitoring. The T1s were carried on different cable paths than the backbone connections, providing another layer of survivability to the system. The MXs support hardwired alarms, but these were not yet deployed in this system.

Design Trade-Offs

A number of trade-offs were discussed prior to the actual design being deployed in this warrior engagement. Some of these were actual “sit down and talk about it” trade-offs, and some were “because that’s the way we want it” discussions.

OSPF

Using OSPF as the interior gateway protocol (IGP) was not really a trade-off; the teleco already deployed OSPF in its customer-facing network, so the protocol was well known. Rather, the big question was whether to integrate the OSPF domain of the customer network with the backbone. Having one contiguous network would provide another level of survivability, but would also bring in the risk of routing loops and nonoptimal traffic.

The decision was made to keep the two domains separate, but to design them so that they could be interconnected at a later time without conflicts. The reason was “service provider simplicity”—this is something we warriors run into a lot, and sometimes the bigger the SP, the more entrenched the rule: the old network worked, the system was operational, and the client was happy. Besides, the new backbone could be deployed with no effect on the other network.

To allow the future combination of the domains, the new backbone was given area 101 to the existing domain’s area 0. If and when these two domains are interconnected, neither will have to be renumbered.

Because the links were all greater than 1 Gbps, the reference bandwidth for the metric calculation was set to 100 Gbps, allowing for growth of the 10 Gbps links to aggregates.



The reference bandwidth is used to calculate the OSPF cost of an interface. In this SP network, the links between the nodes are 2 Gbps. The reference bandwidth of 100 Gbps/2 Gbps gives a cost of 50 for these links. If the reference bandwidth were reduced to 10 Gbps, the cost would be 5 for the same links.

On the MXs, all the interconnecting links were OSPF links forming adjacencies to the other MXs. The loopback interfaces for each MX were also included in the OSPF configurations.

VPLS

Juniper supports two types of VPLS implementations, LDP-based and BGP-based, with the trade-off lying between simplicity and capabilities.

The deployment of an LDP/VPLS system is very much plug-and-play: just activate LDP between all the MXs and install the VPLS instances. However, the LDP/VPLS system has some scalability limitations.

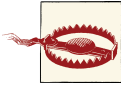
BGP/VPLS deployments, on the other hand, have the added complexity of requiring the deployment of BGP, but have an autodetect mechanism that provides for improved scalability. Juniper's website has a [great white paper](#) that discusses all the pros and cons of each choice.

Since the implementation plans also called for later deployment of a Layer 3 VPN on the same network, the decision was made to go with the BGP/VPLS implementation. (All the configurations will be provided later in the chapter.)

BGP

BGP can be deployed in an internal or an external manner. For this installation, the internal type was chosen. The reason was very simple: survivability. Internal BGP supports multihop connections by default. This allowed adjacencies to be formed between the loopback interfaces, thus allowing any interface to be used for transport of the BGP messages. As long as connectivity remains between the nodes, BGP will continue to operate.

For internal arrangements, BGP can be deployed in a full mesh or a star (route reflector) topology. The trade-off here is one of configuration hassle (full mesh) versus network complexity (route reflectors). Due to the limited number of sites in this network, the benefits of the use of a full mesh outweighed the reduced configuration required for the star topology. Another consideration was that when VPNs are added to a route reflector BGP deployment, the chances of connectivity issues increase.



When a route reflector BGP deployment is used with MPLS, in order for the VPN addressing to be active, the route reflector must have a default static route or an MPLS path to all other nodes. In either case, this is another thing to troubleshoot and remember when things are going south.

BGP is configured to handle not only IPv4 address prefixes, but also the next-layer reachability information (NLRI) for VPLS and L3VPNs. This same protocol can also be used to signal IPv6 prefixes when that is required.

MPLS

MPLS, as its name implies, provides support for multiple protocols over a label-switched path (LSP). The path creation is where the trade-off occurs, as two protocols can be used: LDP and RSVP.

LDP provides paths based on the OSPF shortest path between any two devices. When a path fails, LDP calculates the next-shortest route and reestablishes the path.

RSVP can set up paths on a similar basis (shortest path) and provide the same failover capability, but RSVP also allows the establishment of many paths between nodes and provides a primary/secondary arrangement between these paths. In the event of a failure of the primary path, the secondary immediately takes over and provides transport. Another advantage of RSVP is the capability of defining specific routes through the network for specific paths. In our teleco's case, they wanted the normal customer traffic to follow the 2 Gbps links as primaries and use the 10 Gbps links only if the primaries failed—easy enough to accomplish with either protocol. However, there was a second requirement that skewed the choice to RSVP: wireless carrier transport should only use the 10 Gbps links, and never the 2 Gbps links. Since the wireless requirement was a future requirement, we needed to plan for it today even if we would only build it out at a later time. This is another of the warrior's creeds (or maybe it's the Boy Scout motto): “Be prepared...” for the future.

Trade-off choices

This series of trade-offs resulted in decisions to use:

- OSPF in a non-backbone-area independent domain
- BGP/VPLS offering autodetection and scalability
- Internal BGP in a full mesh carrying IPv4, VPLS, and L3VPN NLRIs
- RSVP/MPLS with named paths for redundancy and control

Other warriors reading this book will know that these decisions were not made at the same time, nor on the first day, but revealed themselves over the first week of the engagement as the design objectives and limitations were tested and developed.

The rest of this chapter examines each of these decisions, along with the corresponding configuration examples.

Configurations

Let's get down to the "how" portion of this engagement. The configurations were performed using the command line, on devices running Junos version 11.1r1.

The initial design was developed with just a pair of MXs connecting two EX VCs and a pair of DLSAMs, as shown in [Figure 6-2](#). The connectivity was all fiber connections to the small form-factor pluggables (SFPs) that would be used in the field. The addressing was designed to include all the other devices and links. The use of private addresses for all internal networking provided a level of security and freedom for adding address space. The voice system used private addressing as well, while all DSL customers were assigned public IP addresses for Internet access. The remote equipment in this topology was designated *Node1* and the central equipment group was designated *Node0*. The actual node names were the town names that housed the MXs, but for security, these designations will do.

(This topology also allowed me to fill in any holes in the configuration examples using my test bed J2320s instead of MXs, which the publisher didn't even consider purchasing for me.)

Boilerplate Configuration

The MXs and EXs arrived with only default configurations. The following management and reliability configurations were added:

- A syslog server, to maintain a copy of the messages stored on the device.
- An SNMP polling server, to allow the MRTG server to gather stats.
- Synchronized commits, to simplify the configuration of the MX.
- Failover between routing engines (REs), to increase the survivability of the MX.
- Out-of-band management, to allow for major outages and troubleshooting via the *fxp0* ports. The group configuration allowed a single address per *fxp0* port and a common address for the node. The backup router allowed the backup routing engine to talk without the routing process running.

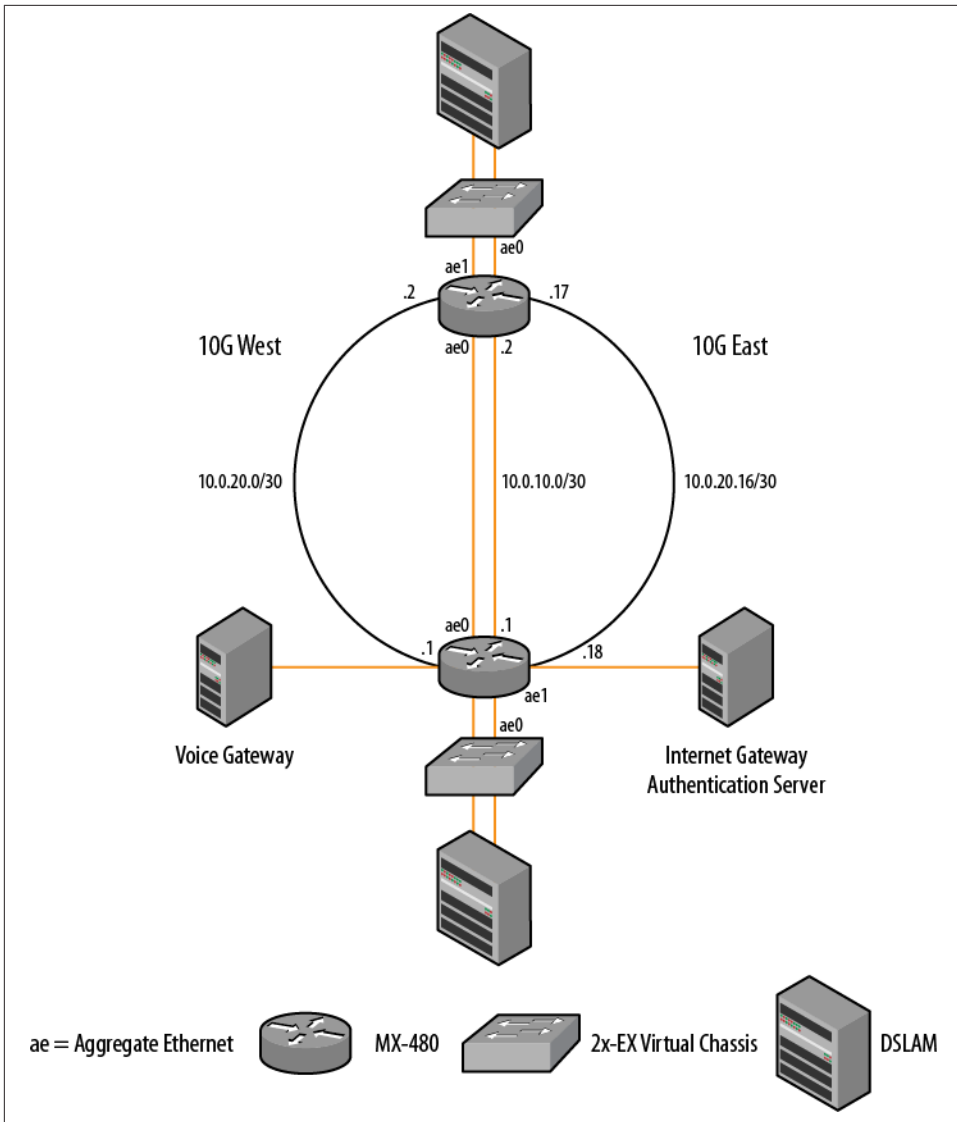


Figure 6-2. Design configuration

The base configuration for the MXs was:

```
group {
  re0 {
    system {
      host-name Node0-re0;
      backup-router 10.0.40.1 destination 0.0.0.0/0;
    }
  }
}
```

```

    interfaces {
        fxp0 {
            description "This is the fxp0 interface for re0";
            unit 0 {
                family inet {
                    address 10.0.40.21/24;
                    address 10.0.40.20/24 {
                        master-only;
                    }
                }
            }
        }
    }
}
re1 {
    system {
        host-name Node0-re1;
        backup-router 10.0.40.1 destination 0.0.0.0/0;
    }
    interfaces {
        fxp0 {
            description "This is the fxp0 interface for re1";
            unit 0 {
                family inet {
                    address 10.0.40.22/24;
                    address 10.0.40.20/24 {
                        master-only;
                    }
                }
            }
        }
    }
}
}
apply-groups [re0 re1]
system {
    root-authentication {
        encrypted-password "$1$SKE9aKDCWmq8SU7QEBDULyN.";
    }
    login {
        user lab {
            full-name Peter;
            uid 2000;
            class super-user;
            authentication {
                encrypted-password "$1$b$3s4K1o9GFten0";
            }
        }
    }
}
services {
    ssh;
}

```

```

syslog {
    user * {
        any emergency;
    }
    file messages {
        any notice;
        authorization info;
    }
    host 10.0.40.100 {
        any notice;
    }
}
commit synchronize;
}
chassis {
    redundancy {
        routing-engine 0 master;
        routing-engine 1 backup;
        failover {
            on-loss-of-keepalives;
            on-disk-failure;
        }
        graceful-switchover;
    }
}
snmp {
    description "Node0 MX480";
    location "Teleco Hub Location - top MX 01 Rack";
    contact "NetOP 800-555-9994";
    community Juniper-mx-ex {
        authorization read-only;
    }
}

```

MX Interfaces

Two types of interfaces are required for this installation. The first are the basic Gigabit interfaces and 10 GE interfaces that are supported on the MX. The second are the aggregate interfaces that link the devices together. The remote site has four 1 GE interfaces (two tied to the EX and two tied to the host MX) and two 10 GE interfaces (the east and west ring legs). The 1 GE interfaces are bundled into two separate aggregate interfaces, while the 10 GE links are used as separate interfaces. The configuration for the remote node is:

```

chassis {
    aggregated-devices {
        ethernet {
            device-count 3;
        }
    }
}

```

```

interfaces {
    xe-0/0/0 {
        description "10Gbps link west";
        unit 0 {
            family inet {
                address 10.0.20.2/30;
            }
        }
    }
    ge-0/2/0 {
        gigether-options {
            802.3ad ae0;
        }
    }
    ge-0/3/9 {
        gigether-options {
            802.3ad ae1;
        }
    }
    xe-1/0/0 {
        description "10 Gbps link east";
        unit 0 {
            family inet {
                address 10.0.20.17/30;
            }
        }
    }
    ge-1/2/0 {
        gigether-options {
            802.3ad ae0;
        }
    }
    ge-1/3/9 {
        gigether-options {
            802.3ad ae1;
        }
    }
    ae0 {
        description "link to Node0";
        aggregated-ether-options {
            minimum-links 1;
            link-speed 1g;
            lacp {
                active;
            }
        }
        unit 0 {
            family inet {
                address 10.0.10.2/30;
            }
        }
    }
}

```

```

    ae1 {
        description "Link to the EX";
        aggregated-ether-options {
            minimum-links 1;
            link-speed 1g;
            lacp {
                active;
            }
        }
    }
}

```



The chassis stanza is needed to allow aggregate interfaces to operate.

The *ae1* is only a placeholder at this point. Its configuration will be shown as part of the VPLS configuration. Also note that each leg of the aggregate interface is on a different flexible PIC concentrator. This provides a level of survivability in the case of an interface card failure. The aggregate interfaces are using LACP to allow interfaces to be added dynamically in the future. Also, the one-link minimum allows the interface to be maintained even if one of the links fails.

The interface configuration for the host MX (*Node0*) is very similar, with a few additions. The number of aggregate interfaces is increased to account for the other remote nodes, and the links to the voice and Internet access devices are added for external connectivity:

```

chassis {
    aggregated-devices {
        ethernet {
            device-count 10;
        }
    }
}
interfaces {
    xe-0/0/0 {
        description "10G link west";
        unit 0 {
            family inet {
                address 10.0.20.1/30;
            }
        }
    }
    ge-0/2/0 {
        gigether-options {
            802.3ad ae0;
        }
    }
    ge-0/2/1 {
        gigether-options {

```



```

        802.3ad ae3;
    }
}
ge-0/2/2 {
    gigether-options {
        802.3ad ae4;
    }
}
ge-0/2/3 {
    gigether-options {
        802.3ad ae5;
    }
}
ge-0/3/4 {
    gigether-options {
        802.3ad ae1;
    }
}
ge-0/3/5 {
    gigether-options {
        802.3ad ae1;
    }
}
ge-0/3/6 {
    gigether-options {
        802.3ad ae1;
    }
}
ge-0/3/8 {
    gigether-options {
        802.3ad ae6;
    }
}
ge-0/3/9 {
    gigether-options {
        802.3ad ae2;
    }
}
xe-1/0/0 {
    description "10G link east";
    unit 0 {
        family inet {
            address 10.0.20.18/30;
        }
    }
}
ge-1/2/0 {
    gigether-options {
        802.3ad ae0;
    }
}
ge-1/2/1 {

```

```

        gigaether-options {
            802.3ad ae3;
        }
    }
    ge-1/2/2 {
        gigaether-options {
            802.3ad ae4;
        }
    }
    ge-1/2/3 {
        gigaether-options {
            802.3ad ae5;
        }
    }
    ge-1/3/4 {
        gigaether-options {
            802.3ad ae1;
        }
    }
    ge-1/3/5 {
        gigaether-options {
            802.3ad ae1;
        }
    }
    ge-1/3/6 {
        gigaether-options {
            802.3ad ae1;
        }
    }
    ge-1/3/8 {
        gigaether-options {
            802.3ad ae6;
        }
    }
    ge-1/3/9 {
        gigaether-options {
            802.3ad ae2;
        }
    }
    ae0 {
        description "Link to Node1 MX-480";
        aggregated-ether-options {
            minimum-links 1;
            link-speed 1g;
            lacp {
                active;
            }
        }
        unit 0 {
            family inet {
                address 10.0.10.1/30;
            }
        }
    }

```

```

    }
}
ae1 {
    description "Link to DSLAM EX4200";
    aggregated-ether-options {
        minimum-links 1;
        link-speed 1g;
        lacp {
            active;
        }
    }
}
ae2 {
    description "Link to Internet Server";
    aggregated-ether-options {
        minimum-links 1;
        link-speed 1g;
        lacp {
            active;
        }
    }
}
ae3 {
    description "Link to Node2 MX-480";
    aggregated-ether-options {
        minimum-links 1;
        link-speed 1g;
        lacp {
            active;
        }
    }
    unit 0 {
        family inet {
            address 10.0.10.5/30;
        }
    }
}
ae4 {
    description "Link to Node3 MX-480";
    aggregated-ether-options {
        minimum-links 1;
        link-speed 1g;
        lacp {
            active;
        }
    }
    unit 0 {
        family inet {
            address 10.0.10.9/30;
        }
    }
}
ae5 {
    description "Link to Node4 MX-480";

```

```

        aggregated-ether-options {
            minimum-links 1;
            link-speed 1g;
            lacp {
                active;
            }
        }
        unit 0 {
            family inet {
                address 10.0.10.13/30;
            }
            family mpls;
        }
    }
    ae6 {
        description "Link to Voice EX4200";
        aggregated-ether-options {
            minimum-links 1;
            link-speed 1g;
            lacp {
                active;
            }
        }
        unit 0 {
            family inet {
                address 10.0.10.21/30;
            }
        }
    }
}

```

Once these interfaces were added and the commits were performed on both sides, a series of pings verified that the interfaces were up and operational. In addition, pulling some interfaces on either side showed that the LAGs were working as expected.

EX Boilerplate and Interfaces

Configuring the EXs was a basic cookie-cutter operation, with the only differences in the different locations' configurations being the VLANs supported on each site. Only one configuration is offered here, and that is the *Node1* site configuration. The switches are only operating at Layer 2, and the only IP information is the management interface:

```

system {
    host-name Node1-EX4200;
    root-authentication {
        encrypted-password "$1$S8aSozMg$paAE8W0Rkzi6P.0";
    }
    login {
        user lab {
            full-name Peter;
            uid 2000;
        }
    }
}

```

```

        class super-user;
        authentication {
            encrypted-password "$1$q3hlxptT3vcsgri/0";
        }
    }
}
services {
    ssh;
}
}
syslog {
    user * {
        any emergency;
    }
    file messages {
        any notice;
        authorization info;
    }
    host 10.0.40.100 {
        any notice;
        authorization info;
    }

    file interactive-commands {
        interactive-commands any;
    }
}
commit synchronize;
chassis {
    redundancy {
        graceful-switchover;
    }
    aggregated-devices {
        ethernet {
            device-count 10;
        }
    }
}
interfaces {
    ge-0/0/0 {
        unit 0 {
            family ethernet-switching {
                port-mode trunk;
                vlan {
                    members [ Data72 Data496 voice114 voice124 ];
                }
            }
        }
    }
    ge-0/0/1 {
        unit 0 {
            family ethernet-switching {

```

```

        port-mode trunk;
        vlan {
            members [ Data72 Data496 voice230 ];
        }
    }
}
ge-0/0/23 {
    ether-options {
        802.3ad ae0;
    }
}
ge-1/0/0 {
    unit 0 {
        family ethernet-switching {
            port-mode trunk;
            vlan {
                members [ Data72 Data496 voice114 voice124 ];
            }
        }
    }
}
ge-1/0/1 {
    unit 0 {
        family ethernet-switching {
            port-mode trunk;
            vlan {
                members [ Data72 Data496 voice230 ];
            }
        }
    }
}
ge-1/0/23 {
    ether-options {
        802.3ad ae0;
    }
}
ae0 {
    description "Link to MX-480";
    aggregated-ether-options {
        minimum-links 1;
        link-speed 1g;
        lacp {
            active;
        }
    }
    unit 0 {
        family ethernet-switching {
            port-mode trunk;
            vlan {
                members all;
            }
        }
    }
}

```

```

    }
  }
}
vme {
  unit 0 {
    family inet {
      address 10.0.40.30/24;
    }
  }
}
}
snmp {
  description "Node1 EX4200";
  location "Teleco Node1 - top MX 01 Rack";
  contact "NetOP 800-555-9994";
  community Juniper-mx-ex {
    authorization read-only;
  }
}
protocols {
  igmp-snooping {
    vlan all;
  }
  inactive: rstp;
  lldp {
    interface all;
  }
  lldp-med {
    interface all;
  }
}
}
ethernet-switching-options {
  unknown-unicast-forwarding {
    vlan Data496 {
      interface ae0.0;
    }
    vlan Data72 {
      interface ae0.0;
    }
  }
  redundant-trunk-group {
    group DSLAM1 {
      preempt-cutover-timer 3;
      interface ge-0/0/0.0 {
        primary;
      }
      interface ge-1/0/0.0;
    }
    group DSLAM2 {
      preempt-cutover-timer 3;
      interface ge-0/0/1.0 {
        primary;
      }
    }
  }
}

```

```

    }
    interface ge-1/0/1.0;
  }
  group DSLAM3 {
    preempt-cutover-timer 3;
    interface ge-0/0/2.0 {
      primary;
    }
    interface ge-1/0/2.0;
  }
}
storm-control {
  interface ge-0/0/0.0 {
    bandwidth 20000;
  }
  interface ge-0/0/1.0 {
    bandwidth 20000;
  }
  interface ge-1/0/0.0 {
    bandwidth 20000;
  }
  interface ge-1/0/1.0 {
    bandwidth 20000;
  }
}
}
vpls {
  Data425 {
    vlan-id 425;
  }
  Data434 {
    vlan-id 434;
  }
  Data453 {
    vlan-id 453;
  }
  Data486 {
    vlan-id 486;
  }
  Data492 {
    vlan-id 492;
  }
  Data496 {
    vlan-id 496;
  }
  Data545 {
    vlan-id 545;
  }
  Data70 {
    vlan-id 70;
  }
  Data71 {

```



```

        vlan-id 71;
    }
    Data72 {
        vlan-id 72;
    }
    Data73 {
        vlan-id 73;
    }
    Data74 {
        vlan-id 74;
    }
    Data75 {
        vlan-id 75;
    }
    voice110 {
        vlan-id 110;
    }
    voice114 {
        vlan-id 114;
    }
    voice120 {
        vlan-id 120;
    }
    voice124 {
        vlan-id 124;
    }
    voice140 {
        vlan-id 140;
    }
    voice230 {
        vlan-id 230;
    }
}
virtual-chassis {
    member 0 {
        mastership-priority 255;
    }
    member 1 {
        mastership-priority 254;
    }
    no-split-detection {
    }
}

```

The switches are the source for all the VPLS and L3VPN traffic. Each location supports unique VLANs that are carried over the backbone in a unique VPN. To simplify the management, all the VLANs are configured on each switch as part of the base configuration, as is the aggregate link to the MX. The site-specific information includes:

- The management interface address
- The DSLAM VLANs

- The hostname
- The SNMP location

There are a few items that should be explained in this configuration. The *storm-control* and *unknown-unicast-forwarding* stanzas limit the number of broadcasts allowed and determine where unknown traffic is sent, respectively. These commands limit the amount of damage that a single subscriber can cause on the network. The redundant trunk groups are defined with a 3-second preempt timer to allow the reversion to the primary once it's back online. If the links are flapping, this timer can be increased to stabilize the interfaces.

OSPF

The configuration of the IGP is simple once the interfaces are defined and the topology is known. In this design, two OSPF domains are used: the backbone IGP and the internal network IGP. These two are not interconnected at this point, and the internal OSPF configuration is shown as part of the L3VPN. The backbone OSPF configuration is only found on the MXs and for the most part is identical in all the MXs. It looks like this:

```

interfaces {
  lo0 {
    unit 0 {
      family inet {
        address 10.0.10.254/32;
      }
    }
  }
}
protocols {
  ospf {
    reference-bandwidth 100g;
    area 0.0.0.101 {
      interface lo0.0;
      interface ae0.0;
      interface ae3.0;
      interface ae4.0;
      interface ae5.0;
      interface xe-0/0/0.0 {
      }
      interface xe-1/0/0.0 {
      }
    }
  }
}
routing_options {
  max-interface-supported 20;
  router-id 10.0.10.254;
}

```

There are no surprises here—OSPF is running on all the backbone interfaces (aggregate and 10 GE) and includes the loopback interface. The loopback interface is used as the router ID for the OSPF databases. (It was added here because there was no reason for it prior to now.)

The routing options stanza is new to the configuration; it is required for certain hardware platforms (MX and SRX). The number of interfaces is the number of OSPF interfaces supported on the device.

A mirror configuration is entered in *Node1*, and once the configurations are committed the OSPF adjacencies come up and the OSPF database and the route table are populated. Here are the commands used to verify these steps:

```
lab@Node0-MX480> show ospf interface
```

Intf	State	Area	DR ID	BDR ID	Nbrs
ae0.0	DR	0.0.0.101	10.0.10.254	10.0.10.253	1
xe-0/0/0.0	DR	0.0.0.101	10.0.10.254	10.0.10.253	1
xe-1/0/0.0	DR	0.0.0.101	10.0.10.254	10.0.10.253	1
lo0.0	DR	0.0.0.101	0.0.0.0	0.0.0.0	0
ae3.0	Down	0.0.0.0	0.0.0.0	0.0.0.0	0
ae4.0	Down	0.0.0.0	0.0.0.0	0.0.0.0	0
ae5.0	Down	0.0.0.0	0.0.0.0	0.0.0.0	0

```
lab@Node0-MX480> show ospf neighbor
```

Address	Intf	State	ID	Pri	Dead
10.0.10.2	ae0.0	Full	10.0.10.253	128	36
10.0.20.2	xe-0/0/0.0	Full	10.0.10.253	128	38
10.0.20.17	xe-1/0/0.0	Full	10.0.10.253	128	33

```
lab@Node0-MX480> show ospf database
```

OSPF link state database, area 0.0.0.101

Type	ID	Adv Rtr	Seq	Age	Cksum	Len
Router	*10.0.10.254	10.0.10.254	0x80000003	916	0xea40	40
Router	10.0.10.253	10.0.10.253	0x80000006	851	0xc95b	40
Network	10.0.10.2	10.0.10.253	0x80000002	916	0x4598	32
Network	10.0.20.2	10.0.10.253	0x80000002	916	0x4598	32
Network	10.0.20.17	10.0.10.253	0x80000002	916	0x4598	32

```
user@host> show ospf route
```

Prefix	Path	Route	NH	Metric	NextHop	Nexthop
	Type	Type	Type		Interface	addr/label
10.0.10.253	Intra	Router	IP	1	ae0.0	10.0.10.2
10.0.10.254	Intra	Network	IP	0	lo0.0	
10.0.20.2	Intra	Network	LSP	1	ae0.0	10.0.20.2

At this point, connectivity between the loopback interfaces can be verified with pings, a necessary step to make sure that the next protocol (BGP) will work:

```
lab@Node0-MX480> ping 10.0.10.253 source 10.0.10.254
PING 10.0.10.253 (10.0.10.253): 56 data bytes
^C
--- 10.0.10.253 ping statistics ---
4 packets transmitted, 4 packets received, 100% packet success
```

MBGP

Typically, BGP is used to distribute IPv4 prefixes between Internet-facing nodes. While this could be a requirement later, in our case, BGP is used for signaling the virtual private networks—both the L3VPN (*family inet-vpn unicast*) and the VPLS (*family l2vpn signaling*) VPNs use BGP for determining members of the VPN instances. BGP can handle more than IPv4 prefixes for next-layer reachability information (NLRI), including L3VPN, L2VPNs, VPLS, and IPv6 prefixes—hence the multiprotocol BGP (MBGP).

Our BGP implementation was the internal arrangement using loopback interfaces as adjacency points. The service provider had an assigned AS number, but for our purposes, a private AS was used. We had the choice of two different implementations for deploying BGP: the first was a full mesh of all of the devices, and the second was the use of a route reflector. If we were only deploying IPv4 traffic, the route reflector would have simplified the remote locations and provided easier scalability. But with the use of signaling for L3VPNs and VPLS, route reflectors complicated next hops and signaling adjacencies, so we used the traditional full mesh implementation. To simplify the configurations, each location was listed at all sites as a neighbor, and each location had a single adjacency that did not become active (to itself).

The initial configuration of BGP only verified the adjacency settings. As part of the L3VPN and the VPLS configurations, we revisited this configuration and added additional NLRIs.

Here are the settings for the BGP configuration:

```
routing-options {
    autonomous-system 65412;
}
protocols {
    bgp {
        group MPLS {
            type internal;
            local-address 10.0.10.254;
            family inet {
                unicast;
            }
            neighbor 10.0.10.253;
            neighbor 10.0.10.252;
            neighbor 10.0.10.251;
            neighbor 10.0.10.250;
```

```

        neighbor 10.0.10.249;
    }
}

```

The other node configurations are exactly the same, with the exception of the local address. The inclusion of the *family inet* unicast is only necessary when other families are going to be added later.

The command to verify the BGP adjacency is:

```

Lab@Node0-MX480> show bgp summary
Groups: 1 Peers: 1 Down peers: 4
Table    Tot Paths  Act Paths Suppressed  History  Damp State   Pending
inet.0           0           0           0           0           0           0
Peer      AS      InPkt   OutPkt   OutQ   Flaps
10.0.10.253 65412   144928  145517     0     3
Last Up/Dwn State|#Active/Received/Accepted/Damped...
6w3d8h
Establ
inet.0: 0/0/0/0

```

This output shows that the BGP session is established but that no routes have been received from this neighbor—not a big surprise, because at this point we are not supposed to be sending any routes. The other nodes are not shown in the display, but each of them shows an idle state (again, no surprise since the other nodes are not present). This initial configuration is in keeping with the KISS principle: put the configurations in place on the central node, and when the other nodes are deployed, only the new nodes need to be configured.

One of the lessons a warrior must learn is to do as much work as possible in the sane environment of the lab prior to hitting the streets and trying to remember what you’ve forgotten.

MPLS

Once BGP is up and running, it is time to set up MPLS. MPLS has multiple parts: a protocol, a family, and a signaling protocol. In our implementation, RSVP is used for signaling between nodes to establish the label-switched paths. The family and the protocol are set in the MPLS protocol stanza in the configuration, as shown below.

Some additional configuration is done at this point to the backbone interfaces, to adjust their maximum transmission unit (MTU). When MPLS is added to an Ethernet frame, the frame size increases by the length of the MPLS header(s). If the MTU on the interfaces is not increased, full-sized packets (e.g., 1,500 octets) that would normally pass are dropped as too large. Depending on the MPLS service to be deployed, one, two, three, or four MPLS labels can be added to the Ethernet frame. To avoid a guessing game down the road, we set the MTU to the maximum for the backbone interfaces (9192).

The MTU modification was made and the MPLS family was installed on all the interfaces facing the backbone, the *ae0* aggregate Gigabit interface, and the 10 GE interfaces:

```
interfaces {
  xe-0/0/0 {
    description "10Gbps link east";
    mtu 9192
    unit 0 {
      family inet {
        address 10.0.20.1/30;
      }
      family mpls;
    }
  }
  xe-1/0/0 {
    description "10Gbps link west";
    mtu 9192
    unit 0 {
      family inet {
        address 10.0.20.18/30;
      }
      family mpls;
    }
  }
}
ae0 {
  description "Link to Node0 MX-480";
  mtu 9192
  aggregated-ether-options {
    minimum-links 1;
    link-speed 1g;
    lacp {
      active;
    }
  }
  unit 0 {
    family inet {
      address 10.0.10.1/30;
    }
    family mpls;
  }
}
```

The MPLS protocol section of the configuration is initially like those of all the other protocols. Later, path statements will be added. Including all interfaces should not cause any problems, but might prevent omitting this step when adding new interfaces to the network:

```
protocols {
  mpls {
    interface all;
    interface fxp0.0 {
```

```

        disable;
    }
}

```

MPLS by itself is like a car without fuel—it looks nice in the driveway, but it won't get you anywhere. To verify that MPLS is actually operating as designed, we have to add a signaling protocol to create the label-switched paths.

RSVP

The MPLS path signaling protocol is RSVP. As with MPLS, there are two parts to the configuration: the protocol section, which is like all the others, and the path stanzas, which are part of MPLS. The protocol section is very simple:

```

protocols {
    rsvp {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}

```

The path statements define the routes that traffic follows between the nodes. If no constraints are added to the path statement, the shortest path (OSPF) is used. However, in our environment, we cannot use the shortest path for all traffic (remember the traffic requirements defined earlier?). A selling point of RSVP was that it supports the definition of constrained paths that are named and may be used in a primary/secondary arrangement.

For the initial traffic, the Gigabit aggregate interface is used as the primary path, the western 10 GE interface as the second path, and the eastern 10 GE path as the tertiary path. The path statements define the intermediate addresses that must be passed on the way to the path destination. In the configuration below, the destination of the path is the loopback address of the node, while the path statements identify the addresses to be used to get to the loopback address.



For those of you that have used a GPS on a boat or for hiking, you could consider these intermediate addresses as waypoints on a GPS route. Each identifies a signpost along the way to the destination. Experienced network warriors will look at this as a token ring source route.

In a more complicated network, the path statements can contain a list of intermediate addresses and specify the use of strict (must hit all in order) or loose (must hit all via any possible path) mode. For this network, a single entry was all that was needed. The configuration looked like this:

```
protocols {
  mpls {
    label-switched-path to-node1 {
      to 10.0.10.253;
      no-cspf;
      standby;
      primary 1G;
      secondary 10G-west;
      secondary 10G-east;
    }
    path 1G {
      10.0.10.2;
    }
    path 10G-east {
      10.0.20.17;
    }
    path 10G-west {
      10.0.20.2;
    }
  }
}
```

The LSP configuration needs a little explanation. The name of the LSP is “to-node1,” and the destination of the LSP is 10.0.10.253 (*lo0.0* of node 1). The *no-cspf* command allows this LSP to be established even if traffic engineering is not enabled on the IGP. The *standby* command establishes the backup paths when the primary path is established. This saves time and traffic in the case of an incident with the primary path.



If a nonrevertive LSP were desired, multiple secondary paths would be defined and no primary. The first secondary would be used initially; in the event of a failure, the second would take over and would keep its role even when the first returned to action.

A corresponding configuration was established on *Node1*, and the configurations were committed on both. To verify that the MPLS was functioning, the following commands were used:

```
lab@Node0-MX480> show mpls path
```

Path name	Address	strict/loose if-id
10G-east	10.0.20.17	strict<empty>
10G-west	10.0.20.2	strict<empty>
1G	10.0.10.2	strict<empty>


```
lab@Node0-MX480> show route table inet.3
inet.3: 1 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.10.253/32 *[RSVP/7/1] 6d 02:03:22, metric 50
    > to 10.0.10.2 via ae0.0, label-switched-path to-node1
    to 10.0.20.2 via xe-1/0/0.0, label-switched-path to-node1
    to 10.0.20.17 via xe-0/0/0.0, label-switched-path to-node1
```

```
lab@Node0-MX480> show mpls lsp
Ingress LSP: 1 sessions
To          From          State   Rt P    ActivePath    LSPName
10.0.10.253 10.0.10.254 Up      0 *    1G            to-node1
Total 1 displayed, Up 1, Down 0
```

```
Egress LSP: 1 sessions
To          From          State   Rt Style Labelin Labelout LSPName
10.0.10.254 10.0.10.253 Up      0 1 FF      3        - to-node0
10.0.10.254 10.0.10.253 Up      0 1 FF      3        - to-node0
10.0.10.254 10.0.10.253 Up      0 1 FF      3        - to-node0
Total 3 displayed, Up 3, Down 0
```

```
Transit LSP: 0 sessions
Total 0 displayed, Up 0, Down 0
```

```
lab@Node0-MX480> show rsvp interface
RSVP interface: 13 active
Interface   State Active Subscr- Static Avail- Reserved Highwater
            Up    resv   iption BW    able BW    BW      mark
ae0.0       Up      1    100% 2Gbps 2Gbps 0bps    0bps
xe-0/0/0.0  Up      0    100% 10Gbps 10Gbps 0bps    0bps
xe-1/0/0.0  Up      0    100% 10Gbps 10Gbps 0bps    0bps
```



The most important display above is the one for route table *inet.3*. All traffic that is using the LSP for transport must receive its next hop from the *inet.3* table. If the next hop for the traffic is not found in the *inet.3* table, the traffic will not pass over the MPLS path. Because all the BGP routes use the loopback address as the next hop, this address has to be found in the *inet.3* table to pass L3VPN or VPLS traffic.

The MPLS routes show the MPLS labels that are used to send MPLS frames between *Node0* and *Node1*, considering that there are no intermediate routers in line between *Node1* and *Node0*. The outgoing frames are marked with a label of 30000, and incoming frames are popped of their labels:

```
lab@Node0-MX480> show route table mpls
mpls.0: 24 destinations, 24 routes (24 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```

0          *[MPLS/0] 7w4d 22:33:15, metric 1
           Receive
1          *[MPLS/0] 7w4d 22:33:15, metric 1
           Receive
2          *[MPLS/0] 7w4d 22:33:15, metric 1
           Receive
299856     *[RSVP/7] 3d 02:46:03, metric 1
           > to 10.0.10.253 via ae0.0, Pop

```

Once MPLS is up and operational, RSVP is active, and paths are established between the edge devices (*Node1* and *Node0*), the VPNs can be created.

Layer 3 VPN

A Layer 3 VPN takes traffic from a customer-facing interface and forwards it to other customer-facing interfaces across the MPLS network. All traffic is encapsulated in a customer label and a transport label. At the remote locations, the transport label is popped and the customer label is used to segregate the traffic between customers. Each customer's traffic is segregated in a routing instance. In our implementation, all voice traffic is grouped together in a routing instance called *L3-Voice-VPN*. The initial configuration of the L3VPN involves the signaling between the L3VPN edge nodes. BGP is used for this signaling. The transformation of BGP from an Internet routing protocol to an L3VPN signaling protocol involves the addition of a new protocol family to the existing BGP hierarchy:

```

protocols {
  bgp {
    group MPLS {
      type internal;
      local-address 10.0.10.254;
      family inet {
        unicast;
      }
      family inet-vpn {
        unicast;
      }
      neighbor 10.0.10.253;
      neighbor 10.0.10.252;
      neighbor 10.0.10.251;
      neighbor 10.0.10.250;
      neighbor 10.0.10.249;
    }
  }
}

```

The second component of the L3VPN is the routing instance. A number of new interfaces (customer facing) are added to the L3VPN routing instance. These interfaces connect the MX480 to the VoIP switch, the local DSLAM, and the management network. Even though the interfaces are referenced in the routing instance, they have to be defined first in the interface stanza:

```
interfaces {
  ae1 {
    description "Link to Node0 DSLAM";
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
  }
  unit 110 {
    vlan-id 110;
    family inet {
      address 10.110.0.1/20;
    }
  }
  unit 120 {
    vlan-id 120;
    family inet {
      address 10.120.0.1/20;
    }
  }
}
ae6 {
  description "Link to VoIP Switch";
  unit 0 {
    family inet {
      address 10.0.10.21/30;
    }
  }
}
```

Now that the interfaces are defined, we can turn our attention to the routing instance. There are a number of components to this stanza:

```
routing-instances {
  L3-Voice-VPN {
    instance-type vrf;
    interface ae1.110;
    interface ae1.120;
    interface ae6.0;
    route-distinguisher 10.0.10.254:114;
    vrf-target target:65412:114;
    vrf-table-label;
    protocols {
      ospf {
        export bgp-ospf;
        area 0.0.0.0 {
          interface ae6.0;
          interface ae1.110 {
```

```
    passive;
  }
  interface ae1.120 {
    passive;
  }
}
}
```

The instance type of *vrf* (virtual routing and forwarding) is a holdover from the specifications that refers to the table for this routing instance. It is used for Layer 3 VPNs. The *route-distinguisher* is the BGP entity that is carried in the *inet-vpn* NLRI. It uniquely names the device and the VPN. The route target is also used by BGP, to identify the incoming and outgoing community string that accompanies the *inet-vpn* NLRI. For simple installations such as this, the single *target* command can be used. For more complex installations, separate input and output target commands are used along with routing policies, as in:

```

routing-instances {
  L3-Voice-VPN {
    instance-type vrf;
    interface ...
    route-distinguisher 10.0.10.254:114;
    vrf-import 114-in;
    vrf-export 114-out;
    ...
  }
}

policy-options {
  policy-statement {
    vrf-import {
      term accept {
        from {
          protocol bgp;
          community target:114;
        }
        then {
          accept;
        }
      }
      term reject {
        then {
          reject;
        }
      }
    }
    vrf-export {
      from {
        protocol direct;
        protocol ospf;
      }
    }
  }
}

```

```

    }
    then {
        community-add target:114;
        accept;
    }
}
}
community {
    target:114 {
        members {
            target:65412:114;
        }
    }
}
}

```

This would accomplish the same thing as the single route target command and allow the administrators to create custom instances and routing situations for customers (extranets and Internet access). These added features are not needed for our installation for voice and management traffic, though, so KISS rules again and the simpler route target command suffices.

The *vrf-table-label* command allows the device to act as a provider router as well as an edge router. Finally, the OSPF protocol is installed in the routing instance. This routing protocol will interface to the other routers in the management network as well as the voice network, allowing all prefixes to be seen by all the other VPN devices. Buried in the OSPF configuration is another routing policy, this one allowing the BGP routes to be installed into the OSPF database. The L3VPN routes are all advertised by the BGP *inet-vpn* NLRI from each of the remote locations. They might be VPN routes, but they are still advertised by BGP, and as such, they will not be transferred to OSPF without a policy:

```

policy-options {
    policy-statement bgp-ospf {
        term 1 {
            from protocol bgp;
            then accept;
        }
        term 2 {
            then reject;
        }
    }
}

```

BGP installs the routes in the VRF routing table, and the policy copies these prefixes to the OSPF database as external routes. Any other routes that happen to be learned are not included in the copy process (*then reject*).

To verify that the configuration is operational, *Node1* has to be set up and committed. The configuration for *Node1* is:

```

interfaces {
  ae0 {
    description "Link to Node0";
    unit 0 {
      family inet {
        address 10.0.10.2/30;
      }
      family mpls;
    }
  }
  ae1 {
    description "Link to DSLAM";
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 114 {
      vlan-id 114;
      family inet {
        address 10.114.0.1/20;
      }
    }
    unit 124 {
      vlan-id 124;
      family inet {
        address 10.124.0.1/20;
      }
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 10.0.10.253/32;
        address 127.0.0.1/32;
      }
    }
  }
}
routing-options {
  autonomous-system 65412;
}
protocols {
  rsvp {
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
  mpls {
    label-switched-path to-node0 {
      to 10.0.10.254;
      no-cspf;
      standby;
      primary 1G;
    }
  }
}

```

```

        secondary 10G-east;
        secondary 10G-west;
    }
    path 1G {
        10.0.10.1;
    }
    path 10G-east {
        10.0.20.18;
    }
    path 10G-west {
        10.0.20.1;
    }
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    group MPLS {
        type internal;
        local-address 10.0.10.253;
        family inet {
            unicast;
        }
        family inet-vpn {
            unicast;
        }
        neighbor 10.0.10.254;
        neighbor 10.0.10.253;
        neighbor 10.0.10.252;
        neighbor 10.0.10.251;
        neighbor 10.0.10.250;
    }
}
ospf {
    reference-bandwidth 100g;
    area 0.0.0.101 {
        interface lo0.0;
        interface ae0.0;
        interface xe-0/0/0.0 {
            metric 200;
        }
        interface xe-1/0/0.0 {
            metric 200;
        }
    }
}
}
routing-instances {
    L3-Voice-VPN {
        instance-type vrf;
        interface ae1.114;
    }
}

```

```

    interface ae1.124;
    route-distinguisher 10.0.10.253:114;
    vrf-target target:65412:114;
    vrf-table-label;
  }
}

```

To verify that the L3VPN is up and operational, three checks are performed. The first is the status of the label-switched paths, which can be verified by using the *show mpls lsp* command:

```

{master}
root@Node1-MX480> show mpls lsp
Ingress LSP: 5 sessions

```

To	From	State	Rt	P	ActivePath	LSPname
10.0.10.253	10.0.10.254	Up	0	*	1G	to-Node1

```

Total 1 displayed, Up 1, Down 0

```

The next step is to verify that the local route table shows the prefixes from the remote node learned via BGP, and points to a next hop via an MPLS label. This can be done with a *show route* command—specifically, *show route table 114*. The output has been shortened here for the sake of brevity, as all we want to show is that prefixes are being learned over the MPLS links:

```

{master}
root@Node1-MX480> show route table 114
114.inet.0: 120 destinations, 121 routes (120 active ...)
+ = Active Route, - = Last Active, * = Both

10.100.255.252/30 *[OSPF/10] 7w3d 03:18:27, metric 12
    > to 10.0.10.22 via ae6.0
10.101.0.0/22 *[BGP/170] 6d 01:57:38, localpref 100, from 10.0.10.249
    AS path: I
    > to 10.0.10.14 via ae5.0, label-switched-path to-node0
10.101.80.0/22 *[OSPF/150] 1w4d 04:57:48, metric 20, tag 0

```

Finally, we need to verify that traffic can be generated over the VPN to the far-end addresses. *ping* and *traceroute* are handy here.

First we verify that the address is not reachable by the straight IP network:

```

{master}
root@Node1-MX480> ping 10.114.0.1
PING 10.114.0.1 (10.114.0.1): 56 data bytes
ping: sendto: No route to host
ping: sendto: No route to host
ping: sendto: No route to host
^Z
Suspended

```

This fails because the address does not appear in the *inet.0* route table, but rather in the *114.inet* table. So, we try again with the *routing-instance* flag:


```
{master}
root@Node1-MX480> ping 10.114.0.1 routing-instance 114
PING 10.114.0.1.21 (172.12.101.21): 56 data bytes
64 bytes from 10.114.0.1: icmp_seq=0 ttl=64 time=1.187 ms
64 bytes from 10.114.0.1: icmp_seq=1 ttl=64 time=0.533 ms
^Z
Suspended
```

This time the ping is successful. Now that the L3VPN is up and operational, the final layer on this configuration can be laid—the VPLS service.

VPLS

Like the L3VPN, VPLS has a number of components that work together to allow Ethernet traffic to be securely passed over an IP backbone. It uses the same MPLS and RSVP LSPs that have been previously defined, and it requires the addition of another NLRI to the BGP stanza, plus a routing instance for each broadcast domain (VLAN). For the test bed, only two VPLS instances are going to be configured (VLAN 70 and VLAN 434).

VPLS acts as a LAN extension technology—it makes the MPLS network look like a virtual VLAN switch with portions in all routers that contain the same VPLS NLRI. The VLAN traffic is carried as a stripped-down Ethernet frame between MXs. In theory, the upper-layer traffic can be any protocol, although this tribe has not had the opportunity to test that theory. Each router maintains a MAC learning table and forwards traffic based on the table; the full mesh of MPLS LSPs allows traffic to pass freely between any two locations.

For this engagement, the VPLS traffic was Internet traffic between DSL customers and the ISP's authentication servers and routers. Because of regulatory issues, the transport side of the house (teleco) had to have a clean handoff to the service side of the house (ISP). VPLS allowed the clean handoff between the MXs and EXs (teleco) and the DSLAMs and Internet routers (ISP).



We warriors mentioned that the ISP's routers could be included in the MX as a routing instance, saving equipment, space, and power. But our customer reminded us of the lack of imagination in the regulatory offices, and that a different box is easier to “see” than a virtual router. As much as technology advances, some things are better left alone.

Like the L3VPN, the VPLS configuration needs additional customer-facing logical interfaces. These interfaces use a special encapsulation called *vlan-vpls*:

```
interfaces {
    ae2 {
        description "Link to Internet";
        flexible-vlan-tagging;
```

```

encapsulation flexible-ethernet-services;
aggregated-ether-options {
    minimum-links 1;
    link-speed 1g;
    lacp {
        active;
    }
}
unit 720 {
    encapsulation vlan-vpls;
    vlan-id 70;
}
unit 434 {
    encapsulation vlan-vpls;
    vlan-id 434;
}
}
ae1 {
    description "Link to Node0 DSLAM";
    flexible-vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 72 {
        encapsulation vlan-vpls;
        vlan-id 70;
    }
    unit 496 {
        encapsulation vlan-vpls;
        vlan-id 434;
    }
}
}

```

The traffic arriving on these interfaces is encapsulated in the MPLS frames as Ethernet frames and transported over the backbone based on the MAC learning tables. In the lab setup, there is just a single destination for the traffic, but in the final configuration, the frame could be sent to any of the other five locations.

The routing instance configurations are the same as a previous chapter and so are omitted here. Once the configuration was committed and the remote device was set up, the VPLS configuration was verified with the following commands:

```
root@Node0-MX480> show vpls connections
```

```
Layer-2 VPN connections:
```

```
Legend for connection status (St)
```

EI -- encapsulation invalid	NC -- interface encapsulation not CCC/TCC/VPLS
EM -- encapsulation mismatch	WE -- interface and instance encaps not same
VC-Dn -- Virtual circuit down	NP -- interface hardware not present
CM -- control-word mismatch	-> -- only outbound connection is up
CN -- circuit not provisioned	<- -- only inbound connection is up

OR -- out of range Up -- operational
 OL -- no outgoing label Dn -- down
 LD -- local site signaled down CF -- call admission control failure
 RD -- remote site signaled down SC -- local and remote site ID collision
 LN -- local site not designated LM -- local site ID not minimum designated
 RN -- remote site not designated RM -- remote site ID not minimum designated
 XX -- unknown connection status IL -- no incoming label
 MM -- MTU mismatch MI -- Mesh-Group ID not available
 BK -- Backup connection ST -- Standby connection
 PF -- Profile parse failure PB -- Profile busy
 RS -- remote site standby SN -- Static Neighbor

Legend for interface status

Up -- operational

Dn -- down

Instance: VLAN-434

Local site: Node0 (1)

connection-site	Type	St	Time last up	# Up trans
5	rmt	Up	Jul 29 06:24:56 2011	1

Remote PE: 10.0.10.250, Negotiated control-word: No

Incoming label: 800004, Outgoing label: 800000

Local interface: vt-0/0/0.1050135, Status: Up, Encapsulation: VPLS

Description: Intf - vpls VLAN-434 local site 1 remote site 5

Instance: VLAN-70

Local site: Node0 (1)

connection-site	Type	St	Time last up	# Up trans
2	rmt	Up	Jun 24 08:25:25 2011	1

Remote PE: 10.0.10.253, Negotiated control-word: No

Incoming label: 800065, Outgoing label: 800008

Local interface: vt-0/0/0.1050112, Status: Up, Encapsulation: VPLS

Description: Intf - vpls VLAN-70 local site 1 remote site 2

root@Richmond-MX480> **show vpls mac-table**

MAC flags (S -static MAC, D -dynamic MAC, L -locally learned

SE -Statistics enabled, NM -Non configured MAC,

R -Remote PE MAC)

Routing instance : VLAN-434

Bridging domain : __VLAN-434__, VLAN : NA

MAC address	MAC flags	Logical interface
00:17:0f:ac:a0:80	D	ae1.434
00:17:c5:99:b5:59	D	vt-0/0/0.1050140
00:25:45:fb:af:da	D	vt-0/0/0.1050140

MAC flags (S -static MAC, D -dynamic MAC, L -locally learned

SE -Statistics enabled, NM -Non configured MAC,

R -Remote PE MAC)

```
Routing instance : VLAN-70
Bridging domain : __VLAN-70__, VLAN : NA
  MAC          MAC          Logical
00:04:ed:aa:5d:c6 D          vt-0/0/0.1050135
00:04:ed:aa:5f:d4 D          vt-0/0/0.1050135
00:04:ed:b1:20:81 D          vt-0/0/0.1050135
00:04:ed:bb:91:1b D          vt-0/0/0.1050135
00:04:ed:db:d1:73 D          vt-0/0/0.1050135
00:04:ed:db:d1:c2 D          vt-0/0/0.1050135
00:06:b1:0e:6f:c9 D          vt-0/0/0.1050135
00:10:db:1c:fc:e6 D          vt-0/0/0.1050135
00:11:24:a7:f1:c2 D          vt-0/0/0.1050135
```

address	flags	interface
---------	-------	-----------

The output from the *show vpls connection* command shows that the remote location is connected, and the *show vpls mac-table* command output shows that remote devices are populating the local table. These commands verify that the VPLS system is up and running. As this is a Layer 2 environment, we cannot use a *ping* or *traceroute* to test connectivity. The teleco engineers verified that DSL users could log in and get Internet connectivity, and that was verification enough for us.

OBM

We now have the customer configuration taken care of, so it is time for the out-of-band management (OBM) network.

Management of the network can be performed in-band or out-of-band. In-band management saves facilities and often used better transmission facilities, but suffers from survivability issues. If the network goes down, so does the management network—not a good situation for a carrier-class implementation. The OBM topology is shown in [Figure 6-3](#).

In keeping with the KISS philosophy, the configurations for the OBM network are simple—the management interfaces are set with a common addressing scheme that is accessible from the teleco’s internal network. The management interfaces are not part of the back-bone OSPF network. Here is the MX configuration:

```
interface
  fxp0 {
    unit 0 {
      family inet {
        address 192.168.1.3/24;
      }
    }
  }
```

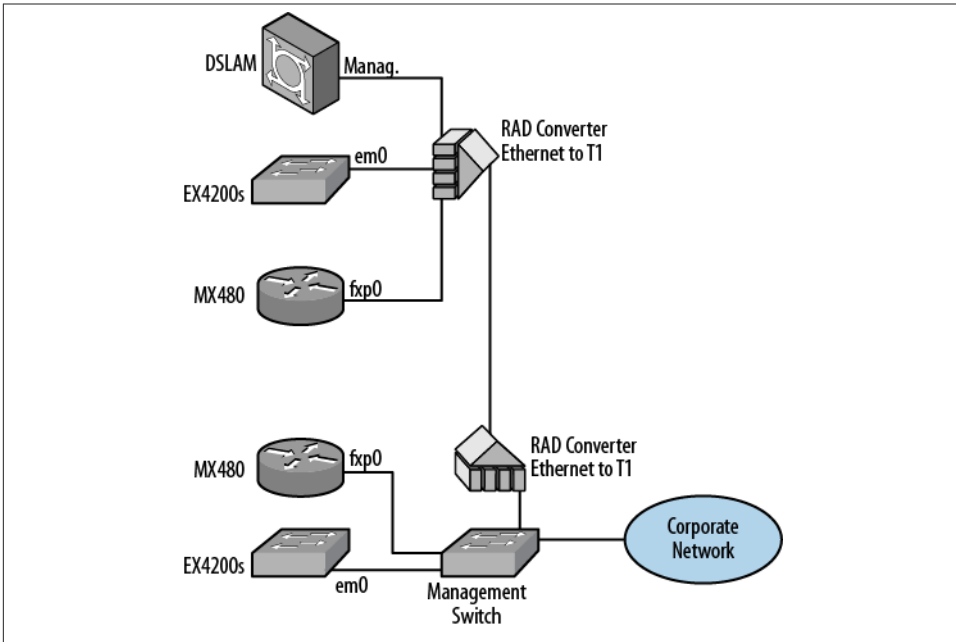


Figure 6-3. OBM network

And here is the configuration for the EX switches:

```
interface
  me0 {
    unit 0 {
      family inet {
        address 192.168.1.4/24;
      }
    }
  }
}
```

These addresses are reached by an EX switch that is carrying Layer 3 traffic and running the OSPF protocol on the area 0 domain. All interfaces can be reached from the comfort of the system administrator's desk rather than from the high-noise environment of the equipment room. You have to love OBM.

Conclusion

This engagement was a great combination of technologies: routing, switching, and local and wide area transport. The added requirements of high availability and survivability made the configurations more extensive than the other implementations we've looked at so far. Further, the use of both VPLS and L3VPNs required a combination of BGP and an IGP.

The configuration required an onion-like approach, with each layer supporting the next. Each layer could be verified independently and, if necessary, troubleshooting could be done prior to the next layer being added, giving us warriors not only a configuration plan but an implementation plan. (Each layer was shown independently in this chapter and could be used in the full implementation.)

The implementation started with two locations and was completed by including all the sites. In addition, there was a requirement by the wireless carrier for an L2VPN connection to a couple of sites. This traffic was limited to the 10 GE fiber links and required some additional MPLS configurations, which were added after this engagement was concluded.

That's typical in this network warrior life—you pick up something from one assignment and it gets carried to the next. You read a good networking book, or talk shop around the bar with some other warriors, and you take those clues and tricks to the next assignment. I've gone back to this carrier several times over the past few years and tweaked what we did based on what I've learned from keeping my ears and eyes open.

A PCI-Compliant Data Center

For most clients, the goal of a Juniper professional service engagement is to improve the client's network. Our clients are typically looking to us network warriors to increase the speed of their networks, or for a new set of services—and this is where the “warrior” in network warrior comes in. We are the smoking gun, the get-this-done fix-it specialist and miracle worker all rolled into one professional service engineer. This is also where Juniper Networks has really made a name for itself this past decade, by being faster, more reliable, and more feature-rich than the competition, and thus making our warrior jobs much easier.

In almost every case, the result of the engagement is an increase in connectivity and/or availability for the client's network. That is why this next engagement was totally different from all the others I have participated in—we were brought in to deter connectivity and reduce the availability of services.

Introduction

For this engagement, the tribe was brought in to make a client's data center compliant with Payment Card Industry (PCI) regulations. This client deals with a high volume of customer credit cards and personal and credit information. As such, they must meet strict regulations set forth by the government for the protection of this information. For those of you that have not played with PCI compliance, check out the information found on the official [PCI Security Standard Council website](#).

From a network warrior's point of view, the basics of PCI compliance are:

- Logging and monitoring access to the networks and servers containing sensitive data
- Protecting sensitive data

- Controlling and authenticating user access to servers with sensitive data
- Detecting and preventing intrusions and vulnerability scans
- Compliance testing and auditing

Our task for this engagement was to secure access to the data center for the client so that they could pass the PCI audits and stay in business.

Our tribe for this engagement consisted of a seasoned team of Juniper engineers, security professionals from the client, and myself. We were to install a new data center infrastructure that would not only meet the requirements of the PCI auditors but also provide a scalable solution for the client. We started with a network assessment that identified the security holes and network bottlenecks in the existing design. We then gathered a set of requirements for the operation of the network going forward (and some surprise requirements that cropped up along the way). What is presented in the following pages has been completely sanitized, for obvious reasons. Some of the configurations are purposely vague, so as to avoid any possibility of guessing or backward engineering vulnerabilities to similar types of implementations.

Client Goals

The client's goals fell into two camps: security goals and scalability goals. On the security side, the overriding goal was PCI compliance. Our focus here was on the protection of the servers that contained the sensitive information. The desktops, laptops, scanners, and other devices with access to this sensitive information were to be dealt with separately. The data center architecture was a wide-open invitation for a hacker or angry employee. There was no security except passwords protecting the client's information.

The first security goal was to create an access and tracking system for users who needed to get to the servers containing the sensitive information. All user access was to be logged on a secured server, and all changes to the security devices were also to be logged.

The next security goal was to block unauthorized access to the sensitive information, or, better put, allow access to the information only for authorized users.

The third major security goal was to prevent prying eyes and machines from looking at or hacking into the servers.

Once these main security goals were recorded and confirmed with upper management, an additional requirement was provided. It seemed that the various departments that managed the servers also defined the access to the data on those servers and required administrative access to the policies that protected that data. To make things a little stranger, the departments were not allowed to manage other departments' security policies, so there was no unified server security group that oversaw all this information. Weird!

On the scalability side of the requirements, any design had to be able to grow to double the existing capacity without a forklift upgrade. All elements in the design had to be able to expand to meet the expected growth in the amount of traffic and the number of servers. Considering that the data center is the bloodline for this client, any solution also had to be survivable.

The existing data center network was a very simple design (Figure 7-1) installed with little thought to throughput, security, or separation of functions or departments. The servers had been installed as necessary by the various departments to ensure sufficient storage for those departments' data. Each department (shown as different letters, R, B, G, and Y, in the diagram) was responsible for its own software maintenance; many of the applications required legacy terminal-type access to databases. Also, each department knew the who, what, and where for each of its applications, but for the most part, did not part with this information easily. You get the idea.

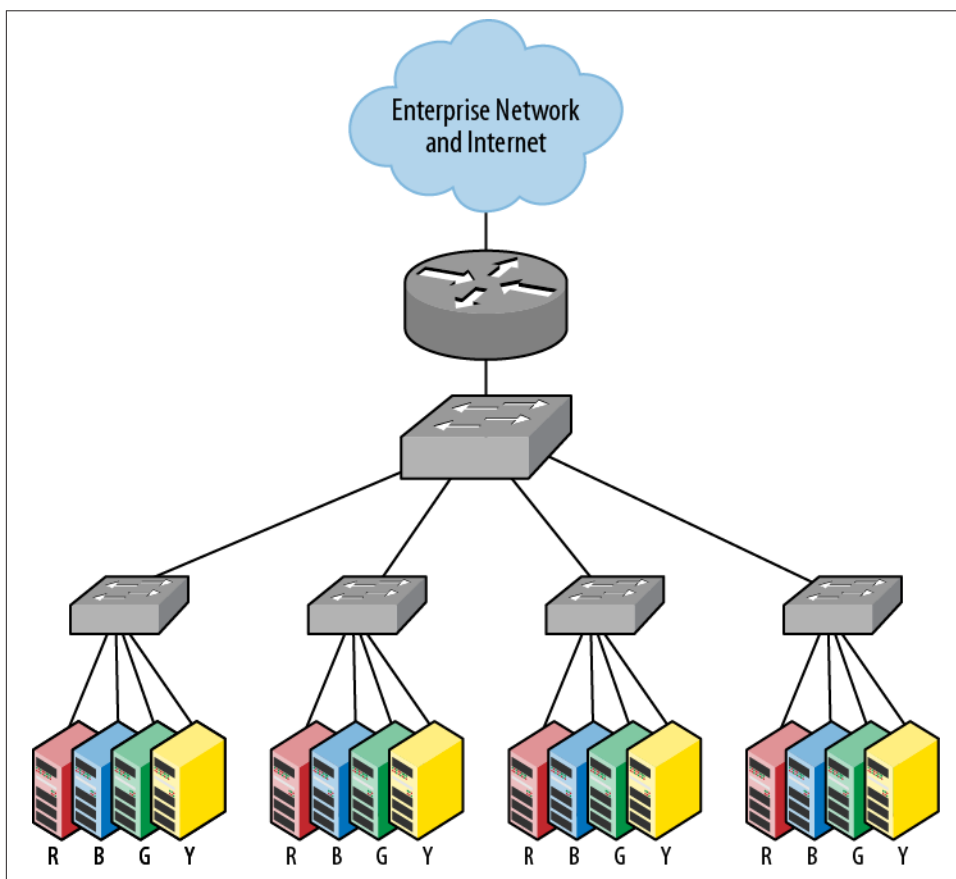
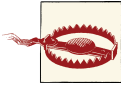


Figure 7-1. Existing data center design



Whenever a company maintains its own applications, beware of strange use of the communications protocols, different timers, and all in all an interesting time identifying traffic on the network.

The connectivity for the data center was provided by a mixture of new and old technologies: Cisco, Foundry, Nortel, and Extreme. The link speeds were a mixture of 10 Mbps, 100 Mbps, and 1 Gbps, depending on the age of the servers and the capabilities of the switching platforms. The routers to the rest of the network (shown in the diagram as a single device) were a pair of Cisco 7200s that had reached their end of life quite a while ago.

The IP addressing was the only piece of the puzzle that was actually thought out—or maybe it worked well by chance, but in any case, it helped in the final design. In keeping with management’s silo mindset, each department was assigned a VLAN and an IP subnet in the data center. All servers used the same default gateway, and all could communicate with one another directly.

As the Juniper sales rep stated, this was a target-rich environment, and we were the warriors that were going to attack it.

Design Trade-Offs

Once the client’s requirements were collected and understood, the process of designing the new secure data center could begin. The design involved a number of trade-offs, but firewalls, routing, addressing, and survivability were the main ones. Let’s review each briefly.

Firewalls

The selection of a firewall or firewalls was the key to the entire design. The scalability and feature set all pointed to a single chassis-built firewall, but the per-department manageability aspect pointed to individual firewalls. The individual firewall solution did not meet the scalability requirements of the customer, and the single chassis did not meet the management requirements. We thought we would have to compromise between the two (that never makes anyone happy, and often makes all parties unhappy), but fortunately, a member of the tribe brought forward another option.

In Junos 11.2, a Netscreen feature called *virtual systems* was introduced to the data center SRX product line (here, the term *logical systems* is used). The release notes provide a great synopsis of this feature:

Logical systems enable you to partition a single device into multiple secure logical routers, each with its own discrete administrative domain, logical interfaces, routing instances, security firewall and other security features.

We had our solution without compromise: we could deploy a chassis-built firewall while offering management capabilities to the individual departments. The only issue with this selection was that the JTAC recommended Junos version chart listed 10.4 as the recommended version for the SRX product line. This meant we would have to lab test and bug scrub the 11.2 version prior to production deployment (bummer, we were going to have to play with equipment in the lab again!).

Routing

The trade-off involved in the routing aspects of the design was one of manageability versus complexity. The current system was a complete Layer 2 design at the data center. All traffic from the same server department never crossed a router. Only when traffic was destined for or originated from the user community was routing involved. The core router was part of an OSPF backbone area. The proposed design would push the firewalls closer to the servers, and they would all be operating in a Layer 3 mode (as opposed to transparent mode for the firewalls).



The Juniper Networks firewalls can operate in a transparent mode where the firewall is a transparent switch. All traffic that passes through the firewall is given the full seven-layer security treatment, but the device itself is a “bump in the wire.”

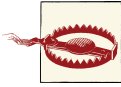
So, the trade-off was to either use static routes between the firewalls and routers, or use a dynamic routing protocol.

The design in this case was a compromise. A combination of OSPF and static routes was proposed:

- OSPF was to run between the router and the backbone network.
- Static (default) routes were to be used for single path elements and to inject non-local routes into the backbone.

This compromise provided us with the greatest flexibility and the most manageability, while keeping the complexity at an acceptable level.

One of the manageability aspects of the design that had to be explained was that the routing portion of the network was not going to be managed by the individual departments. All addressing and routing for the firewalls and the routers had to be performed by the networking group. This idea initially met with resistance, but the networking department agreed to work with the individual services departments to meet their requests in a timely manner.



When dealing with a dysfunctional family, it is best to understand where the long-standing feuds are rooted. As network warriors, you'll see many battles, some internal, some external. In all cases, however, it is best not to pick sides in these feuds, but rather to offer mediation services and identify the middle ground so all sides can advance. It's just like dealing with your own family.

Addressing

As stated previously, the addressing was the only portion of the existing design that was logical and orderly. Each of the switches was set up with virtual LANs (VLANs), and each department had an IP prefix range that was used for all the servers in that VLAN. The servers all pointed to the hot standby router protocol (HSRP) virtual IP address that was being used by the routers.

This addressing would make the migration and the installation of new connectivity hardware much easier than we had originally expected. We thought we would have to totally readdress the servers and the DNS system, entering dual addresses and cutting over the DNS records on the same evening as the hardware, but every now and then the network gods smile on us warriors as we trudge through these engagements.

Survivability

The last of the major trade-offs was in survivability. The client wanted any new installation to provide four-nines availability.



“Four-nines” availability is saying the system is available 99.99% of the time. Assuming my math is correct, there are 525,600 minutes in a year, and 99.99% of that is 525,547.44 minutes/year. The difference amounts to approximately 52 minutes per year of downtime. So, if you can fix it in under an hour, you can meet the four-nines availability requirement, as long as there is not more than one outage-causing failure in a year. Five-nines availability cuts that down to 5 minutes per year of downtime!

This requirement forced us to provide redundancy in all aspects of the design, so the trade-off here was one of cost versus availability. We started redundancy at the interconnections to the servers. The top-of-rack (ToR) switches, EX4200s in our initial design, were interconnected to form a virtual chassis, adding redundancy with a minimal cost of additional cabling. The uplinks from the ToR switches to the router (an MX240 in our design) could be individual links, link aggregation groups (LAGs), or redundant

trunk groups. All three choices offer reliability, link detection, failover, and adequate bandwidth. When we add the scalability factors into the mix, the use of LAGs offers an in-service upgrade of bandwidth along with a graceful failover in the event of the loss of a line card or a link.

The option of a virtual chassis and a multiswitch LAG configuration is one of the features that set the Juniper Networks EX-series product line apart in the industry today.

Moving up the design to the router, the trade-off here was one of the reliability of a single device versus the simplicity of the installation of multiple devices. Due to the redundancy built into the high-end MX Universal Edge routers (redundant power systems, routing engines, and switching systems), the use of multiple smaller routers did not make sense. Add to this the scalability factors, and the chassis-built MX240 was the clear choice for the router functions. If this decision proved to be an issue in the future, an MX virtual chassis could also be an option (at a cost).

The final piece of the data center design was the firewall. The SRX series of firewalls provide reliability through clustering, and in our design, clustering offered virtually unlimited scalability while meeting the availability numbers required by the client. This wasn't really a trade-off, but rather a design criterion. Other options exist for a survivable firewall solution, but none offer the cost benefits of a clustered SRX.

Recommended Design

Our recommended design is shown as a simplistic diagram in [Figure 7-2](#). The diagram shows a two-layer approach to the data center connectivity. The first layer is Layer 2 switching, providing interconnectivity between any two servers. The second layer is a routing layer supported by a combination of a router and firewall. The router provides connectivity between the user community and the servers, while the firewall restricts and logs access from those users to the same servers. [Figure 7-2](#) does not show the various levels of reliability and virtualization proposed for the design.

Switching Layer

The switching layer is composed of top-of-rack switches interconnected into a virtual chassis. These switches are EX-series switches. Depending on the model, the EXs can support between 6 and 10 switches in a virtual chassis. We selected EX4200s because their combination of 10G ports and 10/100/1000 ports provided the features and capabilities needed for the design. They also supported the scaling requirements and the survivability requirements of the design. Their connectivity within the virtual chassis is provided by 10 Gbps links between the switches, arranged in a survivable ring configuration. If a single link were to fail, no traffic loss would be experienced.

Each switch can be configured with two interconnection ports, providing an upstream and a downstream interconnection. In a virtual chassis, a single switch's routing engine

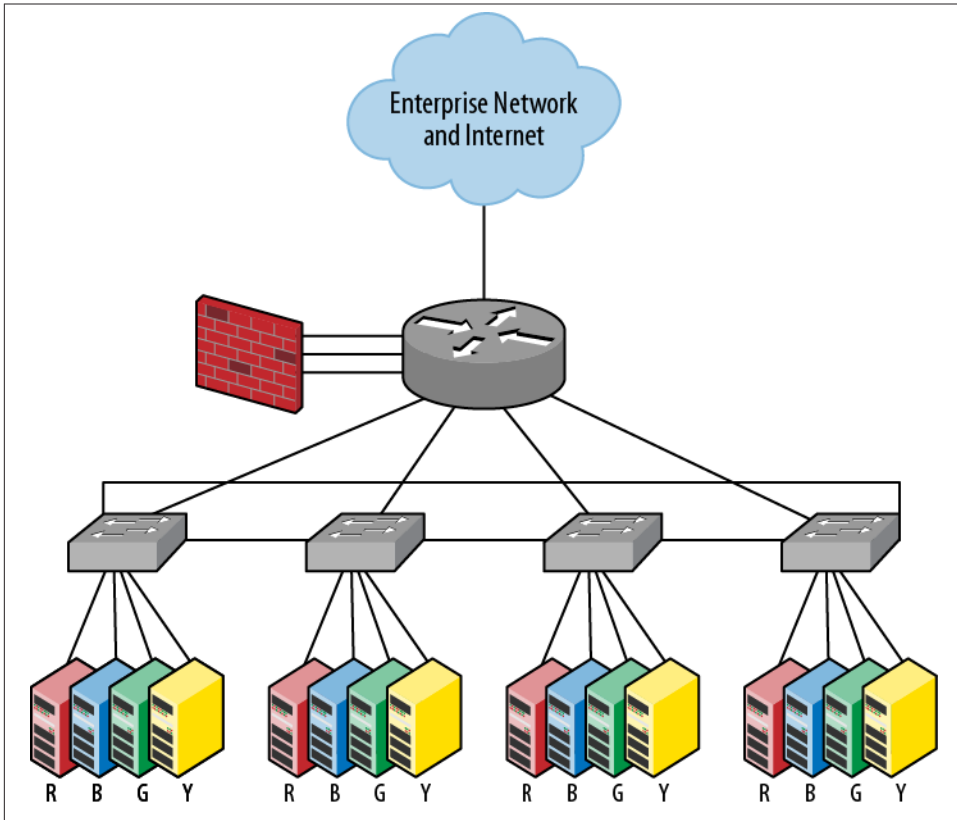


Figure 7-2. Recommended design

is active. One switch is designated as a backup, and the routing engines in the remaining switches think of themselves as line cards. This allows for recovery from a failure of the routing engine (again, without service interruption). The interchassis links carry control traffic as well as interswitch user traffic. The EXs are connected to the routing layer by a multichassis LAG. The LAG supports bandwidth updates and loss of links without a total loss of connectivity. The four interconnecting links between the switching layer and the routing layer are viewed as a single link as far as routing and switching are concerned, but act as four links as far as survivability is concerned. This configuration does away with the inefficiencies of the spanning tree protocol while providing superior failure recovery.

The switching layer supports intradepartment connectivity via the VLANs that were used in the original addressing scheme. Traffic on the same VLAN is connected at the

switch layer directly between the ingress switch and the egress switch. If the data center grows larger than the number of switches in a virtual chassis, then the routing layer will be used for interconnection between the servers in the same department. For our client, this need is probably a number of years away.

Traffic to a department's servers from the user community is provided by the routing layer and the firewall. The same is true for intraserver traffic when the servers belong to different departments.

In the introductory paragraphs of this chapter, I stated that this was an engagement that required us to provide less than optimum connectivity between the servers. The inter-department server connectivity is a prime example of the degradation. Consider an example where a server in department B must update customer records on a server in department R (it will be easier to follow the path if you refer to [Figure 7-2](#)). The traffic passes from the B server over the B VLAN to the routing layer, where it is passed to the firewall and then back to the router. The traffic is then passed back through the firewall and through the router again, finally arriving on the R VLAN and at the R server. Ouch, six hops to travel one rack away! The rationale for all the hops is defined in the routing and firewall sections to follow, but for now, I'll just say that this path is a requirement for security purposes and because each department needed to manage its own firewalls.

Routing Layer

As we've just seen, the switching layer is composed of multiple components and inter-connecting links. The routing layer is just as complex, but virtually rather than physically. The router chosen for this engagement was the MX240 3D Universal Edge Router. This router supports modular design and sufficient throughput to accommodate the customer's data center requirements at the time of the engagement and well into the future. The Juniper Networks website provides this description:

MX240 3D Universal Edge Router

Juniper Networks MX240 3D Universal Edge Router design delivers increased port density over traditional Ethernet platforms as well as performance of 960 Gbps throughput, scalability and reliability in a space-efficient package. The MX240 offers fully redundant hardware options that include a redundant Switch Control Board (SCB) and Routing Engines (REs) to increase system availability.

The MX240 supports multiple routing instances. Each of these routing instances is assigned to either a department or the user/Internet portion of the network.

[Figure 7-3](#) shows the virtual configuration of the MX240. The routing instances are virtual routers with the departmental interfaces assigned in the instance. Each instance interfaces to a firewall and the switching layer. The enterprise/Internet routing instance connects to the routers of the enterprise network for connectivity to all users. Interconnection between routing instances is supported by the firewall links.

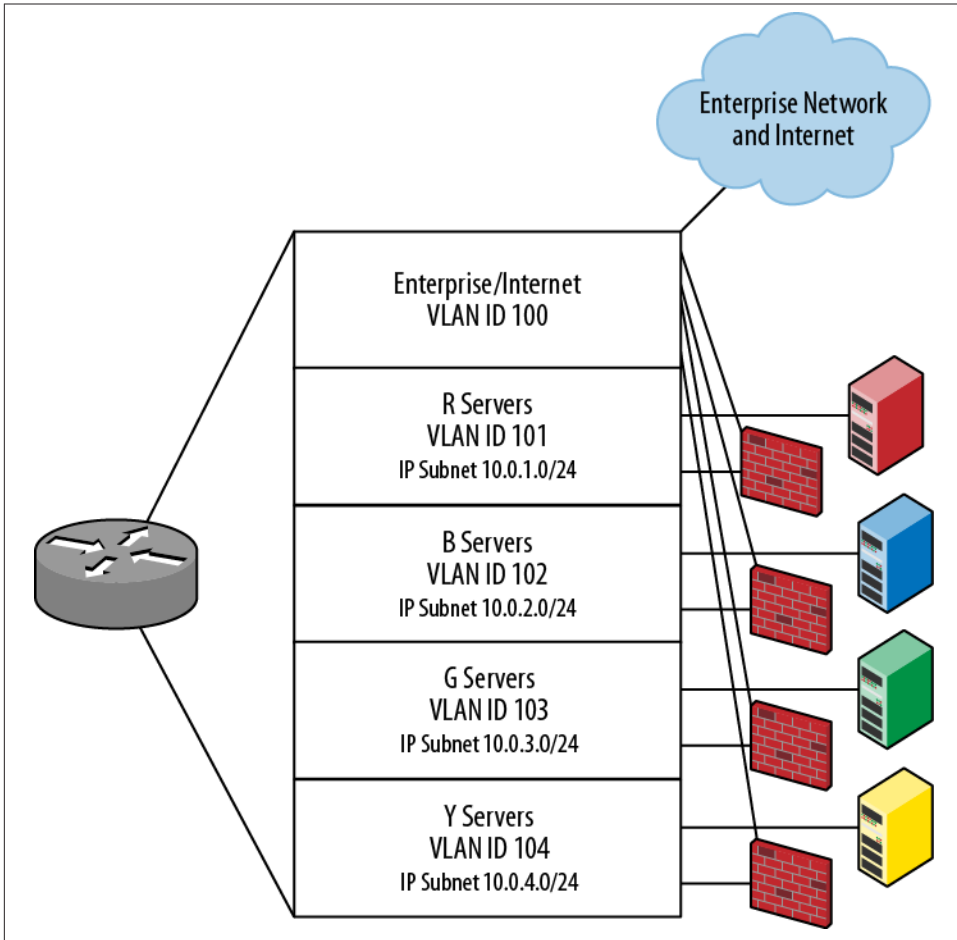


Figure 7-3. Virtual router configuration



Interconnection between routing instances can be performed by a number of mechanisms (for example, tunnel interfaces, static routes, or firewall filters). These allow communications between the routing instances without the expense of an external interface. In this case, the interconnection requirements needed to be firewalled, though, so these interior options were not possible.

Each MX routing instance is, in reality, an individual router. Each has interfaces, a routing table, an instance of routing protocols, and routing options.

The four separate links shown between the routing layer and the switching layer in [Figure 7-2](#) are combined into a single link aggregation group (LAG) and divided into multiple logical channels (by department) with the use of VLAN tags. Each logical channel is associated with a routing instance. This arrangement allows flexibility, scaling, and survivability. The physical links can be terminated on different line cards, additional links can be added, and the addition of or loss of a link will not cause a loss of communications. Adding a department requires nothing more than the configuration of a new VLAN and routing instance—no physical changes are required.



One additional feature of the MX240 that was not used for this engagement (but that adds to the appeal of the use of this router in a data center environment) is its support for the use of a virtual chassis: two MX240s can be interconnected and act as a single expanded router. This doubles the interfaces while increasing the reliability of the connectivity and processing planes of the router.

Firewall Layer

If you remember, the main goal of this engagement was to provide compliance with the PCI edicts, and here the compliance is provided with the firewall. The stateful policies of the Juniper Networks SRX Services Gateway support the rule-based connectivity that forms the basis of compliance. Like the MX240, the SRX supports virtual divisions. In addition to the logical separation of the routing, the virtualization supported in the SRX allows the separation of the administrative and security functions. These virtual separations are referred to as *logical systems*, rather than routing instances (the SRX also supports routing instances, but that did not meet the client's requirement of individual administrative control).

The virtualization of the SRX is shown in [Figure 7-4](#). The single chassis is logically divided into the four departmental firewalls. Each logical system supports interfaces (logical or physical), administrators, security zones, and security policies.

The interconnection between the SRX and the MX240 is provided by a multilink LAG. Up to eight individual links can be interconnected between the devices and bundled into a single link. Like the LAG between the switching layer and the routing layer, this LAG is divided along departmental lines using VLANs. Each VLAN is assigned to a different logical interface on the LAG.

The reliability requirements of the SRX are met with chassis clustering. Similar to a virtual chassis, two SRXs are interconnected and act as a single device. The principal difference between a virtual chassis and a chassis cluster is that the SRX maintains state information on the backup routing engine, so that in the case of a failover, all connections are maintained through the firewall.

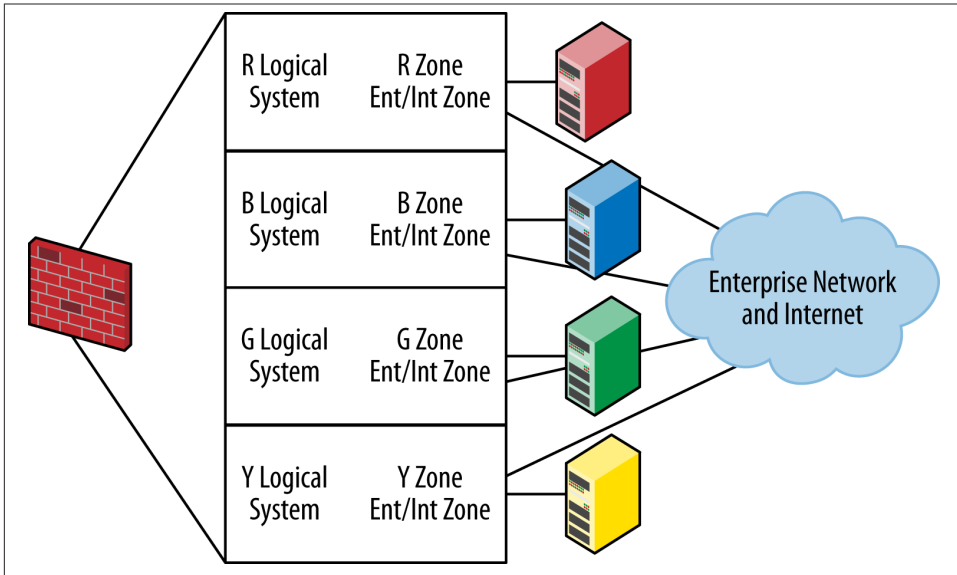


Figure 7-4. Firewall virtualization

For this engagement, a pair of SRX1400s was chosen. The SRX1400 is the smallest of the data center–designed SRXs that offer logical system support. It offered the highest throughput in the smallest package of any comparably priced firewalls at the time, too.

Virtualization

Throughout this engagement, the use of virtualization allowed a minimal suite of equipment to meet the client’s requirements. Virtualization is the wave of the future, and one of the reasons the clan of Juniper warriors is growing. The virtualized elements support the client’s requirements for scaling and reliability, and as the needs of the company change, the virtual aspects of the network can be altered without adding equipment, links, or cards.

Five different virtualization elements were used in this engagement:

- VLANs were used throughout to differentiate and segregate traffic on the switches and on the physical links.
- LAGs were used to combine physical links into a single logical entity, allowing growth and reliability.
- *Virtual chassis* combined the multiple switches (or two MX240s) into a single distributed switch, with redundant switching planes and redundant control planes.

- *Routing instances* allowed a single router to look and act as multiple routers. Routing instances provide ease of deployment, separation of traffic, and flexibility of routing for each instance.
- *Logical systems* allowed the SRX (M-series and T-series devices also support logical systems) to support a separation of administrative functions on a single device.

The combination of these technologies changed the topology of the data center dramatically. [Figure 7-2](#) showed the physical layout of the devices and the physical interconnecting links. [Figure 7-5](#) shows almost the same network when all the virtual elements are added: the single MX looks like five different routers, the SRX looks like four firewalls, and the switches make it look like the servers of the same department are connected to the same switch.

Configurations

The deployment had seven physical devices, yet only three configurations (the four EX4200s are controlled by a single configuration, and there was one configuration for the two SRXs). Only the relevant portions of each of those configurations are discussed here. As stated previously, presenting the actual configurations used in the engagement would reveal too much information about the client and its network, so in their place are generic configurations that perform the same functions. Warriors obey the confidentiality requests of their clients.

EX4200 Configuration

Starting at the bottom of the topology, the EX4200s are stacked and configured as a single device. The stacking (creation of the virtual chassis) can be performed in one of two ways. The first option is to use the default settings of the switches, interconnect them, and let them figure out which is the boss, which is the backup, and which are the line cards. The second method is called *prepositioning* and requires the stack details to be configured. For our system, we wanted the best reliability and the most predictability, so the prepositioning method was used. The configuration required the entry of the serial numbers of the EXs and the identification of the active master, the backup, and the line cards. It looked like this:

```
[edit]
peter@DC-SWITCH# show virtual-chassis | display set
set virtual-chassis preprovisioned
set virtual-chassis member 0 serial-number 1010101010 role
  routing-engine
set virtual-chassis member 1 serial-number 1111111111 role line-card
set virtual-chassis member 2 serial-number 1212121212 role line-card
set virtual-chassis member 3 serial-number 1313131313 role
  routing-engine
```

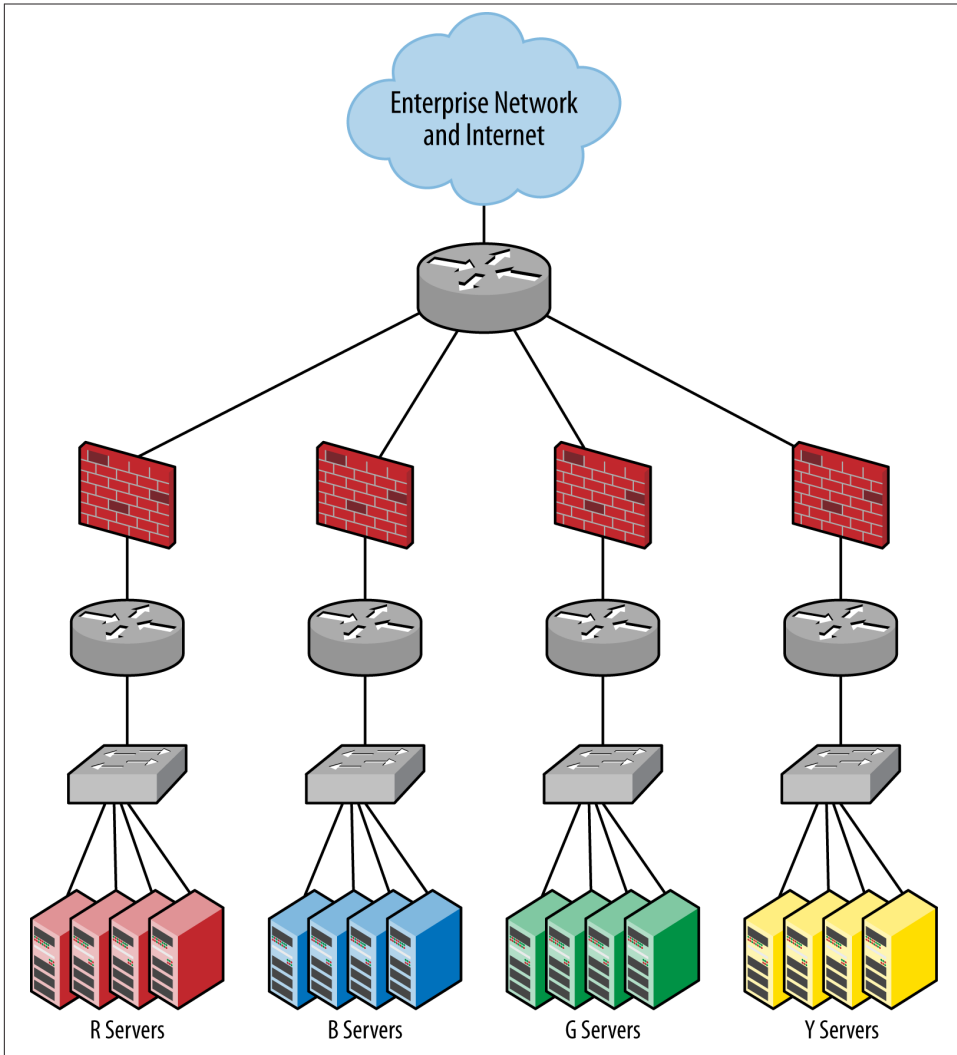


Figure 7-5. Virtual configuration

To form the virtual chassis, the EX4200 uses either purpose-built interconnection cables or 10 Gbps links. For our engagement, the purpose-built cables saved ports and had a lower cost than the 10 Gbps XFPs. The switches are shipped with a .5-meter cable, but 1.5-, 3-, and 5-meter cables can be purchased. Due to the distances between the server racks, the 3-meter and 5-meter cables were used. The interconnection between the rack tops was a braided ring configuration, as shown in [Figure 7-6](#).

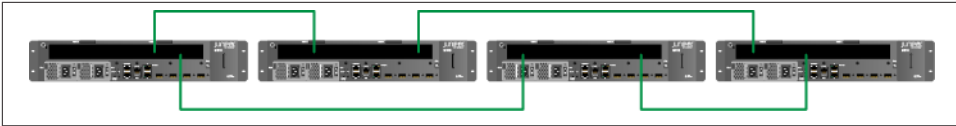


Figure 7-6. EX4200 interconnection cabling

The braided ring configuration required the cables to be connected to each device. This provided failure protection and the lowest possible latency between the devices. The cables are autodetected by the EXs and enable sharing of connectivity and control information between the members of the virtual chassis.

The activation of the EX4200 virtual chassis was a simple sequence. Initially, the master of the stack was configured with the prepositioning configuration. Once the configuration was committed, each other device (backup and line cards) was cabled and powered on. During the boot sequence, the interconnecting cables are detected and the switch joins the virtual chassis. Once the virtual chassis is created, the master routing engine contains the entire configuration, and a connection to the console port on any switch shows a connection to the master routing engine.

The management configuration of the virtual chassis (VC) was identical to that of a non-VC EX, except for the management interfaces. Instead of the normal *me0* interfaces, *vme0* interfaces were configured to perform out-of-band management for the VC. In our system, each of the switches was connected to the OOB management LAN, and the *vme0* address was set for this LAN. The management configuration was:

```
[edit]
peter@DC-SWITCH# show interface vme | display set
set interfaces vme unit 0 family inet address 10.0.0.101/24
```

We set the address of the OOB network with:

```
[edit]
peter@DC-SWITCH# show routing-options | display set
set routing-options static route 0/0 next-hop 10.0.0.1
```

Then we set the default route to send all traffic.

Once the switches were powered on and operational, the virtual switch could be verified using the *show virtual switch* commands. The two that provide the most data are:

```
peter@DC-SWITCH > show virtual-chassis status
Preprovisioned Virtual Chassis
Virtual Chassis ID: 0000.e255.00fe
```

Member ID	Status	Serial No	Model	Mastership Priority	Role	Neighbor List ID	Interface
0 (FPC 0)	Prsnt	101010	ex4200-48p	129	Master*	1	vcp-0
						2	vcp-1
1 (FPC 1)	Prsnt	111111	ex4200-48p	0	Linecard	3	vcp-0
						0	vcp-1

```

2 (FPC 2)  Prsnt  121212  ex4200-48p      0  Linecard  0  vcp-0
                                           3  vcp-1
3 (FPC 5)  Prsnt  131313  ex4200-48p     129  Backup    1  vcp-0
                                           2  vcp-1

```

```
peter@DC-SWITCH > show virtual-chassis vc-port all-members
```

```

fpc0:
-----
Interface      Type      Status
or
PIC / Port
vcp-0          Dedicated Up
vcp-1          Dedicated Up

fpc1:
-----
Interface      Type      Status
or
PIC / Port
vcp-0          Dedicated Up
vcp-1          Dedicated Up

fpc2:
-----
Interface      Type      Status
or
PIC / Port
vcp-0          Dedicated Up
vcp-1          Dedicated Up

fpc3:
-----
Interface      Type      Status
or
PIC / Port
vcp-0          Dedicated Up
vcp-1          Dedicated Up

```

Once the stack was created and verified, the real meat of the configuration was entered—the VLAN configuration. Each server was connected to a port on the EX4200s. That port was placed in a VLAN. Each VLAN was associated with a VLAN identifier, and all servers in the same VLAN were able to communicate with one another without the need of a router.



Inter-VLAN connections can be created on the EX, or the MX, for that matter. This would allow servers to communicate at a low level in the communications hierarchy, but it would also invalidate the entire purpose of this exercise: the securing of the departmental servers from one other.

The VLAN configuration on the EX4200 was:

```
[edit]
peter@DC-SWITCH# show vlans | display set
set vlans R vlan-id 101 interface ge-0/0/1.0
set vlans R vlan-id 101 interface ge-1/0/1.0
set vlans R vlan-id 101 interface ge-2/0/1.0
set vlans R vlan-id 101 interface ge-3/0/1.0
set vlans B vlan-id 102 interface ge-0/0/2.0
set vlans B vlan-id 102 interface ge-1/0/2.0
set vlans B vlan-id 102 interface ge-2/0/2.0
set vlans B vlan-id 102 interface ge-3/0/2.0
set vlans G vlan-id 103 interface ge-0/0/3.0
set vlans G vlan-id 103 interface ge-1/0/3.0
set vlans G vlan-id 103 interface ge-2/0/3.0
set vlans G vlan-id 103 interface ge-3/0/3.0
set vlans Y vlan-id 104 interface ge-0/0/4.0
set vlans Y vlan-id 104 interface ge-1/0/4.0
set vlans Y vlan-id 104 interface ge-2/0/4.0
set vlans Y vlan-id 104 interface ge-3/0/4.0
```

There are a couple of things to note in this configuration. Each stanza associates an interface with a VLAN. We could also have associated the VLANs with the interfaces—the EX4200 can be configured either way. The other item of interest is that for the EX4200s, the interface identifiers are in the *type-switch#/pic/port* format (unlike for the MX, where they are in the normal *type-fpc/pic/port* format). The initial digit identifies the switch location in the virtual chassis. The *pic* is one of two identifiers for the EX4200: 0 indicates the built-in ports and 1 indicates the uplink module (*fpc* = flexible PIC concentrator, *pic* = physical interface card). The port assignments are nice and logical and match the VLAN IDs—it would be nice to say that this was the way that they were assigned at the client's, but that was never the case.



According to Juniper data center design best practices, each server should be connected to two virtual chassis; this way, in the event that one of the VCs is out of service, the other can still connect to the servers. However, the added cost of this redundancy was not justified in this environment.

On the router side of the EX4200s, the link aggregation group (LAG) was used for interconnectivity. The LAG was composed of four links, one from each EX4200 to the MX240. LAGs on Juniper Networks devices support the link aggregation control protocol (LACP) for dynamic bandwidth allocation and failover. The links were set to operate with a single link active, and the ports' speeds were set to 10 Gbps. This was overkill in terms of bandwidth if all ports were active, but close to the bandwidth requirements in the event of failures. The configuration for the LAG was:


```
[edit]
peter@DC-SWITCH# show chassis | display set
set chassis aggregated-devices ethernet device-count 4
[edit]
peter@DC-SWITCH# show interfaces | match ae0 | display set
set interfaces ae0 aggregated-ether-options minimum-links 1
set interfaces ae0 aggregated-ether-options link-speed 10g
set interfaces ae0 aggregated-ether-options lacp active
set interfaces xe-0/1/0 ether-options 802.3ad ae0
set interfaces xe-1/1/0 ether-options 802.3ad ae0
set interfaces xe-2/1/0 ether-options 802.3ad ae0
set interfaces xe-3/1/0 ether-options 802.3ad ae0
```

The LAG configuration starts with the configuration of how many LAGs are going to be used on the chassis. This number has to be the same as or higher than the number of deployed groups. In our case, four was a nice number, allowing for some growth.



Setting the device count to a high value does not cause issues in performance, but it does make the *show interface terse* command no longer terse, because every *aeX* interface that is identified in the chassis command is put in the list. For example, if only one aggregate device is used but you provide a device count of 10, you'll see output like this:

```
peter@DC-SWITCH > show interfaces terse
Interface      Admin Link Proto  Local  Remote
ae0             up    up
ae1             down  down
ae2             down  down
...
ae10            down  down
```

The interfaces associated with the LAG were the 10 Gbps interfaces, one from each switch of the virtual chassis. The uplink module from each switch was used for the 10 Gbps interfaces.

Once the LAGs were identified, they were configured to carry the switched traffic. The LAG was set as a trunk port carrying tagged traffic between the EX4200 and the MX240. The configuration that was used for the LAG trunk was:

```
[edit]
peter@DC-SWITCH# show interfaces ae0 unit 0 | display set
set interfaces ae0 unit 0 family ethernet-switching port-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members
[ 101-104 ]
```

The commands signify that the LAG 0 interface is an 802.1q trunk and that the VLAN identifiers of 101 through 104 are to use this trunk. On the opposite side of the interface, each of these VLANs will be treated as a different virtual router (see the next section).

MX240 Configuration

The configuration of the MX240 was only special in terms of the routing mechanisms that were used to divide the traffic to and from the servers. Five virtual router routing instances provided the transport between the EX4200s and the SRX firewall cluster, as shown in [Figure 7-3](#). The redundant routing engines (REs) of the MX240 required the use of groups for management, as well as a few chassis commands to assure nonstop routing in the case of a failover between routing engines.

The system commands to enable nonstop routing and graceful switchover were entered, as well as the commit synchronize command. These assure that the REs take over for one another and that when a commit is made, both routing engines are updated:

```
[edit]
peter@DC-ROUTER# show chassis redundancy | display set
set chassis redundancy graceful-switchover
```

```
[edit]
peter@DC-ROUTER# show routing-options | display set
set routing-options nonstop-routing
```

```
[edit]
peter@DC-ROUTER# show system | display set
set system commit synchronize
```

The management configuration was assigned in a group and applied in the body of the configuration. The reference for the groups was assigned to the appropriate routing engine when they were configured. The management groups contained the *fxp0* addressing, the RE-specific hostname, and the backup router definition for each routing engine:

```
[edit]
peter@DC-ROUTER-re0# show groups | display set
set groups re0 system host-name DC-ROUTER-re0
set groups re0 system backup-router 10.0.0.1 destination 0.0.0.0/0
set groups re0 interfaces fxp0 description "This is the fxp0
  interface for RE0"
set groups re0 interfaces fxp0 unit 0 family inet address
  10.0.0.102/24
set groups re1 system host-name DC-ROUTER-re1
set groups re1 system backup-router 10.0.0.1 destination 0.0.0.0/0
set groups re1 interfaces fxp0 description "This is the fxp0
  interface for RE1"
set groups re1 interfaces fxp0 unit 0 family inet address
  10.0.0.103/24

[edit]
peter@DC-ROUTER-re0# show apply-groups | display set
set apply-groups [re0 re1]
```

The remainder of the management configuration was stock: we set the DNS, NTP, authentication, and default route features. All of these were set in the default routing instance. The default routing instance was used solely for management traffic. All server traffic was handled by the other routing instances.

There were four server routing instances that each contained a pair of interfaces (servers and the firewall), and one enterprise/Internet routing instance that contained five interfaces (the four firewall public interfaces and the user network interface). While the same result could have been accomplished in a single routing instance with a bit more complication in the routing, the brute-force method of routing instances appealed to the security guys in the tribe.

The interfaces were initially defined and then assigned to the routing instances. The interface configuration to the EX4200 was a four-link LAG that contained the four VLANs from the servers. The LAG configuration was split between two modular port concentrators (MPCs) for reliability and looked identical to the EX4200 configuration (how I love a single operating system):

```
[edit]
peter@DC-ROUTER-re0# show chassis | display set
set chassis aggregated-devices ethernet device-count 4
[edit]
peter@DC-ROUTER-re0# show interfaces | match ae0 | display set
set interfaces ae0 aggregated-ether-options minimum-links 1
set interfaces ae0 aggregated-ether-options link-speed 10g
set interfaces ae0 aggregated-ether-options lacp active
set interfaces xe-0/1/0 ether-options 802.3ad ae0
set interfaces xe-0/1/1 ether-options 802.3ad ae0
set interfaces xe-1/1/0 ether-options 802.3ad ae0
set interfaces xe-1/1/1 ether-options 802.3ad ae0
```

The server-facing routing instances' interfaces were the default gateways for the servers. The LAG's logical interfaces held the default gateway addresses. The configuration for the interfaces accepted the incoming tagged traffic and associated it with a logical interface:

```
[edit]
peter@DC-ROUTER-re0# show interfaces ae0 | display set
set interfaces ae0 aggregated-ether-options minimum-links 1
set interfaces ae0 aggregated-ether-options link-speed 10g
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 vlan-tagging
set interfaces ae0 unit 0 description "interface to R servers"
set interfaces ae0 unit 0 family inet address 10.0.1.1/24
set interfaces ae0 unit 0 vlan-id 101
set interfaces ae0 unit 1 description "interface to B servers"
set interfaces ae0 unit 1 family inet address 10.0.2.1/24
set interfaces ae0 unit 1 vlan-id 102
set interfaces ae0 unit 2 description "interface to G servers"
set interfaces ae0 unit 2 family inet address 10.0.3.1/24
```

```

set interfaces ae0 unit 2 vlan-id 103
set interfaces ae0 unit 3 description "interface to Y servers"
set interfaces ae0 unit 3 family inet address 10.0.4.1/24
set interfaces ae0 unit 3 vlan-id 104

```

The interface to the firewall was also divided between logical interfaces on a LAG. The configuration was identical to that of the LAG interface to the EX4200, except for the physical location of the component links to the LAG and the number of logical interfaces supported. Four LAG links were supported between the MX240 and the SRX1400. This allowed the LAG to be split between the two members of the SRX cluster (as discussed in the next section). The LAG between the MX and the SRX was divided into five logical interfaces: four for traffic on the server side of the firewall (trusted), and one for traffic on the enterprise side of the firewall (untrusted). From a physical perspective, the interfaces between the MX and the SRX could look to be three “in” interfaces and one “out” interface, but for better performance and reliability there was no need to split these “directions” into physical entities, when they could be created using VLANs on a single LAG. Here’s the configuration:

```

[edit]
peter@DC-ROUTER-re0# show interfaces | match ae1 | display set
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options link-speed 10g
set interfaces ae1 aggregated-ether-options lacp active
set interfaces xe-0/0/0 ether-options 802.3ad ae1
set interfaces xe-0/0/1 ether-options 802.3ad ae1
set interfaces xe-1/0/0 ether-options 802.3ad ae1
set interfaces xe-1/0/1 ether-options 802.3ad ae1

```

We bumped the VLAN identifiers on the MX-SRX LAG so as not to cause confusion between the two LAGs:

```

[edit]
peter@DC-ROUTER-re0# show interfaces ae1 | display set
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options link-speed 10g
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 vlan-tagging
set interfaces ae1 unit 0 description "interface to R ZONE"
set interfaces ae1 unit 0 family inet address 10.1.1.1/24
set interfaces ae1 unit 0 vlan-id 201
set interfaces ae1 unit 1 description "interface to B ZONE"
set interfaces ae1 unit 1 family inet address 10.1.2.1/24
set interfaces ae1 unit 1 vlan-id 202
set interfaces ae1 unit 2 description "interface to G ZONE"
set interfaces ae1 unit 2 family inet address 10.1.3.1/24
set interfaces ae1 unit 2 vlan-id 203
set interfaces ae1 unit 3 description "interface to Y ZONE"
set interfaces ae1 unit 3 family inet address 10.1.4.1/24
set interfaces ae1 unit 3 vlan-id 204
set interfaces ae1 unit 4 description "interface from ENT/INT-R ZONE"
set interfaces ae1 unit 4 family inet address 10.1.5.1/24

```

```

set interfaces ae1 unit 4 vlan-id 205
set interfaces ae1 unit 5 description "interface from ENT/INT-B ZONE"
set interfaces ae1 unit 5 family inet address 10.1.6.1/24
set interfaces ae1 unit 5 vlan-id 206
set interfaces ae1 unit 6 description "interface from ENT/INT-G ZONE"
set interfaces ae1 unit 6 family inet address 10.1.7.1/24
set interfaces ae1 unit 6 vlan-id 207
set interfaces ae1 unit 7 description "interface from ENT/INT-Y ZONE"
set interfaces ae1 unit 7 family inet address 10.1.8.1/24
set interfaces ae1 unit 7 vlan-id 208

```

The final interface was the enterprise/Internet interface that connected to the user community. This was a 1 Gigabit interface to an existing router. It seemed that this interface was going to be a bottleneck for traffic, but we were assured that as the level of traffic increased, additional routing resources would be made available. Its configuration looked like this:

```

[edit]
peter@DC-ROUTER-re0# show interfaces ge-3/0/0 | display set
set interfaces ge-3/0/0 unit 0 description "interface to ENT/INT"
set interfaces ge-3/0/0 unit 0 family inet address 10.2.2.1/24

```

The routing on the MX240 was a combination of default routes, static routes, and OSPF routes. The management (default) routing instance used a static route to the management network (see the configuration statements above). The server routing instances used a static route and a default route to and from the SRX. Finally, the enterprise/Internet routing instance was part of the OSPF backbone area. The routing is configured as part of the routing instances in Junos, so the full routing instance configurations are defined below to include the routing elements.

The routing instances used for this engagement were the virtual router routing instances. These hold interfaces and routing information, and each creates a separate routing table for the traffic touching that instance. In the design, five routing instances were configured, one each for the servers and one for the enterprise/Internet:

```

[edit]
peter@DC-ROUTER-re0# show routing-instances | display set
set routing-instances R description "Server R VR"
set routing-instances R instance-type virtual-router
set routing-instances R interface ae0 unit 0
set routing-instances R interface ae1 unit 0
set routing-instances R routing-options static route 0.0.0.0/0
  next-hop 10.1.1.2
set routing-instances B description "Server B VR"
set routing-instances B instance-type virtual-router
set routing-instances B interface ae0 unit 1
set routing-instances B interface ae1 unit 1
set routing-instances B routing-options static route 0.0.0.0/0
  next-hop 10.1.2.2
set routing-instances G description "Server G VR"
set routing-instances G instance-type virtual-router

```

```

set routing-instances G interface ae0 unit 2
set routing-instances G interface ae1 unit 2
set routing-instances G routing-options static route 0.0.0.0/0
  next-hop 10.1.3.2
set routing-instances Y description "Server Y VR"
set routing-instances Y instance-type virtual-router
set routing-instances Y interface ae0 unit 3
set routing-instances Y interface ae1 unit 3
set routing-instances Y routing-options static route 0.0.0.0/0
  next-hop 10.1.4.2
set routing-instances ENT-INT description "Enterprise/Internet VR"
set routing-instances ENT-INT instance-type virtual-router
set routing-instances ENT-INT interface ge-0/0/3 unit 0
set routing-instances ENT-INT interface ae1 unit 4
set routing-instances ENT-INT interface ae1 unit 5
set routing-instances ENT-INT interface ae1 unit 6
set routing-instances ENT-INT interface ae1 unit 7
set routing-instances ENT-INT protocol ospf export static
set routing-instances ENT-INT protocol ospf area 0
set routing-instances ENT-INT protocol ospf area 0 interface
  ge-0/0/3.0
set routing-instances ENT-INT protocol ospf area 0 interface ae1.4
  passive
set routing-instances ENT-INT protocol ospf area 0 interface ae1.5
  passive
set routing-instances ENT-INT protocol ospf area 0 interface ae1.6
  passive
set routing-instances ENT-INT protocol ospf area 0 interface ae1.7
  passive
set routing-instances ENT-INT routing-options static route
  10.0.1.0/24 next-hop 10.1.5.2
set routing-instances ENT-INT routing-options static route
  10.0.2.0/24 next-hop 10.1.6.2
set routing-instances ENT-INT routing-options static route
  10.0.3.0/24 next-hop 10.1.7.2
set routing-instances ENT-INT routing-options static route
  10.0.4.0/24 next-hop 10.1.8.2
set routing-instances ENT-INT routing-options static route
  10.1.1.0/24 next-hop 10.1.5.2
set routing-instances ENT-INT routing-options static route
  10.1.2.0/24 next-hop 10.1.6.2
set routing-instances ENT-INT routing-options static route
  10.1.3.0/24 next-hop 10.1.7.2
set routing-instances ENT-INT routing-options static route
  10.1.4.0/24 next-hop 10.1.8.2

```

The only other part of the configuration was the redistribution of the static routes into the OSPF area. The enterprise/Internet routing instance had the export policy referenced, but the actual policy is a global configuration and not part of the routing instance. The policy is:

```
[edit]
peter@DC-ROUTER-re0# show policy-options | display set
set policy-options policy-statement static term static from protocol
  static
set policy-options policy-statement static term static from
  route-filter 10/8 orlonger
set policy-options policy-statement static term static then accept
set policy-options policy-statement static term other then reject
```

The policy is very simple: if the routes are statically configured and they are in the 10.0.0.0 address range, they are exported to the OSPF database for inclusion in link state calculations. These routes will appear as external OSPF routes in the rest of the network.

Once these configurations were entered and committed (after all the little issues had been fixed and the fat-fingered entries found), the routing instances could be seen in the routing table, as shown in the output of the *show route summary* command:

```
peter@DC-ROUTER-re0> show route summary
inet.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
    Direct:    4 routes,    4 active
    Local:     3 routes,    3 active
    Static:    1 routes,    1 active

B.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
    Direct:    2 routes,    2 active
    Local:     2 routes,    2 active
    Static:    1 routes,    1 active

R.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
    Direct:    2 routes,    2 active
    Local:     2 routes,    2 active
    Static:    1 routes,    1 active

G.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
    Direct:    2 routes,    2 active
    Local:     2 routes,    2 active
    Static:    1 routes,    1 active

Y.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
    Direct:    2 routes,    2 active
    Local:     2 routes,    2 active
    Static:    1 routes,    1 active

ENT-INT.inet.0: 26 destinations, 26 routes (39 active ...)
    Direct:    8 routes,    8 active
    Local:     8 routes,    8 active
    OSPF:      2 routes,    2 active
    Static:    8 routes,    8 active
```

This command shows the tables and the protocols that are active in each routing instance. Each routing instance is indicated by a `<NAME>.inet.0` entry. The ENT-INT routing instance showed that OSPF routes were learned from the backbone, as well as a default route that eventually led to the Internet.

Any time static routes are redistributed to a dynamic routing protocol, care must be taken to assure that default routes are not inadvertently introduced into the network in a manner that would cause them to blackhole traffic. In our policy, we purposely added a route filter statement that limited the static routes that were to be redistributed into OSPF.



It was in another engagement that we extended OSPF into an area that was originally statically routed, and then had to spend a full week finding all the static defaults and redistribution points in that network. This is a case of an ounce of prevention worth a pound of cure. We put our pound in there, and then some!

Firewall Configuration

The last device to be configured was also the device that took the most time to become fully functional. Securing an operational system that was not protected in the past is a very difficult thing to do without upsetting users. “I was always able to get this information from that server, and now I can’t even get to the server—how am I going to get my job done? I might as well just go home!” Ever hear that before? How about this one: “You guys are supposed to be fixing my systems. Why am I receiving all these reports of staff not being able to reach their assets?”

To avoid these types of responses, the system was initially deployed in a wide-open manner. The default policy was changed from a deny-all to a permit-all when the system was initially installed. This allowed all traffic to flow through the system.



In the last section of this chapter, the cut-over details of this engagement are described; these include the firewall policy development and flow analysis.

The firewall configuration was one of the first times that this warrior had ever set up logical systems in an SRX platform. The use of virtual systems in the Juniper Networks Netscreen firewalls had been commonplace for the life of these products, but the SRX had just recently gained the logical system capabilities.



The virtual system (VSYS) of Netscreen is the same as a logical system in Junos. The logical system capability has been available in the larger (M/T-series) devices for quite some time, so when this feature was ported to the SRX line, the Junos name (logical system) was used rather than the Netscreen “VSYS.” While they are not exactly the same, they offer the same type of service: a separation of administrative domains in a single device.

For a logical system (like a routing instance), there are configuration items that are performed in the global configuration and those that are set in the logical system configuration. Our architecture has four logical systems (R, B, G, and Y) and one shared logical system (ENT-INT). All logical systems have a trusted side and an untrusted side. All traffic between departments flows through the firewall twice, once through the ingress department’s policies and once through the egress department’s policies. The full flow of traffic is shown in [Figure 7-7](#).

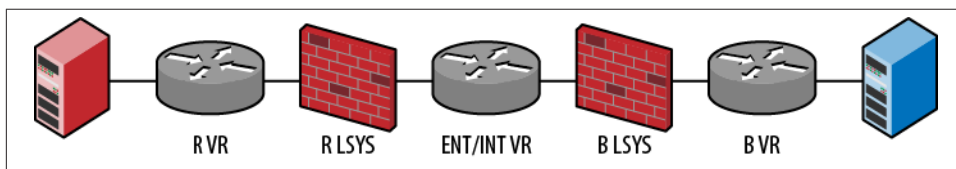


Figure 7-7. Traffic flow between departmental servers

The configuration for the global logical system began by adding the license for logical systems to enable the logical system features. Licenses can be loaded from the Juniper Networks website, from a license file, or from a text file. In our case, the file was loaded with the following command:

```
peter@DC-FIREWALL> request system license add license.keys.1763
```

The file was sent to us from the Juniper Networks rep and was for 10 logical systems. The licenses were verified by the use of the following command:

```
peter@DC-FIREWALL# run show system license status logical-system
root-logical-system
logical system name      license status
root-logical-system      enabled
```

Once the license had been added to both of the SRXs, the cluster was created and normal management functions were added (the configurations for the clustering and management of an SRX can be found in [Chapter 5](#)).

Once these functions were performed, the logical systems were defined and security profiles and administrators were created for each of the logical systems. All of this was done by the default logical system's administrator. First, the four logical systems to be used in the firewall were defined:

```
[edit]
peter@DC-FIREWALL# show logical systems | display set
set logical-systems R
set logical-systems B
set logical-systems G
set logical-systems Y
```

A logical system was defined for each department, to be managed by that department. The next piece to create was a security profile defining which of the resources of the full firewall a department could use. Assets that could have been included in the list below but were not needed include NAT and IDP resources. If the master logical system actually had policies, then it too would be allocated resources.

This profile assured that no single logical system would hog the resources of the firewall:

```
[edit]
peter@DC-FIREWALL# show system security-profile | display set
set system security-profile resources cpu-control
set system security-profile resources cpu-control-target 85
set system security-profile profile-1 cpu reserved 20
set system security-profile profile-1 policy maximum 200
set system security-profile profile-1 policy reserved 50
set system security-profile profile-1 zone maximum 5
set system security-profile profile-1 zone reserved 2
set system security-profile profile-1 flow-session maximum 25000
set system security-profile profile-1 flow-session reserved 20000
set system security-profile profile-1 logical-system R
set system security-profile profile-1 logical-system B
set system security-profile profile-1 logical-system G
set system security-profile profile-1 logical-system Y
```

Once the security profiles had been created, administrators for the logical systems were created. These are the logins that would be used by the department staff to update the firewall policies in their respective logical systems. The users are each affiliated with a class that is affiliated with a logical system and permissions on that logical system:

```
[edit]
peter@DC-FIREWALL# show system login | display set
set system login user peter class super-user
set system login user peter authentication encrypted-password
"1$76389fRljED32m/"
set system login class R-class logical-system R
set system login class R-class permissions all
set system login user admin-r class R-class
set system login user admin-r authentication encrypted-password
"$1$VYfdRheIa90iS/"
set system login class B-class logical-system B
```

```

set system login class B-class permissions all
set system login user admin-b class B-class
set system login user admin-b authentication encrypted-password
"$1$VYfdRheG4JNKa90iS/"
set system login class G-class logical-system G
set system login class G-class permissions all
set system login user admin-g class G-class
set system login user admin-g authentication encrypted-password
"$1$VYfdRG4JNKa90iS/"
set system login class Y-class logical-system Y
set system login class Y-class permissions all
set system login user admin-y class Y-class
set system login user admin-y authentication encrypted-password
"$1$VYfdRhRG4JNKa90iS/"

```

Each of the administrators was associated with a logical system class of service that defined the commands allowed. In this example, the administrators were set with the logical system names as part of the usernames. This is for illustration purposes only; the usernames could be any name (e.g., Sam, Joe, Sally, or Peter). The user class is what associates the user to the logical system.

The next part of the configurations is the definition of the interfaces to be used. Interfaces and addressing are configured at the default system and referenced in the individual logical systems. The interfaces are a mirror of the MX240's interfaces, except that the interfaces on the SRX1400 are split between chassis members. A LAG was created in the default logical instance, and the logical interfaces were created in the individual logical systems:

```

[edit]
peter@DC-FIREWALL# show interfaces | match ae0 | display set
set interfaces ae0 vlan-tagging
set interfaces ae0 aggregated-ether-options minimum-links 1
set interfaces ae0 aggregated-ether-options link-speed 10g
set interfaces ae0 aggregated-ether-options lacp active
set interfaces xe-0/0/10 ether-options 802.3ad ae0
set interfaces xe-0/0/11 ether-options 802.3ad ae0
set interfaces xe-1/0/10 ether-options 802.3ad ae0
set interfaces xe-1/0/11 ether-options 802.3ad ae0

```

At this point, we could have deployed the firewall and told each department it was time for them to step in and set up their firewalls. But rather than fighting the firestorm of outages that would have caused, we set up a set of default policies for each department.

The first policy was to change the default security policy to a permit-all rather than the normal deny-all. The purpose of this was to allow traffic to flow as normal until the true traffic patterns were known.

To configure a logical system, we logged in as the logical system administrator and were connected to that logical system. All the logical systems were initially configured identically. As each was analyzed, each was modified to suit the traffic allowed for it. The configuration below is for the R logical system. The same commands were entered for the B, G, and Y logical systems:

```
login: admin-R
password:

admin-R@DC-FIREWALL:R> config

[edit]
admin-R@DC-FIREWALL:R#
```

The prompt indicates which logical system the administrator is connected to. The normal firewall processes were used to set up the logical systems:

1. The logical interfaces were defined.
2. The interfaces were associated with zones.
3. The host-inbound traffic was defined for the zones.
4. The address books were created for the zones.
5. The policies were written for intrazone traffic.
6. The routing for each logical system was defined.

Each logical system was defined with two zones, the departmental zone (trusted) and the enterprise/Internet zone (untrusted):

```
[edit]
admin-R@DC-FIREWALL:R# show interfaces | display set
set interfaces ae0 unit 0 description "interface to R VR"
set interfaces ae0 unit 0 family inet address 10.1.1.2/24
set interfaces ae0 unit 0 vlan-id 201
set interfaces ae0 unit 4 description "interface from ENT/INT-VR"
set interfaces ae0 unit 4 family inet address 10.1.5.2/24
set interfaces ae0 unit 4 vlan-id 205

[edit]
admin-R@DC-FIREWALL:R# show security zones | display set
set security zone security-zone R
set security zone security-zone R interface ae0.0
set security zone security-zone R host-inbound-traffic
  system-services all
set security zone security-zone ENT-INT
set security zone security-zone ENT-INT interface ae0.4
set security zone security-zone ENT-INT host-inbound-traffic
  system-services ssh

[edit]
```

```

admin-R@DC-FIREWALL:R# show security policies | display set
set security policies default-policy permit-all
set security policy from-zone R to-zone ENT-INT policy permit-all
  match source-address any
set security policy from-zone R to-zone ENT-INT policy permit-all
  match destination-address any
set security policy from-zone R to-zone ENT-INT policy permit-all
  match applications any
set security policy from-zone R to-zone ENT-INT policy permit-all
  then permit
set security policy from-zone R to-zone ENT-INT policy permit-all
  then count
set security policy from-zone R to-zone ENT-INT policy permit-all
  then log session-init
set security policy from-zone ENT-INT to-zone R policy permit-all
  match source address any
set security policy from-zone ENT-INT to-zone R policy permit-all
  match destination-address any
set security policy from-zone ENT-INT to-zone R policy permit-all
  match applications any
set security policy from-zone ENT-INT to-zone R policy permit-all
  then permit
set security policy from-zone ENT-INT to-zone R policy permit-all
  then count
set security policy from-zone ENT-INT to-zone R policy permit-all
  then log session -init

```

The above security configurations were repeated for each department, with the only difference being the interfaces associated with each logical system.

The final part of the logical system configuration was the routing for each department. The routing in the firewall was very simple—a default route was pointing out the ENT-INT VR, and the server address prefixes were pointed to the R VR. The static routes were:

```

[edit]
admin-R@DC-FIREWALL:R# show routing-options | display set
set routing-options static route 10.0.1.0/24 next-hop 10.1.1.1
set routing-options static route 0.0.0.0/0 next-hop 10.1.5.1

```

PCI compliance has two main components: the restriction of users' access to the servers with customer information, and the recording of changes to any security policies and the logging of any attempts to access the servers. The access logging was initiated by the security policies, while the configuration logging was part of the syslog setup. The logging for configuration changes was best accomplished by archiving the configuration each time a commit was performed and logging the interactive commands. The configuration entries were:

```

[edit]
admin-R@DC-FIREWALL:R# show system archival configuration | display set

```

```
set system archival configuration archive-sites
  scp://peter:peter123@10.10.10.12/syslog/configuration/
set system archival configuration transfer-interval 1440;
set system archival configuration transfer-on-commit;
```

This configuration sent a secure file to the server located at 10.10.10.12 whenever a commit was performed on the SRX. The actual interactive commands were logged:

```
[edit]
admin-R@DC-FIREWALL:R# show system syslog | display set
set system syslog host 10.10.10.12 interactive-commands info
```

The final part of the logging was the security logging for the activity to the servers. The permitted traffic was logged by the matching policy, and the final deny policy was logged as well. The security logging was performed in the security stanza:

```
[edit]
admin-R@DC-FIREWALL:R# show security log | display set
set security log format sd-syslog
set security log event-rate 500
set security log source-address 10.1.1.1
set security log stream LOGGING-SVR severity debug
set security log stream LOGGING-SVR host 10.10.10.12
```

This configuration sent the policy logs to the syslog server in the structured format at a rate of 500 messages per minute. The messages can be sent from the routing engine or the packet-forwarding engine (PFE); the event-mode operation routes the traffic from the routing engine, and the stream mode sends the logs directly from the PFE. For larger firewalls, it is better to send the logs directly from the PFE.

Once the firewalls were clustered and configured, the system was interconnected (switches, MX240, and SRX1400s), connectivity was verified, and we were ready for the deployment.

Deployment

The deployment of this system was an install and swap cut-over. The systems were installed in the data center, the MX was interconnected to the backbone, and the servers were repatched to the new switches. Sounds simple enough, but the pain was in the details.

Initial Connectivity

Once the initial connectivity of the system was verified, it was time to cut the system into the backbone and verify that all components could talk to the world. Prior to doing

this, a few changes had to be performed on the MX. If you remember, the static routes for the servers were redistributed to OSPF at the MX. This redistribution would have cut off service to the servers before we were ready if it remained active. The export policy on the MX was deactivated to prevent this issue.



This is where warriors earn their money. Many an evening has this warrior sat thinking what went wrong with a deployment, what I had learned from an outage, what I could have done differently to avoid the stress and the panic. These sessions are often accompanied by beer and other warriors; we compare notes, laugh at gaffs that could have been avoided, and commiserate about things that were totally unknown prior to the cut-over. It is all about not making the same mistake a second time. Plan ahead and think through the steps beforehand, record everything, and learn.

The remaining address prefixes were new to the network, so there were not going to be problems in that area. We connected the MX to the backbone and verified that the internal routes (10.1/16 prefixes) were being seen in the rest of the network. All looked good for the cut-over. We could ping from the server switches to the MX and from the MX to the server switches, and we could ping from the internal network to the departmental MX routing instances (through the firewall's logical systems).

The Maintenance Window

Sometimes, when all is set and the trigger is pulled, Murphy joins the crew and all goes to hell. Thankfully, Murphy stayed at home that day (or was busy somewhere else), and the actual cut-over went as planned. The new switches were in place, configured, and powered. The servers were repatched from the old switches to the new, and the old switches and the old routers were cut out of the loop.

As the patching was in progress, the export policy on the MX was activated and the routes were learned in the backbone for the servers. The servers' address resolution protocol (ARP) caches were cleared, and connectivity was verified by the server teams.

All in all, connectivity was lost for about 10 minutes. Great work, team!

PCI Compliance

The reason for this engagement was to bring the client into compliance with the PCI regulations. As the system stood, it had the capability of being compliant, but it was a long way from compliance. The road to compliance begins with a definition of who should be able to talk to what servers, and what servers should be able to talk to what other servers.

Each department was required to meet the PCI regulations. We provided them with the tools that were going to be used to meet these regulations: the firewalls, the security policy logs, and the session flows. The firewalls are obvious, the other two not so much:

Security policy logs

This feature of the firewall creates a syslog entry each time a session is created on the firewall that hits a security policy. All security policies in a PCI-complaint system must be logged.

Session flows

The SRX is a flow-based system. Each flow through the system is tracked by the system from the time that it starts to the time that it terminates. The session logs give a snapshot of what traffic is on the machine at that instant in time.

These two features allowed the administrators to bring the firewalls into compliance. The steps to compliance were:

1. The known user communities for a server were recorded as address book entries.
2. The known applications for a server were recorded as default or defined applications.
3. A policy or policies were added to the logical system to permit these users to gain access to the servers using these applications.
4. These policies were placed ahead of the permit-all policy.
5. The logs were checked to see what traffic was still hitting the permit-all policy.
6. The address book entries were updated to include users not initially identified but who had legitimate access to the servers.
7. The applications were updated with the legitimate uses of the servers.
8. These users and applications were used to create additional policies.
9. These policies were again added above the permit-all policy.
10. Steps 5 through 9 were repeated for a couple of weeks of normal operation.
11. The default policy was changed to a deny-all, and the permit-all policy was changed to a deny-all policy with logging still turned on.
12. The security log session flow tables and help desk calls were monitored to determine what additional users or applications had to be added to the policies.

The policy logs provided the records needed to show compliance with the PCI regulations: that the known users were permitted and that all others were being denied was what the regulations were designed to enforce.

Some departments achieved compliance faster than others, and many folks complained that things were not like they were used to. So be it with any restrictive regulation!

Summary

This engagement offered many learning opportunities for this warrior (old warriors, unlike old dogs, can learn new tricks). PCI compliance was one area that was totally new to me, and logical systems in the SRX was another.

Companies are becoming more comfortable with virtual technologies. This was a case where if the client had not been comfortable with such technologies, the equipment suite would have been racks deep. Instead, we were able to install a system that met the client's existing connectivity requirements and scalability requirements for the future, as well as the security requirements of the PCI regulations, by using the virtual technologies and systems available from Juniper Networks. Not bad, eh?

Facilitating Dark Fiber Replacement Using a QFX3500

A common theme in most of these engagements is the continual learning that we network warriors experience. I have had the privilege of being associated with this industry for over three decades. In that time there has never been a period where the technology has been in a steady-state condition for any length of time. The state of the art of telecommunications changes continually, and keeping up with it requires constant education. This engagement once again was a learning experience: it involved a new client vertical industry (drugs), a new transmission technology (Fibre Channel), and a new Juniper Networks device (the QFX3500).

In many areas, Juniper Networks leads the state of the art in telecommunications. They have continually increased speed, increased density, and introduced groundbreaking features and services. In this engagement, the tribe used a new Juniper Networks product to help a customer reduce cost and standardize its data center, while meeting the constraints of some very persnickety applications.

Existing Design

For obvious reasons, I won't reveal the identity of this Eastern Seaboard company; for the purposes of this chapter, we'll call it ACME Drug.

The client runs a drug research facility at a location that is remote from the main data centers. Prior to this engagement, the client's research servers and storage facilities were tied to the main data center via leased-line fiber facilities, for which the company paid by the mile, by the month, through the teeth. When the client's CTO decided to review the technical options for interconnectivity, he found that interconnecting with Ethernet

between the research facility and the data center, rather than the existing dark fiber, would save the company substantial operating expenses. That research led the CTO to call in the warriors to determine what would have to be done to switch between these two technologies.

The warrior tribe for this engagement consisted of the Juniper Networks SE for the client, the lead data center network engineer, and me. We offered a mixed bag of technology expertise, but a solid core of determination and attitude to support this client.

To realize the full savings of the new transmission system, we knew that the impact of the change on the rest of the system would have to be minimized. A forklift solution to communications was not going to be possible in this environment.

Figure 8-1 shows the existing design, where the research facility's servers are interconnected to the storage area network in the data center using a Fibre Channel connection. The dark fiber provides native transport of the Fibre Channel's 4 Gbps signal.

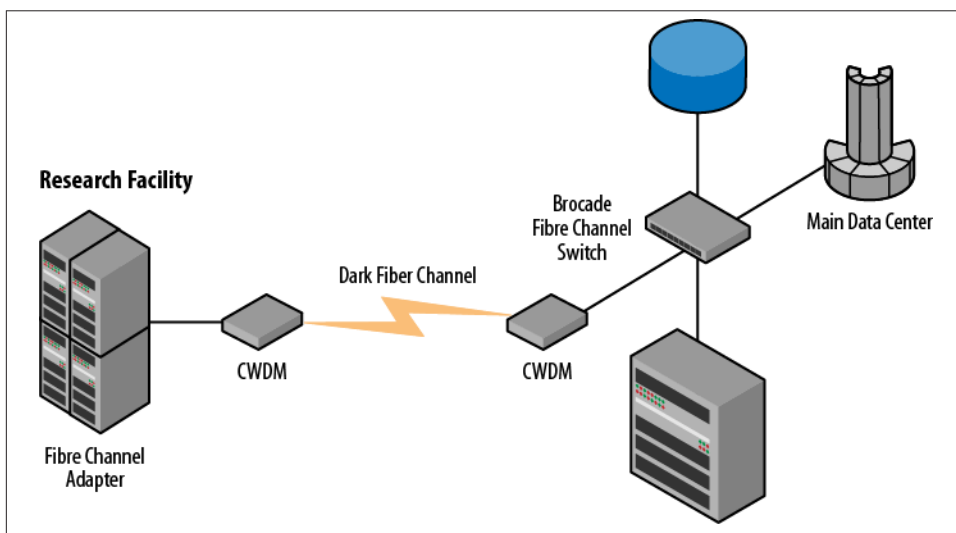


Figure 8-1. Existing research connectivity

To recommend and install a solution, the tribe had to go back to school and learn about a new technology: Fibre Channel. (Fibre Channel isn't really a new technology as far as the state of the art is concerned, but it was new to this tribe!)

Introduction to Fibre Channel

Created in the late 1990s, Fibre Channel supports fast, reliable transmissions between data center elements. The elements are called *initiators* and *targets*, where the initiator originates the blocks of information to be sent to the target. Fibre Channel runs at speeds from 1 Gbps to 16 Gbps, in 2 Gbps increments. The next generation of adapters and switches will run at 32 and 64 Gbps.

Fibre Channel can be compared to LAN protocols in many ways, as shown in [Table 8-1](#). It has a hardware address similar to a media access control (MAC) address: the World Wide Name (WWN), which is composed of an assigned organizationally unique identifier (OUI) and a vendor-specific portion. At the network layer, Fibre Channel uses Fibre Channel Identifiers (FC IDs), a hierarchical addressing scheme that defines domains, zones, and ports for elements.

Table 8-1. Fibre Channel to LAN comparison

	Local area networking	Fibre Channel fabrics
Hardware address	6 octet MAC (OUI/vendor-defined)	8 octet WWN (OUI/vendor-defined)
Network adapter	NIC (access and L2 protocol)	HBA (access, L2, and L3 protocols)
Network address	IP 32 bit (network/host)	FC ID 24 bit (domain/area/port)

Interdomain communications can be controlled by zone filters and an OSPF routing protocol lookalike called *fabric shortest path first* (FSPF). Fibre Channel switches control all communications in a Fibre Channel network. These devices operate on the FC ID and forward blocks of information in a lossless manner.

Devices are connected to a Fibre Channel network (called a *fabric*) by means of a host bus adapter (HBA). These adapters support the protocol stack and addressing in a manner similar to a LAN adapter; the difference is that HBAs typically also support the Fibre Channel Layer 3 protocol. The Fibre Channel fabric is a collection of Fibre Channel switches supporting servers, storage devices, and controllers. The interfaces in a Fibre Channel fabric are designated as either *N ports* (node), *F ports* (fabric), or *E ports* (extended) and are assigned as shown in [Figure 8-2](#).

Our research into Fibre Channel introduced us to a new set of terms and interfaces that did not seem to make a lot of sense, until we started looking at the solutions. All the technical documentation for Fibre Channel and storage area networking uses these identifiers, and the selection of an appropriate set of devices relies on the understanding of each port type and interfacing device.

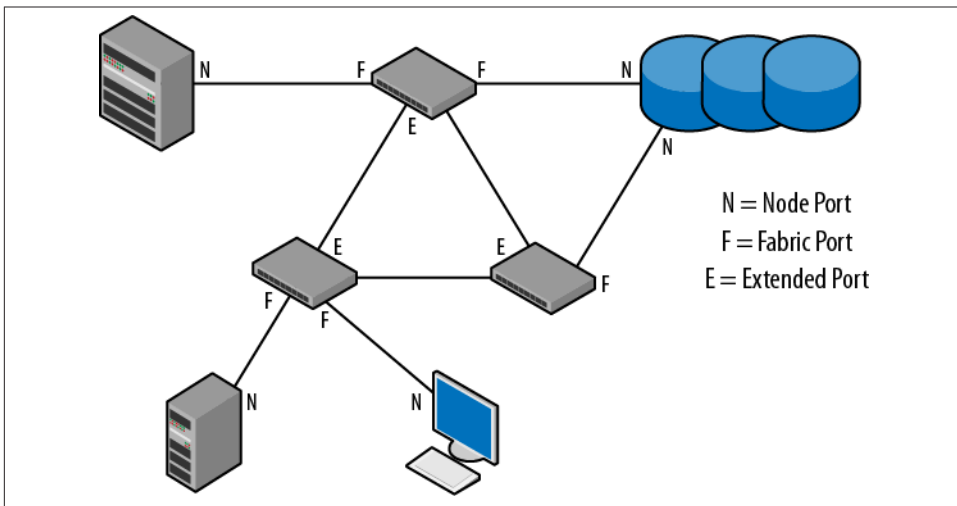


Figure 8-2. Fibre Channel fabric



It seems that it is time for another one of those *Been There, Done That* tee shirt moments. Back in the early 1990s, the telephone companies were introducing a radical new technology called ISDN. The sales and engineering folks were having a hard time remembering all the U/T/R/S reference points: the 2B1Qs, BRIs, PRIs, and the like. The issue was that the tariffs, the equipment manufacturers, and the standards were all written in this new Greek. In order to provision, install, and configure ISDN service, the telephone personnel had to learn the new terminology. Some say that the ultimate demise of the service was due to the difficulty of bridging this technical gap. Still, the fact remains that most new technologies arrive from the standards bodies with new terminology, and generally the equipment manufacturers grasp the new jargon and we have to get new tee shirts to keep up. (BTW, I do have an ISDN tee shirt from SuperComm sitting in a pile somewhere.)

What makes Fibre Channel a perfect fit for data centers and storage systems are its service characteristics, namely:

- High bandwidth
- Lack of congestion
- Scalability
- Low packet overhead

To assure these service characteristics, the transmission medium has to provide in-sequence lossless delivery of frames. The frames have a maximum transmission unit (MTU) of 2,148 octets.

Proposed Design

The transport of Fibre Channel traffic over an Ethernet infrastructure is standardized in a catchy technology called *Fibre Channel over Ethernet* (FCoE). FCoE defines a network adapter for the Fibre Channel devices—a converged network adapter (CNA)—and places the Fibre Channel stack on top of the LAN stack, as shown in [Figure 8-3](#).

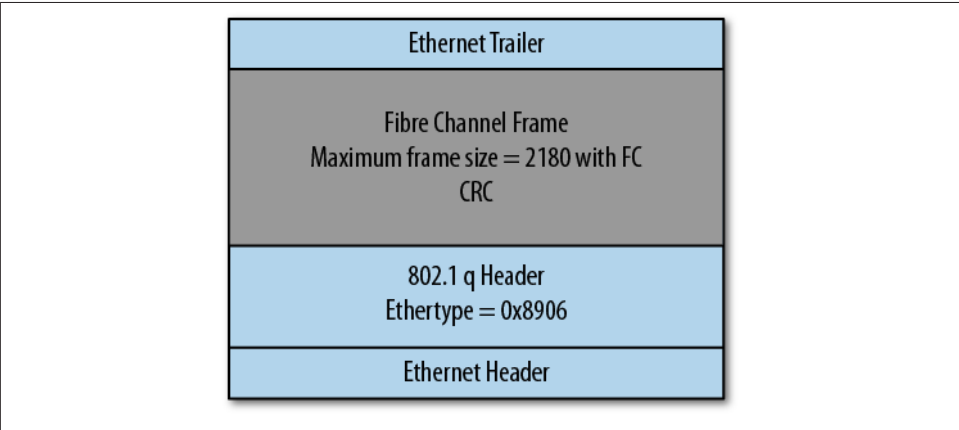


Figure 8-3. FCoE protocol stack

The concept was simple enough: install a new adapter in the local server and send the traffic over the metropolitan-area (metro) Ethernet service, then sort it out on the far end. But of course, the devil is in the details. There were a number of technical hurdles that had to be crossed to assure a successful installation.

Concerns and Resolutions

The major technological concerns were in the areas of naming and network quality.

Naming

In a Fibre Channel network, the applications often use the Fibre Channel naming for identification and name resolution purposes. These names (WWNs), as mentioned earlier, are equivalent to the LAN MAC addresses and are associated with the HBAs.

The solution to this issue involves the use of virtualization of the Fibre Channel names on the new adapters. The converged network adapters emulate HBAs and are referred to as *VN-ports* (virtual node ports).

This approach allows the servers to be migrated to the FCoE network with a minimum of service interruption and does not require a full rewrite of the server applications.

Network quality

The network warriors reading about this adventure are now smiling, wondering, *How is he going to get around the elephant in the room?* Fibre Channel requires a lossless transmission system, while Ethernet offers anything but a reliable and lossless facility. The answer to this dilemma is a rather new development in Ethernet technology called *data center bridging* (DCB).

DCB is a LAN standard that adds reliability features to Ethernet systems. DCB is an add-on to the existing Ethernet stack and operates on Ethernet bridges in the data center. To successfully support Fibre Channel over an Ethernet backbone, DCB is highly recommended. The only catch is that few (read that as *none* that I have encountered) metro Ethernet service providers support DCB at this time. That means that the link between the Ethernet switches will be an unknown as far as buffering and quality are concerned. As a service provider, the service level agreement will have to suffice. The components of DCB that are supported on the Juniper Networks platforms are:

802.1Qbb—Priority Flow Control (PFC)

Enables the use of the Ethernet pause frame in congestion conditions. The prioritization of these pause frames allows up to eight buffers to be controlled on the Ethernet egress interface. Each buffer can be independently monitored and controlled for outgoing rates. The switches use requests and indicators to monitor and limit the individual queues. The switches enable PFC on each of the queues that are enabled for Fibre Channel traffic.

802.1Qaz—Enhanced Transmission Selection (ETS)

In a class of service offering, each traffic type is assigned to a traffic class, and each traffic class is assigned to an egress interface buffer and queue. In a typical system, if bandwidth is allocated to a traffic class/buffer/queue, that bandwidth is not used by other classes on the first pass through the queues. When ETS is enabled, this unused bandwidth can be allocated to other traffic classes for use. This capability allows a smoother utilization of the bandwidth, preventing bursty traffic from clogging the system (and causing congestion that would lead to traffic loss for the Fibre Channel traffic).

802.1Qaz—Data Center Bridge Exchange (DCBX)

DCBX is the protocol defined to transfer the DCB information between data center switches. DCBX is an extension of the link layer discovery protocol (LLDP). DCBX

initially discovers other DCB switches; once they have been identified, DCBX is used to pass DCB parameters and identify parameter mismatch instances. DCBX not only passes the DCB parameters, but is also capable of passing the profile information for Fibre Channel over Ethernet instances.

802.1Qau—Quantized Congestion Notification (QCN)

While not supported on the Juniper Networks QFX3500 at the time of this engagement, QCN is a proactive congestion notification scheme that lets downstream switches know of congestion conditions. Like ECN in ATM networks and FECN and BECN in frame relay networks, QCN is a nicety but is not required for the other aspects of DCB to operate.

The use of the DCB Ethernet enhancement allows Ethernet to perform at a reliability level that is sufficient to keep Fibre Channel happy.

Once these pieces are in place, the full solution to get rid of the dark fiber links can be realized.

Network Upgrade

While the client was anxious to start saving money, they were equally anxious about system uptime and crashing applications. To assure that we maximized the first and minimized the latter, we decided to install the new network in parallel with the existing system and run test applications over the new system.

The solution used the metro Ethernet service offering, the Juniper Networks QFX3500 as a Fibre Channel gateway, and the Juniper Networks EX4500 as a Fibre Channel over Ethernet passthrough device. This combination is shown in [Figure 8-4](#).

The only change to the existing equipment was the addition of another network adapter in the servers at the research facility. These adapters looked to the applications like Fibre Channel adapters, but carried the traffic on industry-standard Ethernet frames. The Ethernet traffic then passed through an EX4500 prior to entering the metro Ethernet service offering. The addition of the EX4500 to the equipment complement not only allowed DCB to operate, but also provided an aggregation point for non-Fibre Channel traffic. The EX4500 used quality of service and DCB to guarantee Fibre Channel performance and allowed other non-FC traffic to use the connection. This added benefit further reduced the cost of connectivity for the client.

On the data center side of the connection, the metro Ethernet connection was terminated in the Juniper Networks QFX3500. In the gateway mode, they accepted FCoE connections on one interface and provided native Fibre Channel on another. A third interface (not shown in [Figure 8-4](#)) supported Ethernet connections for non-FC traffic.

The Fibre Channel reference diagram with the QFX3500 in the network ([Figure 8-5](#)) is useful to explain the operation of this solution.

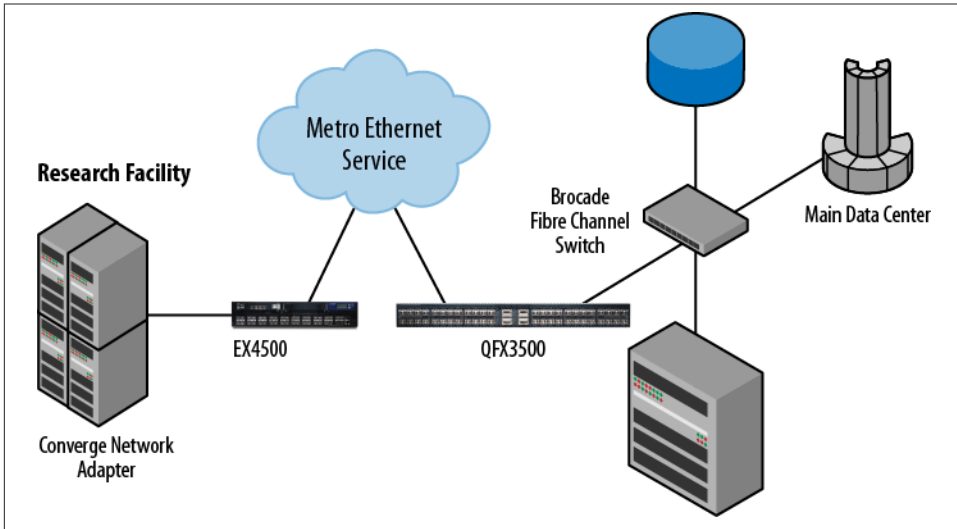


Figure 8-4. Dark fiber replacement topology

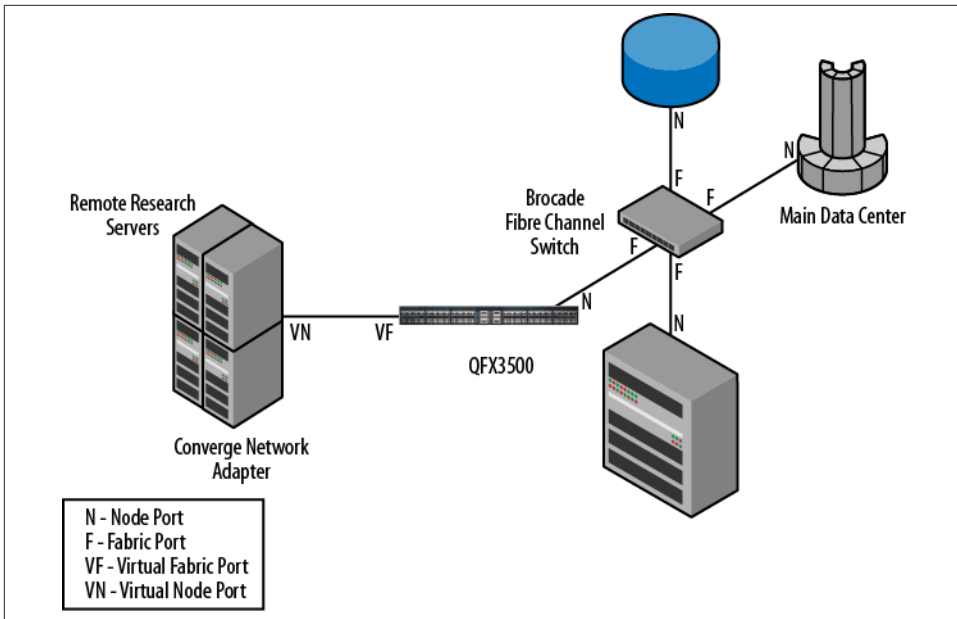
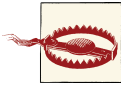


Figure 8-5. Solution reference points

The Ethernet link between the QFX3500 and the remote research servers is shown as a direct connection in this figure. In a sense this is a correct interpretation, as the metro

Ethernet service, the EX4500, and the interconnecting links are all transparent to the operation of the FCoE protocol. The converged network adapter (CNA) supports the convergence of both Fibre Channel and non-Fibre Channel traffic on the same adapter. Management and back office functions share the same adapter. From the point of view of Fibre Channel, the CNA looks just like an HBA, supporting the same addressing at the FC data link layer and the FC network layer. It is referred to as a *virtual node port* because it is creating the addressing and FC functions in firmware rather than hardware.

The other end of the Ethernet link is attached to the QFX3500, which is operating as a converter (gateway) device. The Ethernet interface is considered a virtual fabric interface, as it mimics the fabric interface of a fully functional Fibre Channel switch.



The QFX is not actually a fully functioning Fibre Channel switch. It lacks the Fibre Channel management functions that are embedded in those switches. For the QFX3500 to operate in this environment, a true Fibre Channel switch is required.

The true Fibre Channel interface on the QFX3500 is designed as a node port. It connects to a fabric port of the Fibre Channel switch.

The system operates as follows:

1. An initialization protocol operates between the CNA and the QFX3500 to negotiate the addressing and set up the logical connection between the VN and VF ports.
2. Once initialized, the initialization protocol only maintains a keepalive stance.
3. The Fibre Channel initialization is then performed between the server and the FC switch, with the QFX3500 providing a proxy function.
4. Fibre Channel setup and termination sequences are performed as normal between the FC end stations.

Advantages and Benefits of the Solution

It is a good feeling when an idea is transformed into a solution that meets the client's expectations and offers benefits that were not requested but are welcomed. This engagement offered that great feeling for the warriors catering to this client's request.

The original request was to reduce the wide area network costs between the remote locations and the data center. The WAN cost reduction was achieved by the replacement of the dedicated dark fiber wavelength with a metro Ethernet service. Today, metro Ethernet services are being offered with the lowest cost per megabit of any WAN service.

The added benefits to the solution were:

Converged WAN traffic

The solution supports the simultaneous transmission of Fibre Channel and non-Fibre Channel traffic on the same metro Ethernet connection. DCB assures that the Fibre Channel traffic receives the highest quality of service and lossless transmission. The other traffic rides at lower classes of service.

Converged server functions

Another benefit of the CNA adapter and the use of FCoE is that a single adapter can support FC applications and normal back office-type services (email, DNS, etc.). This allowed the client to retire older servers that were being used just for these purposes.

QFX3500 Fibre Channel Gateway Configurations

While the two devices added to the network were functionally rather simple, the configurations were nonetheless standard Junos configurations. The management functions were installed to allow control of the devices, and security filters were installed to protect the devices. Once the management functions had been activated and the devices could be managed, the Fibre Channel and DCB configurations were added. Finally, the non-FC traffic configurations were added to the devices.

Management Configurations

The management configuration was broken into three parts: the initial system configuration, the management interface configuration, and the protection of that interface. The system configuration set the identity of each device and, as such, was unique per device. The management configuration for the EX4500 transit switch is:

```
system {
    host-name ACME-FC-TRANSIT;
    domain-name acme.drug;
    time-zone America/New_York;
    authentication-order [tacplus password];
}
root-authentication {
    encrypted-password "$1$RsqT0ClQ$Nhsgsretnlz2qaGtD1";
}
tacplus-server {
    6.23.19.23 secret "$9$t7-K7dbLXVYgKvJGUDmf69ApBIAt";
    6.22.65.3 secret "$9$t70-uEcLKMXxdvMagntynDikm ";
}
login {
    message "+==== Acme Research and Development - unauthorized
            access prohibited ====+\n";
    class admin {
        idle-timeout 10;
    }
}
```

```

        permissions all;
    }
    class admin_ro {
        idle-timeout 10;
        permissions view;
    }
    user admin {
        class admin;
        authentication {
            encrypted-password "$1$XvT7cpSTLM28F.";
        }
    }
    user admin_ro {
        uid 2001;
        class admin_ro;
        authentication {
            encrypted-password "$1$KfUog85IHanJm6C2.";
        }
    }
}
services {
    ssh {
        root-login deny;
    }
}
syslog {
    user * {
        any emergency;
    }
    host 6.23.19.24 {
        any any;
        log-prefix JNPR;
    }
    file messages {
        any notice;
        authorization info;
    }
    file interactive-commands {
        interactive-commands any;
    }
}
ntp {
    server 6.23.19.22 prefer;
    server 6.22.6.22;
}

```

The system configurations were straightforward. A TACACS Plus server assigned the administrators, with the local passwords acting as a backup. The services for the device were limited to SSH only; this limited the exposure of the device to hackers and prying eyes.

The syslog entries used the default local files but added a JNPR prefix to any syslog entries sent to the external syslog host. The prefix allowed the syslog messages from the Juniper Networks devices to be sorted based on this prefix.

Finally, two time servers were identified for the Network Time Protocol (NTP), an essential ingredient for tracking faults in the network and troubleshooting cascading issues.

The system management configuration for the QFX3500 is identical to that of the EX4500, except for the host identifier stanza.

The next management function to be configured was the management interface and its addressing. The EX4500 and the QFX3500 both support an out-of-band management port. This allows an external network to control the devices and offers an added level of survivability for troubleshooting and management. However, this client considered OOB management a fancy system better suited to use by carriers than in an enterprise.

Consequently, management support was provided by an in-band VLAN (*vlan-id* 1). Each device was assigned an IP address on this VLAN. The configuration for the interfaces and VLANs looked like this:

```
interfaces {
  xe-0/0/7 {
    mtu 2180;
    unit 0 {
      family ethernet-switching {
        port-mode trunk;
        vlan {
          members all;
        }
      }
    }
  }
}
vlan {
  unit 1 {
    family inet {
      address 6.23.10.20/24;
    }
  }
}
vlangs {
  vlan1 {
    vlan-id 1;
    interface {
      xe-0/0/7.0;
    }
    l3-interface vlan.1;
  }
}
```

The configuration assigned an IP address to a VLAN interface (a logical interface assigned to a virtual LAN, nothing here that you can put a finger on). The VLAN interface was associated with VLAN 1 and was assigned to use interface *ge-0/0/0*, a trunk port that was attached to the metro Ethernet service.

In order for the device to reach a destination, the default gateway had to be defined for the device. The gateway definition was performed by setting a default static route:

```
routing-options {
  static {
    route 0.0.0.0/0 next-hop 6.23.10.1;
  }
}
```

Again, the configuration of the QFX3500 was very similar to that of the EX4500. The differences between the transit switch and the gateway switch were in the addresses and the interfaces used for the management traffic. In the transit switch (EX4500), the management interface was the trunk going to the metro Ethernet service. In the gateway switch (QFX3500), the management interface was the non-FC traffic port.

The final portion of the management configuration was the part of the configuration that protected the devices from hackers. The protection was provided by a series of firewall terms that blocked access to the devices from all but the specified servers and management agents. The firewall filter was applied to the management VLAN interface. The same configuration was used for both devices:

```
firewall {
  family inet {
    filter ACME-MGMT-VLAN-IN {
      term ICMP {
        from {
          protocol icmp;
        }
        then policer ICMP-POLICER;
      }
      term SSH {
        from {
          source-address {
            6.22.10.128/25;
            6.23.10.0/25;
          }
          protocol tcp;
          destination-port ssh;
        }
        then accept;
      }
      term NTP {
        from {
          source-address {
            6.23.19.22/32;
            6.22.6.22/32;
          }
        }
      }
    }
  }
}
```

```

        }
        protocol udp;
        source-port ntp;
    }
    then accept;
}
term SNMP {
    from {
        source-address {
            6.23.10.200/24;
            6.22.69.45/24;
        }
        protocol udp;
        destination-port snmp;
    }
    then accept;
}
term TACACS {
    from {
        source-address {
            6.23.19.23;
            6.22.65.3;
        }
        protocol tcp;
        source-port tacacs;
    }
    then accept;
}
term SSH-REPLY {
    from {
        source-port ssh;
    }
    then accept;
}
term TACACS-REPLY {
    from {
        destination-port tacacs;
    }
    then accept;
}
term NTP-REPLY {
    from {
        protocol udp;
        source-port ntp;
    }
    then accept;
}
term SNMP-TRAP {
    from {
        protocol udp;
        destination-port snmptrap;
    }
}

```

```

    }
    term SNMP-REPLY {
        from {
            protocol udp;
            source-port snmp;
        }
        then accept;
    }
    term SYSLOG {
        from {
            protocol udp;
            destination-port syslog;
        }
        then accept;
    }
    term DENY-ALL {
        then {
            count drop_list;
            discard;
        }
    }
}

}

policer ICMP-POLICER {
    if-exceeding {
        bandwidth-limit 100k;
        burst-size-limit 100k;
    }
    then discard;
}

}

interfaces {
    vlan {
        unit 1 {
            family inet {
                filter {
                    input ACME-MGMT-VLAN-IN;
                }
                address 6.23.10.20/24;
            }
        }
    }
}

}

}
```

The firewall configuration designed for the customer was taken from a number of references. There is a superb **Juniper Networks book** on the topic of hardening Junos devices that provides examples similar to the filter shown above. The filter starts with allowing ICMP traffic up to a 100 kbps rate; the policer referenced is shown at the bottom of the filter. The next half dozen terms allow management access from the various management servers and workstations. The second set of terms restricts return traffic for

these same services. The difference is the port identifiers. All of the terms are used in the INPUT direction on the interface. The final term in the filter is a catch-all for all other traffic—traffic hitting this term is counted for later analysis. This gives a good indicator of the amount of unwanted traffic reaching the device.

Fibre Channel Gateway Interface Configuration

The QFX has 12 interfaces that can be configured as Fibre Channel (*fc*) interfaces, divided into blocks of 6 interfaces at the start (*xe-0/0/0* to *xe-0/0/5*) and end (*xe-0/0/42* to *xe-0/0/27*) of the switch. The interfaces are all or nothing in the sense that setting one interface to Fibre Channel requires that all the interfaces in the block be set to Fibre Channel. This means that even if only a single interface is needed (as in our client's case), the remaining interfaces of the block cannot be used for regular Ethernet.

We used interface *xe-0/0/0* as our Fibre Channel port, so the configuration was:

```
chassis {
  fpc 0 {
    pic 0 {
      fibre-channel {
        port-range 0 5;
      }
    }
  }
}
```

Once the interface block has been defined, the individual interfaces are assigned based on the roles that they play in Fibre Channel. In our case, the interface was defined as a node port. The port modes were assigned under the family *fibre-channel*. There are two other configuration settings for the *fc* interface: the speed of the interface (4 Gbps in our case) and the buffer-to-buffer state change number (the number of receiver-ready messages to exchange when negotiating the buffers for a Fibre Channel link; the recommended setting is 8). The configuration for the interface was:

```
interfaces {
  fc-0/0/0 {
    fibrechannel-options {
      bb-sc-n 8;
      speed 4g;
    }
    unit 0 {
      family fibre-channel {
        port-mode np-port;
      }
    }
  }
}
```

Once the Fibre Channel interface had been defined, it was time to define the VLANs for the switch. Our client required three VLANs to be created: the Fibre Channel VLAN (*vlan-id* 100), the management VLAN (the native VLAN, *vlan-id* 1), and the non-Fibre Channel VLAN (*vlan-id* 50). The non-FC traffic was sent out port *xe-0/0/6* to the enterprise network. The Fibre Channel VLAN was also assigned a port identifier and a Layer 3 interface. These entries allowed the VLAN to be associated with the Fibre Channel fabric. The configurations of the FC and non-FC VLANs looked like this (the management VLAN was created in an earlier section):

```
vlan {
  vlan1 {
    vlan-id 1;
    interface {
      xe-0/0/7.0;
    }
    l3-interface vlan.1;
  }
  FC-VLAN {
    vlan-id 100;
    interface {
      fc-0/0/0.0;
    }
    l3-interface vlan.100;
  }
  Enterprise-VLAN {
    vlan-id 50;
    interface {
      xe-0/0/6.0;
    }
  }
}
```

The trunk interface to the metro Ethernet service was set earlier, but is shown here again—note the MTU setting to support the Fibre Channel messaging within the Ethernet frame (2,180 bytes):

```
interfaces {
  xe-0/0/7 {
    mtu 2180;
    unit 0 {
      family ethernet-switching {
        port-mode trunk;
        vlan {
          members [ vlan1 FC-VLAN Enterprise-VLAN ];
        }
        native-vlan-id 1;
      }
    }
  }
  vlan {
    unit 100 {
```

```

        family fibre-channel {
            port-mode f-port;
        }
    }
}

```

When the definitions of the interfaces and the VLANs were completed, the Fibre Channel switching fabric had to be defined in the switch. This configuration was something totally new for this warrior team: we had worked on switches from X.25 through frame relay and ATM to Ethernet, but we had never had to define the characteristics of the fabric. The configuration steps were rather straightforward, however:

```

fc-fabrics ACME-DRUG {
    fabric-id 100;
    fabric-type proxy;
    interface {
        fc-0/0/0.0;
        vlan.100
    }
}

```

The “configuration” was more of a definition of the interfaces to be interconnected in the Fibre Channel fabric.

Once the fabric was defined, the configuration was complete for the Fibre Channel portion of the switch. The only interface that needed to be added to the configuration was for the non-FC traffic originating at the remote research location. This traffic was carried on VLAN 50 from the remote location and was transferred to the enterprise network on interface *xe-0/0/6*. This interface was an access port, as it connected to an enterprise router. Its configuration was:

```

interfaces {
    xe-0/0/6 {
        unit 0 {
            family ethernet-switching {
                port-mode access;
                vlan {
                    members Enterprise-VPAN;
                }
            }
        }
    }
}

```

DCB Configuration

Data center bridging allowed the Fibre Channel gateway to assure that Fibre Channel over Ethernet traffic was given the necessary priority and flow control settings to bring

Ethernet up to the expectations of the Fibre Channel standard. If the metro Ethernet link was not being shared with enterprise traffic, DCB would not be needed, but since the decision had been made to use the metro Ethernet service for both Fibre Channel traffic and enterprise traffic, DCB was necessary in this case.

The DCB features are negotiated between the switches using a modified link layer discovery protocol (LLDP). The DCB-specific protocol is called DCBX (data center bridging exchange) and has to be configured along with LLDP:

```
protocols {
  lldp {
    interface xe-0/0/6.0;
  }
  dcbx {
    interface xe-0/0/6.0;
  }
}
```

The DCBX negotiated the operation of the DCB features between the Juniper Networks switches. The parameters that were negotiated were set in the class of service stanzas for the Fibre Channel and enterprise traffic. The class of service parameters enabled priority flow control (PFC) and enhanced transmission selection (ETS). These were combined with the normal class of service classifiers, forwarding classes, schedulers, and scheduler maps to form a full class of service capability on these devices.

The initial configuration looked at the classifier of the FCoE traffic. This traffic uses the 802.1 level 3 priority for traffic differentiation and is mapped to a traffic class called *FCOE-TRAFFIC*. The enterprise traffic did not carry an 802.1 priority and was mapped to an *ENTERPRISE-TRAFFIC* class. The configuration is:

```
class-of-service {
  classifiers {
    ieee-802.1 FC-PFC {
      forwarding-class FCoE-TRAFFIC {
        loss-priority low code-points 011;
      }
      forwarding-class ENTERPRISE-TRAFFIC {
        loss-priority low code-points 000;
      }
    }
  }
}
```

The *FCOE-TRAFFIC* class was assigned to a queue (3), as was the enterprise traffic (1) that was seen on the interface:

```
class-of-service {
  forwarding-classes {
```

```

        class FCoE-TRAFFIC queue-num 3;
        class ENTERPRISE-TRAFFIC queue-num 1;
    }
}

```

Priority flow control was based on congestion levels on the switch. The congestion notification profile associated the traffic with the PFC process. The configuration for the congestion notification profile was:

```

class-of-service {
    congestion-notification-profile {
        DCB-FCOE {
            input {
                ieee-802.1 {
                    code-point 011 {
                        pfc;
                    }
                }
            }
        }
    }
}

```

Once the congestion profile had been created, it was assigned to the interface (*xe-0/0/6*) that connects to the metro Ethernet service:

```

class-of-service {
    interfaces {
        xe-0/0/6 {
            congestion-notification-profile DCB-FCOE;
            unit 0 {
                classifiers {
                    ieee-802.1 FC-PFC;
                }
            }
        }
    }
}

```

The congestion profile associated the FCoE interface with the PFC process. The next step was to create a traffic control profile that would manage the buffers on the interface based on the traffic classes. The traffic profile correlated the traffic classes, the bandwidth percentages per class, and the traffic scheduler maps:

```

class-of-service {
    traffic-control-profiles {
        FCoE-TCP {
            scheduler-map FCoE-MAP;
            guaranteed-rate percent 90;
        }
        ENTERPRISE-TCP {
            scheduler-map ENTERPRISE-MAP;
        }
    }
}

```

```

        guaranteed-rate percent 10;
    }
}

```

In this case, both of the traffic classes were assigned to the same interfaces. In such circumstances, Juniper allows either multiple classes of service to be associated with an interface, or the creation of a service set to be applied to the interface. We chose to use the service set approach in this implementation. The forwarding classes were assigned to the forwarding set, and the set was applied to the interface. The traffic control profile was also added to the interface to accompany the congestion notification profile that was applied earlier:

```

class-of-service {
    forwarding-class-sets {
        FCoE-TRAFFIC-SET {
            class FCoE-TRAFFIC;
        }
        ENTERPRISE-TRAFFIC-SET {
            class ENTERPRISE-TRAFFIC;
        }
    }
}
interfaces {
    xe-0/0/24 {
        forwarding-class-set {
            FCoE-TRAFFIC-SET {
                output-traffic-control-profile FCoE-TCP;
            }
            ENTERPRISE-TRAFFIC-SET {
                output-traffic-control-profile ENTERPRISE-TCP;
            }
        }
        congestion-notification-profile DCB-FCOE;
        unit 0 {
            classifiers {
                ieee-802.1 FC-PFC;
            }
        }
    }
}

```

The class of service configuration was completed with the definition of the schedulers and the scheduler maps, each referencing the forwarding classes. The scheduler maps were referenced in the traffic profiles assigned above. Where the traffic profiles defined the percentages of traffic guaranteed per traffic class, the schedulers allocated the bandwidth on the outgoing buffers. In this case we assigned bandwidth numbers that were approximately equal to the 90/10 allocation defined in the traffic profiles:

```

class-of-service {
    scheduler-maps {

```

```

FCoE-MAP {
    forwarding-class fcoe scheduler FCOE-SCHED;
}
ENTERPRISE-MAP {
    forwarding-class best-effort scheduler ENTERPRISE-SCHED;
}
}
schedulers {
    FCOE-SCHED {
        transmit-rate 3500m;
        priority strict-high;
    }
    ENTERPRISE-SCHED {
        transmit-rate 500m;
        priority low;
    }
}
}

```

EX4500 Transit Switch Configurations

Although the Ethernet frames were carrying a different payload from most installations (the Fibre Channel messages), the EX4500 transit switch carried solely Ethernet traffic. As such, the configuration could have been just that of an Ethernet switch. The reason for the special treatment and special configuration was to provide the service levels defined for Fibre Channel. The EX4500 formed the remote end of the DCB pair: it acted as the FCoE transit switch, prioritizing Fibre Channel traffic ahead of the enterprise traffic.

As for the QFX3500, the configuration was divided into management and traffic portions. The management configuration was identical to the configuration defined for the QFX3500, with the exception of the host identifier and the management addresses.

The traffic portion of the configuration was significantly different and is provided here.

Interfaces and VLANs

The EX4500 was greatly underutilized for this installation—only 2 of the 48 ports were initially used, although we were assured that the switch would be used for other traffic once the installation was complete. A little knowledge transfer would allow the new owners to set interfaces and VLANs on the device themselves.

The interfaces used for the FCoE traffic were the *xe-0/0/0* interface going to the converged network adapter on the research server and interface *xe-0/0/1* going to the metro Ethernet service. The server tagged traffic for the appropriate VLAN as follows:

- VLAN 100 was for FCoE traffic with an 802.1p priority of 011 (3).

- VLAN 50 was for back office (email and DNS) enterprise traffic with no 802.1p priority.
- Any FCoE Initialization Protocol (FIP) traffic was untagged and was sent over the metro Ethernet service as *vlan-id* 1 (the native VLAN). (FIP is the control protocol used to support the FCoE signaling.)

The configuration for the interfaces and the VLANs was:

```

interfaces {
    xe-0/0/1 {
        mtu 2180;
        unit 0 {
            family ethernet-switching {
                port-mode trunk;
                vlan {
                    members [ FC-VLAN Enterprise-VLAN ];
                }
                native-vlan-id 1;
            }
        }
    }
    xe-0/0/0 {
        mtu 2180;
        unit 0 {
            family ethernet-switching {
                port-mode trunk;
                vlan {
                    members [ FC-VLAN Enterprise-VLAN ];
                }
                native-vlan-id 1;
            }
        }
    }
}

```

The initial interface configuration assigned the interfaces as trunk ports and associated the various VLANs with these interfaces. The VLANs can be associated in the interfaces stanza or the VLAN stanza. Associating the interfaces in both places doesn't hurt.

One additional interface was defined on the transit switch that was for management traffic on the native VLAN (*vlan-id* 1):

```

interfaces {
    vlan {
        unit 1 {
            family inet {
                filter {
                    input ACME-MGMT-VLAN-IN;
                }
                address 6.23.10.21/24;
            }
        }
    }
}

```



```

    }
  }
}

```

As for the QFX3500, the management interface was protected by a firewall filter that controlled all management access to the device. This filter was copied and pasted directly from the QFX3500 configuration to the EX4500 configuration, so it is not shown here.

The next configuration added to the transit switch was the VLAN configuration. The three VLANs were defined and associated with the interfaces. The management interface was also associated with the native VLAN:

```

vlans {
  vlan1 {
    vlan-id 1;
    interface {
      xe-0/0/0.0;
      xe-0/0/1.0;
    }
    l3-interface vlan.1;
  }
  FC-VLAN {
    vlan-id 100;
    interface {
      xe-0/0/0.0;
      xe-0/0/1.0;
    }
  }
  Enterprise-VLAN {
    vlan-id 50;
    interface {
      xe-0/0/0.0;
      xe-0/0/1.0;
    }
  }
}

```



The Juniper Networks guidelines for an FCoE transit switch define a FIP snooping configuration. FIP snooping allows the transit switch to monitor the Fibre Channel traffic and intercept the FIP traffic (the control traffic for FCoE virtual links). The transit switch creates dynamic filters based on the Fibre Channel login traffic to prohibit a rogue FC server from accessing the Fibre Channel fabric. In our client's network, the remote servers and transit switches are in secured areas, so the possibility of a rogue server was small. As such, we decided to forgo the complexity of the FIP snooping and dynamic filters.

While the interfaces and VLANs were needed to get the FCoE traffic to the metro Ethernet service provider, we also had to configure the class of service to assure that the Fibre Channel service level guarantees were met. The class of service configurations were very similar to those defined for the QFX3500, but contained different interfaces and values.

Transit Switch DCB Configuration

The data center bridging at the transit switch assured that the Fibre Channel traffic was given priority over the enterprise traffic. Both traffic types appeared on both channels, so the DCB configuration included both the interfaces of the switch. DCBX was running on the interface to the metro Ethernet service, while on the server interface, only the class of service elements were assigned:

```
protocols {
  lldp {
    interface xe-0/0/1.0;
  }
  dcbx {
    interface xe-0/0/1.0;
  }
}
```

This configuration formed the remote end of the QFX3500 DCBX configuration. The full capabilities of the DCB were negotiated between the two devices. The lowest common configurations were activated between the two devices.

To activate PFC and ETS, a configuration similar to that found in the QFX3500 was installed in the EX4500. The configuration was:

```
class-of-service {
  classifiers {
    ieee-802.1 FC-PFC {
      forwarding-class FCoE-TRAFFIC {
        loss-priority low code-points 011;
      }
      forwarding-class ENTERPRISE-TRAFFIC {
        loss-priority low code-points 000;
      }
    }
  }
  forwarding-classes {
    class FCoE-TRAFFIC queue-num 3;
    class ENTERPRISE-TRAFFIC queue-num 1;
  }
  congestion-notification-profile {
    DCB-FCOE {
      input {
        ieee-802.1 {
          code-point 011 {
```

pfc;

While the traffic classifier was assigned to both interfaces, the congestion notification profile was assigned only to the interface to the metro Ethernet service:

```

class-of-service {
    interfaces {
        xe-0/0/1 {
            congestion-notification-profile DCB-FCOE;
            unit 0 {
                classifiers {
                    ieee-802.1 FC-PFC;
                }
            }
        }
        xe-0/0/0 {
            unit 0 {
                classifiers {
                    ieee-802.1 FC-PFC;
                }
            }
        }
    }
}

```

To provide symmetry between the switches, the same traffic control profiles were used on the EX4500 as defined on the QFX. These were then assigned to the outgoing interface:

```
class-of-service {
  traffic-control-profiles {
    FCoE-TCP {
      scheduler-map FCoE-MAP;
      guaranteed-rate percent 90;
    }
    ENTERPRISE-TCP {
      scheduler-map ENTERPRISE-MAP;
      guaranteed-rate percent 10;
    }
  }
}
forwarding-class-sets {
  FCoE-TRAFFIC-SET {
    class FCoE-TRAFFIC;
  }
  ENTERPRISE-TRAFFIC-SET {
    class ENTERPRISE-TRAFFIC;
  }
}
```

```

}
interfaces {
  xe-0/0/1 {
    forwarding-class-set {
      FCoE-TRAFFIC-SET {
        output-traffic-control-profile FCoE-TCP;
      }
      ENTERPRISE-TRAFFIC-SET {
        output-traffic-control-profile ENTERPRISE-TCP;
      }
    }
    congestion-notification-profile DCB-FCOE;
    unit 0 {
      classifiers {
        ieee-802.1 FC-PFC;
      }
    }
  }
}
}

```

The server interface only needed the traffic profile set assigned to provide proper buffer management for returning traffic. The interface configuration was:

```

class-of-service {
  interfaces {
    xe-0/0/0 {
      forwarding-class-set {
        FCoE-TRAFFIC-SET {
          output-traffic-control-profile FCoE-TCP;
        }
        ENTERPRISE-TRAFFIC-SET {
          output-traffic-control-profile ENTERPRISE-TCP;
        }
      }
      unit 0 {
        classifiers {
          ieee-802.1 FC-PFC;
        }
      }
    }
  }
}

```

The schedulers and the schedule maps are a direct copy from the QFX3500:

```

class-of-service {
  scheduler-maps {
    FCoE-MAP {
      forwarding-class fcoe scheduler FCOE-SCHED;
    }
    ENTERPRISE-MAP {
      forwarding-class best-effort scheduler ENTERPRISE-SCHED;
    }
  }
}

```

```

    }
}
schedulers {
    FCOE-SCHED {
        transmit-rate 3500m;
        priority strict-high;
    }
    ENTERPRISE-SCHED {
        transmit-rate 500m;
        priority low;
    }
}
}

```

Verification

In addition to the normal interface status and address resolution protocol entries, the QFX3500 supports a number of Fibre Channel–specific verification commands. These display the operation of the ports, the fabric, and the protocols used in the configuration. The first command we’ll look at shows the operation of the FCoE Initialization Protocol (the protocol that runs on the native VLAN). The command and its display are:

```

admin@ACME> show fibre-channel fip
Fabric Name : ACME-DRUG (100)
Member
FCF-MAC : 84:18:88:d4:d3:92 (Interface vlan.100)
Enode
Enode-MAC : 00:00:c9:1b:e4:a4 State : Logged-in
Session
VN_Port MAC : 0e:fc:00:05:00:01

```

This command shows that FCoE is up and operational and that the far end of the circuit is logged in. The MAC addresses are from the server (E node) and the virtual node ports of the QFX3500.

The QFX3500 has a switching fabric for Fibre Channel connections. The next command displays details on its operation and the MAC addresses found in that fabric. The command and its display are:

```

admin@ACME> show fibre-channel routes
Fabric: ACME-DRUG
Route-prefix State Interface Mac-Address Index Flags
0x010000/24 Active fc-0/0/0.0 1660 kernel
0x010001/24 Active vlan.100 0e:fc:00:05:00:01 1661 kernel

```

This display shows that the fabric connections are active and that the incoming interface *vlan.100* is connected to the Fibre Channel interface (*fc-0/0/0.0*).

In the normal operation of Fibre Channel, the servers log into the switch fabric and are associated with their worldwide names and Fibre Channel identifiers. This login se-

quence passes through the QFX3500 on its way to the actual Fibre Channel switch. The login sequence is called a FLOGI (Fabric LOGIn). The QFX3500 is a proxy in the FLOGI process and keeps track of the Fibre Channel users and their state. This command shows the FLOGI users for the fabric (F) ports:

```
admin@ACME> show fibre-channel flogi fport
Fabric: ACME-DRUG
Interface Mac-Address      State Logins NPIV  FLOGI-Port-WWN
vlan.100  00:00:c9:a3:87:a3 Up    1      Yes   10:00:00:00:c9:a3:87:a3
```

As can be seen in this display, the WWN for the Fibre Channel server is the server's MAC address plus a prefix of 10:00.

The same command can be run to display the characteristics of the node port on the gateway switch. This display shows similar information, but from the perspective of the node rather than the fabric:

```
admin@ACME> show fibre-channel flogi nport
Fabric: ACME-DRUG
Interface FCID      Port WWN      Node WWN
vlan.100  0x010003  10:00:00:00:c9:1b:e4:a4  20:00:00:00:c9:1b:e4:a4
State
online
```

The last element to verify is that the DCBX parameters are correct. The interface to the metro Ethernet service has DCBX running, and it negotiates the other DCBX components. The DCBX elements are displayed by using:

```
admin@ACME> show dcbx neighbors interface xe-0/0/6
Interface : xe-0/0/6

Neighbor-id: 143540981
Protocol-State: in-sync

Local-Advertisement:
  Operational version: 1
  sequence-number: 198, acknowledge-id: 156

Peer-Advertisement:
  Operational version: 1
  sequence-number: 156, acknowledge-id: 198

Feature: PFC, Protocol-State: in-sync

Operational State: Enabled

Local-Advertisement:
  Enable: Yes, Willing: No, Error: No
  Maximum Traffic Classes capable to support PFC: 8

Code Point      Admin Mode
000             Enabled
```

001	Disabled
010	Disabled
011	Enabled
100	Disabled
101	Disabled
110	Disabled
111	Disabled

Peer-Advertisement:

Enable: Yes, Willing: Yes, Error: No
Maximum Traffic Classes capable to support PFC: 8

Code Point	Admin Mode
000	Enabled
001	Disabled
010	Disabled
011	Enabled
100	Disabled
101	Disabled
110	Disabled
111	Disabled

Feature: Application, Protocol-State: in-sync

Operational State: Enabled

Local-Advertisement:

Enable: Yes, Willing: No, Error: No

Appl-Name	Ethernet-Type	Socket-Number	Priority-Map	Status
FCoE	0x8906	N/A	00000011	N/A

Peer-Advertisement:

Enable: Yes, Willing: Yes, Error: No

Appl-Name	Ethernet-Type	Socket-Number	Priority-Map	Status
FCoE	0x8906	N/A	00000011	N/A

Feature: ETS, Protocol-State: in-sync

Operational State: Enabled

Local-Advertisement:

Enable: Yes, Willing: No, Error: No
Maximum Traffic Classes capable to support PFC: 8

Code Point	Priority-Group
000	0
001	0
010	0
011	7
100	0

101	0
110	0
111	0

Priority-Group	Percentage B/W
0	5%
1	95%

Peer-Advertisement:

Enable: Yes, Willing: No, Error: No

Maximum Traffic Classes capable to support PFC: 8

Code Point	Priority-Group
000	0
001	0
010	0
011	7
100	0
101	0
110	0
111	0

Priority-Group	Percentage B/W
0	5%
1	95%

This display shows that the DCB elements PFS and ETS are up and operational and that the bandwidth allocated and the priorities that were assigned have been accepted across the interface.

Once we had verified the configurations, tested the circuits, and verified the FCoE as operational, it was time to pass application traffic. The test server that was installed had test applications that could communicate with the data center. After a little tweaking in the area of server applications (an area viewed by this engineer as akin to wizardry), the test sequences worked like a charm.

The Ethernet replacement for the dark fiber was connected, operational, and awaiting real traffic.

Conclusions

This engagement allowed a newly created tribe to work together on a new technology (Fibre Channel), a new product (QFC3500), and a new solution (FCoE)—all in all, not a run-of-the-mill engagement. The client had a vision of what was possible and what the gains from that vision might be, and we warriors were brought in to make the vision a reality. What fun this profession can be!

MX Network Deployment

Most professional service engagements that we warriors are privileged to be associated with amount to the proverbial “cleanup on aisle 4.” The tribe is requested to fix a situation where the customer has deployed Juniper Networks equipment and things have gone very wrong. We pack our tools and start troubleshooting and repairing.

This engagement was a refreshing change to the normal warrior battle. In this case, our team was engaged prior to the deployment of the equipment and was given the opportunity to design, test, and deploy the initial installment of Juniper Networks MX edge routers.

The customer was part of a growing trend in the US, a regional power company looking to diversify and provide wholesale Internet services as well as power. The company had the rights of way and the manpower to install fiber and add routers—and poof, a regional ISP was born. In the final stages of deployment the company hoped to offer Internet services to its stable of 15 power companies.

The company had existing or planned peering agreements with three national carriers and one regional carrier. The regional carrier had an active link to the customer’s lab, pushing a full Internet BGP feed of routes. This is what we warriors call a fertile green-field installation. The customer had the plans, the contacts, and the hardware to implement the network.

The initial hardware was a quintet of MX480s, each with a single dense port concentrator (GE and XE modular interface cards) and redundant routing engines. The MXs provided backbone and edge service functions for the customer. These were deployed in the customer’s lab and were powered up and operational.

This type of engagement excites a warrior—a fresh canvas to work with, new hardware to deploy, and a game plan that allows for the best possible outcome for the customer. Let’s look at the toys and the plans.

Plans and Topology

The company was planning on deploying a four-node backbone that had full mesh interconnectivity. The remaining locations were to be fed from this backbone square. The peering ISPs connected to two of the backbone nodes. The initial network diagram is shown in [Figure 9-1](#).

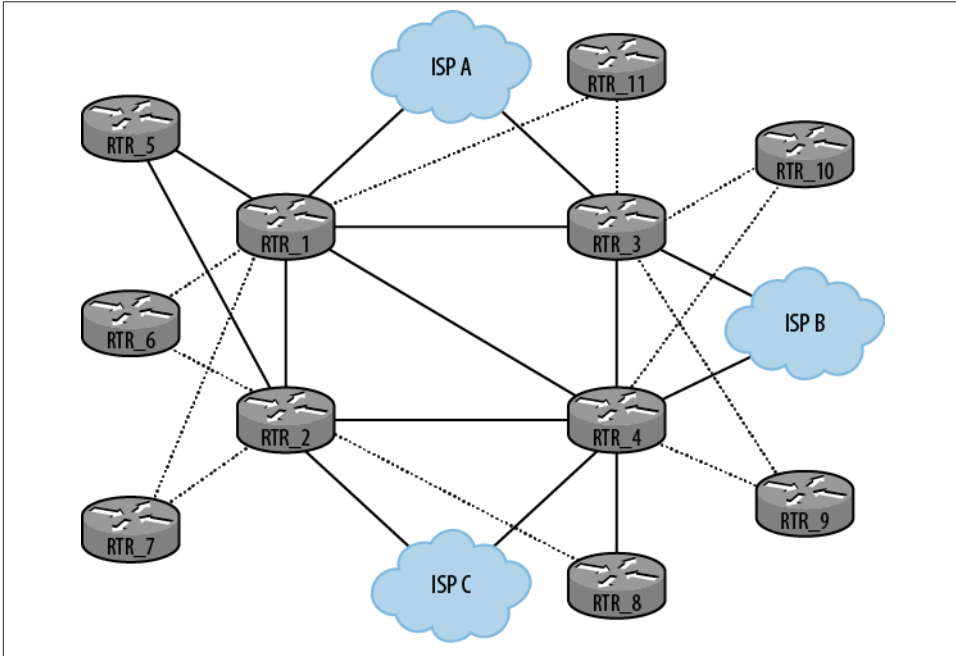


Figure 9-1. Network topology

Routers *RTR_1* through *RTR_5* were to be deployed initially, and the other six as those customers came online. The backbone links were all 10 Gbps links, while the access links and the ISP links were initially 1 Gbps links. The plan was to upgrade the ISP links to 10 Gbps as well as the business demands increased.

The key elements of the design were:

Survivability

The backbone and the edge devices should be as reliable as possible. The links and the hardware should be able to absorb a single failure and not isolate users.

Availability

As demands for bandwidth increase, the design should be such that as much traffic as possible can be handled by the available bandwidth, and when the well is dry, more bandwidth can be added in a noninterruptive manner.

Bandwidth

The bandwidth demands are expected to increase. The initial deployment is intended to verify the design and bring the customers online; as soon as traffic patterns are known, additional fibers can be added to meet the growing demand.

The plans for the deployment were pretty simple:

- Phase 1 was to get *RTR_1* up to par with best practices in BGP and Internet security.
- Phase 2 was to bring *RTR_2*, *RTR_3*, and *RTR_4* online in the test bed.
- Phase 3 was to bring up *RTR_5* and attach it to the administrative network for Internet access from the office (proof of concept).
- Phase 4 was to install the MXs in the field and go live (not part of this engagement).

Phase 1

A little research identified a couple of sources for brushing up this warrior's knowledge in this area: a Juniper Day One book called *Securing the Routing Engine on M, MX, and T Series*, two RFCs (RFC 3013, Recommended Internet Service Provider Security Services and Procedures, and RFC 2979, Behavior of and Requirements for Firewalls), and a Cisco document (Cisco Document ID: 13608 Cisco Guide to Hardened Cisco IOS Devices—Cisco Systems). Yes, a Juniper Networks warrior will use any sources for ideas—even Cisco. These resources prescribe a number of elements for securing an Internet router. There is also an **excellent example** on the Juniper site that identifies the configuration for a secured router. Pulling from each of these sources, the following is a list of the security requirements for the MXs:

- Set manageable but secure password criteria.
- Disable unused services.
- Use firewall filters to restrict management access.
- Set thresholds for routing engine (RE) access.
- Use secure command-line interface (CLI) access with timeouts.
- Install warning banners.
- Use a two- or three-part authentication, authorization, and accounting (AAA) process.
- Use SNMPv3 with filters.
- Log both locally and to a secured server.
- Use the comment and archive on commit options.
- Configure network time protocol (NTP) authentication if NTP is being used.

- Use secure routing protocols:
 - BGP (TTL, MD5, filters)
 - IGP (MD5, passive interface, filters)
 - BFD for all protocols
 - Secure VRRP
- Filter undesirable IP packets.
- Create anti-spoofing filters (both egress and ingress).

For the BGP configuration, the regional ISP provided us with its interconnect document. Here is what it expected from this new ISP:

- Prefer to receive an aggregate of the locally assigned addresses
- No subnets longer than /24
- No RFC 1918 prefixes
- Authentication on all BGP links

The document also indicated that the upstream ISPs:

- Will ignore the use of MEDs
- Will respond to the application of local preference with the receipt of community strings of the form `xxxxx:YY`, where `YY` is the local preference to be applied to the prefix
- Will forward a default route if required
- Will accept prepending only for the local AS

Finally, it was specified that the new ISP should not act as a transit network for any traffic other than its customers'.

Continuing Education

One of the many things that I enjoy about being a network warrior, and especially a Juniper Networks warrior, is the opportunity to continue my education. I started in this industry 35 years ago as college student; I've been pushing wire and making things communicate ever since. When I joined Juniper I was able to broaden my education by learning about the wide range of devices and systems that Juniper Networks makes available to customers. I came into Juniper knowing routers, quickly learned firewalls, then moved into switches,

infranet controllers, and secure access devices—and I know that I have more to learn. Each day and each assignment is a learning experience. How about you? Are you a continuing educator, a tech warrior? Look into the educational services that Juniper Networks and its partners offer (unabashed plug for Proteus Networks).

MX Configuration

With this list of key elements, security ideas, and routing requirements in hand, it was time to create the initial configuration for the first MX. This initial configuration would be used as the template for the other backbone MXs. The non-backbone (*RTR_5*) MX would also inherit most of the configuration.

Management Configuration

The configuration is broken into multiple parts, each described separately so you can absorb it more easily. The first part is the definition of the out-of-band management network. Each MX is connected to an in-house management network that uses private addressing (192.168.0.0/16). The network operations center, the administrative staff, and the management servers are all found on this network. There are two components to configure for this network—the first is the addressing for the *fxp0* ports, and the second is the static route for sending traffic to the management network:

```
groups {
  re0 {
    interfaces {
      fxp0 {
        unit 0 {
          family inet {
            address 192.168.0.251/24;
            address 192.168.0.253/24 {
              master-only;
            }
          }
        }
      }
    }
  }
}

re1 {
  interfaces {
    fxp0 {
      unit 0 {
        family inet {
          address 192.168.0.252/24;
          address 192.168.0.253/24 {
            master-only;
          }
        }
      }
    }
  }
}
```

```

    }
  }
}
apply-groups "${re}";

```

The above configuration assigns a unique address to each of the management ports and a virtual IP address to the routing engine that is acting as master of the MX.

The routing options portion of the configuration allows all management traffic to be pushed out the *fxp0* ports. The *retain* flag allows the backup routing engine to keep this address in the forwarding table even when it is not master of the MX. The *no-readvertise* knob is a safety precaution to assure that this route is never sent to the world (or internally):

```

routing-options {
  static {
    route 192.168.0.0/16 {
      next-hop 192.168.0.1;
      retain;
      no-readvertise;
    }
  }
}

```

The default services supported on the MX allow SSH connections. This is a good start, but even SSH has been known to be blocked by denial of service (DOS) attacks. To counter these attacks, the SSH service is modified with connection restrictions, and root access is denied from SSH (this might cause headaches at some point, but for now, safety is the overriding factor):

```

services {
  ssh {
    root-login deny;
    connection-limit 5;
    rate-limit 2;
  }
}

```

The connection restrictions allow only five users to be logged in at once and a connection rate of two connections per minute. This reduces the chance of a DOS attack.

To set the session timeout for an idle terminal, you enter the command at the operational prompt:

```
set cli idle-timeout 5
```

Routing Engine Protection

The next portion of the configuration is the user login. The guidelines call for a two- or three-part authentication scheme. At this point the ISP did not have the authentication servers online, so a SecureID scheme was not possible. The only thing we were able to implement at this time was a strict password scheme to make it hard for hackers to guess the passwords. Junos supports the capability of requiring stupidly hard passwords (e.g., 20 characters and 10 changes), but a little research showed that the consensus for a strong password is a minimum of 8 characters with 3 of the following four changes: uppercase, lowercase, numbers, and punctuation. The configuration that supports these constraints is:

```
system {
  password {
    minimum-length 8;
    change-type character-sets;
    minimum-changes 3;
  }
}
```

The routing engine is the critical component for Juniper Networks routers—and as warriors, we know that we must always protect the critical components. An interface can be lost to a DOS attack, but if the routing engine is lost, the entire router is compromised. To protect the routing engine, two different configurations are used. The first protection is to set up the Internet options to limit certain traffic types. The configuration for this protection is:

```
system {
  internet-options {
    icmpv4-rate-limit packet-rate 100 bucket-size 5;
    no-source-quench;
    tcp-drop-synfin-set;
  }
}
```

The first term limits the rate of ICMP traffic to 500 packets per second. Warriors like to test and test again, and when that's done, test some more. The principal connectivity test tool is the ping, so to a warrior, allowing ICMP echo requests and responses is a requirement. Some firewall warriors will argue with us network warriors, saying that ping gives up too much information about the device, and so on. From our point of view, though, testing should be possible, and if other controls are in place, pings should not hurt. The second term says to not respond to source quench messages. These can restrict the RE and really should never been seen. The final term says to have the RE drop any TCP message with both the SYN and FIN control bits set. These can only mean trouble, so it's best to get rid of them.

The second part of the RE protection is a firewall filter to control the traffic to the RE. This filter is placed on the loopback interface and controls what traffic can enter the RE (it's an input filter).



To protect the RE, filters are applied to the loopback interface. Even though not all traffic to the RE (exception traffic and the like) is addressed to the loopback address, the filter on the loopback address “sees” all traffic between the flexible PIC concentrators and the RE. Traffic on the *fxp0* ports is not subject to these filters.

To craft the loopback (RE) filter, an analysis is done on what traffic should be passing between the network and the RE. This traffic includes:

- Routing protocols (BGP, OSPF, BFD)
- Administrative protocols (SSH, SNMP, NTP, DNS)
- Testing protocols (*ping* and *traceroute*)

The filter should also deal with the security concerns that were researched above:

- Address spoofing
- Private/Martian addresses
- Undesirable IP packets
- Restricting SSH access to known addresses

Now that the *whats* have been defined for the filter, the *hows* have to be determined.

There are many ways to set up a filter to protect the routing engine. This is the most secure and simplest of the possible options:

```
set firewall family inet filter PROTECT then reject
set interface lo0.0 family inet filter input PROTECT
```

However, it's not functional, as it will close all traffic to the RE and not allow the router to perform its functions. An operational option is to create a single firewall filter and add a term for each protocol listed above. This approach works, but it will cause problems in the future—because of the order and the complexity of each term, it will quickly become a nightmare for anyone to manage (except for the original author, who might or might not be around).

Another option is to create a filter list and attach that to the interface. But a filter list can only reference up to 16 filters, so to allow for growth, nesting of terms and filters will be necessary. That means you'll have two places to order and reference protocols: in the filter list and in the nested filters themselves. This isn't a bad option, but again, it's likely to result in management headaches.

The option that we agreed upon was to use a single firewall filter applied to the loopback interface that references all the individual filters. Each protocol has its own filter; the ordering is done at the main filter level. So, the content is separated from the order, with the content being found in the individual filters and the ordering found in the overall filter.

The next question was how to create a filter that can be used on any of the routers in the system. Each router has its own set of interface addresses, its own neighbors, and possibly its own servers (DNS, NTP, etc.). Each filter could be created manually (read management nightmares and bookkeeping), or each filter can rely on Junos to define these variable elements. This is the option that we chose. By using apply lists, we allowed the variables of the routers to be determined by Junos and added to the filters. This way, when the addresses change or new ones are added, the filters need not be changed manually.

This command reads the configuration, looking for the last portion of the *apply-path* statement (*address <*>*), and copies this to the prefix list called *Interface*:

```
set policy-options prefix-lists Interface apply-path "interfaces <*>  
unit <*> family inet address <*>"
```

On the test router, the above prefix list returns a list of direct addresses, as in:

```
show policy-options prefix-list Interface | display inheritance  
##  
## apply-path was expanded to:  
## 10.10.10.0/24;  
## 10.10.10.0/24;  
## 10.1.188.184/30;  
## 10.10.11.1/32;  
##  
apply-path "interfaces <*> unit <*> family inet address <*>"
```

KISS

One of guiding principles of a Juniper Networks warrior is the constant focus on KISS—Keep It Super Simple (or Keep It Simple, Stupid when we forget). One of the tenets of this focus is to let the machine do the work. If there are ways to automate the mindless work, use them. If there is a way to accomplish a requirement in one line versus five lines, use it. Do not add complexity to make the customer think we are geniuses. The simple, elegant solution is always the best.

The prefix lists are used to find the local interfaces, the BGP neighbors, and the DNS, NTP, SNMP, and Radius servers. Prefix lists are also used to identify the OSPF multicast addresses and the remote administrators that will be allowed to SSH to the MXs from locations on the Internet (we also added the addresses of the other MX interfaces, 11.13.132.0/28, to allow SSH from one device to another). Local administrators use the *fxp0* port, as defined previously.

The prefix lists are:

```
policy-options {
  prefix-list INT {
    apply-path "interfaces <*> unit <*> family inet address <*>";
  }
  prefix-list BGP {
    apply-path "protocols bgp group <*> neighbor <*>";
  }
  prefix-list OSPF {
    224.0.0.5/32;
    224.0.0.6/32;
  }
  prefix-list NTP {
    apply-path "system ntp server <*>";
  }
  prefix-list SNMP {
    apply-path "snmp community <*> clients <*>";
  }
  prefix-list DNS {
    apply-path "system name-server <*>";
  }
  prefix-list ADMIN {
    12.100.103.45/32;
    45.126.78.190/32;
    100.10.10.0/24;
    11.13.132.0/28
  }
}
```

The firewall filters will allow specific traffic but will not control the amount of that traffic that is allowed to enter the router. If a hacker were to initiate a DOS attack using OSPF frames, our firewall filter would validate the traffic as allowed and present the traffic to the routing engine. Junos has a threshold limit for the RE, but why leave this up to chance? A policer will limit the traffic at the interface rather than tying up all the facilities between the interface and the RE. A safe amount of traffic is 5 Mbps; any traffic source sending more than that amount is probably trouble.

When a policer is created, two quantities are entered: the maximum data rate and the maximum burst rate. The first is whatever you choose (5 Mbps, in our case), and the

second should be set based on the amount of traffic that can pass over the interface in 5 milliseconds. Since the RE interface is a 1 Gbps link in the MX, the maximum burst size should be 650,000 bytes (yes, the units are bytes, not bits or bits per second). The resulting policer is:

```
policer 5M {
    filter-specific;
    if-exceeding {
        bandwidth-limit 5m;
        burst-size-limit 650k;
    }
    then discard;
}
```

Once the policer is entered and committed, the firewall filters can be created. There are two possible locations for the filters, the *firewall filter* hierarchy and the *firewall family inet filter* hierarchy. Either will work, but the former is an older hierarchy that many folks wish was deprecated. For our filters, we used the newer and preferred firewall family inet filter hierarchy. Each filter requires the use of a term and each specifies a single protocol. The firewall filters are:

```
firewall {
    family inet {
        filter BGP {
            term BGP {
                from {
                    source-prefix-list {
                        BGP;
                    }
                    destination-prefix-list {
                        INT;
                    }
                    protocol tcp;
                    port bgp;
                }
                then {
                    count bgp;
                    accept;
                }
            }
        }
    }
    filter OSPF {
        term OSPF {
            from {
                source-prefix-list {
                    OSPF;
                }
                destination-prefix-list {
                    INT;
                }
            }
            protocol ospf;
        }
    }
}
```

```

        }
        then {
            count ospf;
            accept;
        }
    }
}
filter ICMP {
    term ICMP-fragments {
        from {
            is-fragment;
            protocol icmp;
        }
        then {
            discard;
        }
    }
    term ICMP-allowed {
        from {
            protocol icmp;
            icmp-type [ echo-reply echo-request time-exceeded
                        unreachable source-quench
                        router-advertisement
                        parameter-problem ];
        }
        then {
            policer 5M;
            count ICMP;
            accept;
        }
    }
}
filter SSH {
    term SSH {
        from {
            source-prefix-list {
                ADMIN;
            }
            destination-prefix-list {
                INT;
            }
            protocol tcp;
            port ssh;
        }
        then {
            count SSH;
            accept;
        }
    }
}
filter SNMP {
    term SNMP {

```

```

        from {
            source-prefix-list {
                SNMP;
            }
            destination-prefix-list {
                INT;
            }
            protocol udp;
            port snmp;
        }
        then {
            policer 5M;
            count snmp;
            accept;
        }
    }
}

filter NTP {
    term NTP {
        from {
            source-prefix-list {
                NTP;
            }
            destination-prefix-list {
                INT;
            }
            protocol udp;
            port ntp;
        }
        then {
            policer 5M;
            count ntp;
            accept;
        }
    }
}

filter TRACERT {
    term UDP {
        from {
            destination-prefix-list {
                INT;
            }
            protocol udp;
            ttl 1;
            destination-port 33435-33450;
        }
        then {
            policer 5M;
            count trace-udp;
            accept;
        }
    }
}

```

```

    term TCP {
        from {
            destination-prefix-list {
                INT;
            }
            protocol tcp;
            ttl 1;
        }
        then {
            policer 5M;
            count trace-tcp;
            accept;
        }
    }
}
filter BFD {
    term BFD {
        from {
            source-prefix-list {
                INT;
            }
            destination-prefix-list {
                INT;
            }
            protocol udp;
            source-port 49152-65535;
            destination-port 3784-3785;
        }
        then {
            count bfd;
            accept;
        }
    }
}
filter SPOOF {
    term SPOOF {
        from {
            source-prefix-list {
                INT;
            }
        }
        then {
            discard;
        }
    }
}
filter BAD-IP {
    term OPTIONS {
        from {
            ip-options any;
        }
        then {

```

```

        discard;
    }
}
filter DISCARD-ALL {
    term DISCARD {
        then {
            discard;
        }
    }
}
}
}
}

```

There are really no surprises in the filters, so I won't spend time explaining each stanza. However, there are a few things to mention here. First, each accept filter is given a counter to allow us to see whether traffic is passing through the filter. This can be very helpful for troubleshooting down the road. Second, there is an implied “discard all” at the end of any filter, but we added our own discard all filter as this allows us to count and log the discarded traffic (again, as a troubleshooting tool). We could also initially change the action of the discard all to accept. This might be a good way to initially deploy the filter: it would allow all traffic through, and adding logging to the filter would allow us to look at what traffic was there and what we might have missed in the initial analysis. It is good to double check your firewall filters, but it is better to have a plan B ready just in case.

Once the filters are in place and committed, all that's left to do is reference them in the ordering filter and apply that filter to the loopback interface. The configuration of the ordering filter is:

```

firewall {
    family inet {
        filter Lo0-INPUT {
            term SPOOF {
                filter SPOOF;
            }
            term BAD-IP {
                filter BAD-IP;
            }
            term BFD {
                filter BFD;
            }
            term BGP {
                filter BGP;
            }
            term OSPF {
                filter OSPF;
            }
            term ICMP {
                filter ICMP;
            }
        }
    }
}

```



```

    }
    term NTP {
        filter NTP;
    }
    term SNMP {
        filter SNMP;
    }
    term SSH {
        filter SSH;
    }
    term TRACERT {
        filter TRACERT;
    }
    term DISCARD-ALL {
        filter DISCARD-ALL;
    }
}
}
}
unit 0 {
    family inet {
        filter {
            input Lo0-INPUT;
        }
        address 10.10.11.1/32;
    }
}

```

With this configuration in place, the commit is done with caution. If you're using Telnet to communicate to the MX over a revenue port, what will happen to the session? When adding this type of configuration, always use a `commit confirm`, because it is really embarrassing to have to drive out to a site just to fix a faulty firewall stanza.

Once the commit is performed and you've verified that you can still talk to the device, it's time to add the other services to the MX. The standard services include DNS, NTP, and SNMP. These servers are all on the public side of the network; the private services will be using the *fxp0* ports and are not covered by this firewall filter. The server stanzas are:

```

time-zone America/Chicago;
name-server {
    4.2.2.2;
}
ntp {
    server 64.90.182.55;
    server 96.47.67.105;
    server 206.246.122.250;
}
community PWRNET2011 {
    clients {

```

```

        12.23.155.198/32;
        156.23.84.90/32;
    }
}

```

Policy Configurations

The other part of the initial configuration is the configuration of the BGP policies that are used to interface to the upstream ISPs, and those that are used to interface to the downstream customers. The policies are the rules that are employed in the BGP configurations to import and export routes. The requirements that were received from the upstream ISP were listed earlier. In the following sections, each is addressed along with the appropriate policy stanza. Because of security issues, I will use bogus addresses from the 11/8 address block and an unassigned AS# of 24878.

Prefer to receive an aggregate of the locally assigned addresses

The block of addresses that have been assigned to the ISP is 11.13.132.0/22. An aggregate address is created for this address block, and that address is identified in the policy *LOCAL-OUT*:

```

policy-statement LOCAL-OUT {
    term IPv4-Prefix {
        from {
            family inet;
            protocol aggregate;
            route-filter 11.13.132.0/22 exact;
        }
        then {
            community add TAG-INTERNAL;
            accept;
        }
    }
}
community TAG-INTERNAL members 24878:1;

```

The community tag allows this prefix to be identified by the upstream routers. This prefix will only become active when a contributing prefix is found in the router table. The aggregate route configuration is:

```

routing-options {
    aggregate {
        route 11.13.132.0/22;
    }
    autonomous-system 24878;
}

```

The aggregate has a next hop of *reject*; any traffic that matches on this address is discarded.

No subnets longer than /24

This requirement can be satisfied on the incoming exchange and/or at the outgoing ISP exchange. We decided that checking on the incoming side would be good enough. The local routes are covered by the aggregate. The policy that is applied to the incoming exchanges from the downstream ISPs (our customers) is:

```
policy-statement CUST-IN {
  term ACCEPT-ROUTES {
    from {
      family inet;
      route-filter 0.0.0.0/0 upto /24;
    }
    then {
      community add TAG-CUSTOMER;
      accept;
    }
  }
}
community TAG-CUSTOMER members 24878:3;
```

This policy accepts all routes that are between /1 and /24. When accepted, the prefixes are tagged with a customer tag. This can be modified at a later time to be customer-specific; for now, a general tag is sufficient.

No RFC 1918 prefixes

While most ISPs will trash private addresses, it is also considered bad manners to advertise RFC 1918 addresses to an ISP. Again, we can filter these addresses at the input or the output, and again we decided to filter on the input. So, we created an import policy that looks for private addresses and rejects them. The policy is:

```
policy-statement NO-PRIV {
  term RFC-1918 {
    from {
      route-filter 10.0.0.0/8 orlonger;
      route-filter 192.168.0.0/16 orlonger;
      route-filter 172.16.0.0/12 orlonger;
    }
    then reject;
  }
}
```

Authentication on all BGP links

BGP supports MD5 authentication, and this is often a requirement for peering. The configuration for BGP authentication is shown as part of the BGP group configuration below.

The ISPs will ignore the use of MEDs

Multi-exit discriminators (MEDs) are used by ISPs to influence traffic entering their networks from a peer. Most ISPs ignore MEDs when received on a prefix, as is the case for our upstream ISP (“We will not apply them or respond to them if received.”). That is the default operation of Junos, so no configuration is needed.

The ISPs will respond to local preference

The upstream ISPs will respond to a request to apply a local preference to a prefix that has a common community string. We have a situation where each upstream ISP has two interfaces to our network. If we had wished to have those ports operate in a primary/secondary arrangement, this capability would have been necessary. We did not decide on that type of arrangement, though, and there was no reason to prohibit asymmetrical routing, so this capability was not required in this case. That being said, the configuration for such a policy would be:

```
policy-options {
  policy-statement LOCAL-PREF-80 {
    term 80 {
      from {
        route-filter 11.13.132.0/22 exact;
      }
      then {
        community add LOCAL-PREF-80;
        accept;
      }
    }
  }
  policy-statement LOCAL-PREF-110 {
    term 110 {
      from {
        route-filter 11.13.132.0/22 exact;
      }
      then {
        community add LOCAL-PREF-110;
        accept;
      }
    }
  }
}
community LOCAL-PREF-110 members 24878:110;
community LOCAL-PREF-80 members 24878:80;
```

On the other side of the equation is the receipt of prefixes from multiple upstream ISPs. For traffic control and troubleshooting purposes, some traffic management is needed to push traffic to one ISP. In a later section, we will directionalize incoming traffic, but

here we use local preference to send traffic out a single ISP. This treatment can be modified as necessary to provide load sharing between links and between ISPs. The policy is an import policy from the ISPs to set the local preference on incoming routes. It looks like:

```
policy-statement LOCAL-PREF-110 {
  term 110 {
    from {
      protocol bgp;
    }
    then {
      local-preference 110;
      accept;
    }
  }
}
```

This policy can be added to the policies for accepting traffic from the ISPs, discussed next.

The ISPs will forward a default route if required

In some cases it will better to receive a default route from an upstream ISP rather than or in addition to the full Internet feed of prefixes. The default route is handled differently than the full Internet feed as far as community strings are concerned. Three policies are created for this requirement. The first allows the default route as well as the full Internet feed, the second allows only the default route, and the final one blocks the receipt of the default route:

```
policy-options {
  policy-statement DEF-ONLY {
    term DEF {
      from {
        route-filter 0.0.0.0/0 exact;
      }
      then {
        community add TAG-DEF;
        accept;
      }
    }
    term REJECT-ALL {
      then reject;
    }
  }
  policy-statement INET-DEF {
    term DEF {
      from {
        route-filter 0.0.0.0/0 exact;
      }
      then {
        community add TAG-DEF;
      }
    }
  }
}
```

```

        accept;
    }
}
term INET {
    from {
        protocol bgp;
        route-filter 0.0.0.0/0 upto /24;
    }
    then {
        community add TAG-PEER;
        accept;
    }
}
term REJECT-ALL {
    then reject;
}
}
policy-statement INET-ONLY {
    term DEF {
        from {
            route-filter 0.0.0.0/0 exact;
        }
        then reject;
    }
    term INET {
        from {
            protocol bgp;
            route-filter 0.0.0.0/0 upto /24;
        }
        then {
            community add TAG-PEER;
            accept;
        }
    }
    term REJECT-ALL {
        then reject;
    }
}
community TAG-PEER members 24878:2;
community TAG-DEF members 24878:4;
}

```

The ISPs will accept prepending only for the local AS

When two or three upstream ISPs are being used, one of them will be the primary provider and the others will be in a secondary role. For outgoing traffic, the closest destination will be used, but for return and incoming traffic, the primary should carry most of the traffic. To create this situation, outgoing prefixes to the secondary ISPs are

prepended with the local AS. Prepending three ASes on the prefixes is typically enough to guarantee the primary/secondary relationship. This can be verified once the system goes live with looking-glass queries. The prepending policy is allied to the outgoing prefixes to the secondary ISPs. The policy that prepends prefixes is:

```
policy-options {
  policy-statement PREPEND {
    term ADD-3 {
      from {
        protocol bgp;
        community ISP-OUT;
      }
      then {
        as-path-prepend "24878 24878 24878";
        accept;
      }
    }
  }
  community ISP-OUT members "24878:(3|1)";
}
```

The ISP will not act as a transit network for any other traffic except for its customers

This requirement really means that our ISP will not advertise prefixes that are not generated locally or received from the downstream ISPs, and that all the prefixes received from the downstream ISPs are local to them or their customers. To meet this requirement, each downstream ISP is filtered to local traffic and those prefixes are tagged. The advertisements to the upstream ISPs then only have to filter based on the tags. Given that the local traffic is already tagged with the community string 24878:1 and the customers' traffic is tagged with the community string 24878:3, the policy to restrict these prefixes is shown above in the prepending policy and is repeated here without the prepending section:

```
policy-options {
  policy-statement UPStream-ISP-Out {
    term Accept {
      from {
        protocol bgp;
        community ISP-OUT;
      }
      then {
        accept;
      }
    }
  }
  community ISP-OUT members "24878:(3|1)";
}
```

To meet the requirement that the downstream ISPs do not advertise prefixes that are not local to them or their customers, a policy is created that looks at the AS path attribute

associated with the prefixes. To accomplish this policy, a regular expression is used to specify the options for the AS path attribute. As warriors, we have to keep current on a multitude of “stuff.” We quickly learn that some things are better left to the reference section of the Juniper Networks techpubs. That is the way that regular expression statements should be learned. The perfect reference spot is <http://bit.ly/UcGP5D>. Because the downstream ISP might use prepending, the filter looks for the downstream ISP’s AS# from 1 to 4 times, preceded by any other AS#. It is also possible that the downstream ISP is sending a local address, so the downstream AS# alone (or repeated multiple times) is allowed. Any other combinations will be rejected. To meet an earlier requirement, a downstream ISP policy was created to limit the prefixes to a maximum of a /24. This policy is revisited here to add these further requirements. The more restrictive policy is:

```

policy-options {
  policy-statement CUST-A-IN {
    term ACCEPT-ROUTES {
      from {
        family inet;
        route-filter 0.0.0.0/0 upto /24;
        as-path [ CUST-A CUST-A-TWO ]
      }
      then {
        community add TAG-CUSTOMER;
        accept;
      }
    }
  }
  community TAG-CUSTOMER members 24878:0003;
  as-path CUST-A-TWO ".24111{1,4}";
  as-path CUST-A "24111{1,4}";
}

```

The regular expressions mean:

".24111{1,4}"

Any AS# “.” followed by 24111 from one to four times

"24111{1,4}"

AS 24111 from one to four times

This policy is created specifically for each downstream ISP customer. We used the fictional (I hope) AS# 24111 as an example. When this system is fielded, this policy can be copied for each customer and the fictional number replaced with the proper AS#.

The use of copy and paste can save lots of time during the addition of downstream ISPs. An existing BGP group configuration can be copied, and well as the routing policies. As an example, the routing policy for downstream ISP “Customer A” (AS# 24111) is used as a template for downstream ISP “Customer B” (AS# 12345):


```
[edit policy-options]
peter@RTR_1# copy policy-statement CUST-A-IN to policy-statement CUST-B-IN
```

```
[edit policy-options]
peter@RTR_1# edit policy-statement CUST-B-IN
```

```
[edit policy-options policy-statement CUST-B-IN]
peter@RTR_1# replace pattern CUST-A with CUST-B
```

```
[edit policy-options policy-statement CUST-B-IN]
peter@RTR_1# show
term ACCEPT-ROUTES {
    from {
        family inet;
        as-path [ CUST-B CUST-B-TWO ];
        route-filter 0.0.0.0/0 upto /24;
    }
    then {
        community add TAG-CUSTOMER;
        accept;
    }
}
```

```
[edit policy-options policy-statement CUST-B-IN]
peter@RTR_1# up
```

```
[edit policy-options]
peter@RTR_1# copy as-path CUST-A to as-path CUST-B
```

```
[edit policy-options]
peter@RTR_1# copy as-path CUST-A-TWO to as-path CUST-B-TWO
```

```
[edit policy-options]
peter@RTR_1# edit as-path CUST-B
```

```
[edit policy-options as-path CUST-B]
peter@RTR_1# replace pattern 24111 with 12345
```

```
[edit policy-options as-path CUST-B]
peter@RTR_1# up
```

```
[edit policy-options]
peter@RTR_1# edit as-path CUST-B-TWO
```

```
[edit policy-options as-path CUST-B-TWO]
peter@RTR_1# replace pattern 24111 with 12345
```

```
[edit policy-options as-path ONLY-TWO-B]
lab@JNCIE-R1# show
".12345{1,4}";
```



I guess that if you counted up the commands used here, it could be faster to just enter a new policy and as-path. Back to the KISS principle, though: typing from scratch is likely to introduce more errors than copying an existing working configuration.

Protocol Configurations

Once the overhead configurations are complete, the working configuration is performed. In the following sections, the BGP and OSPF protocol configurations are defined. Once these are complete for *RTR_1*, phase 1 is just about complete and we can add the other MXs to the mix and see what we have created.

In order to understand the next set of configurations, more of the network has to be shown. **Figure 9-2** shows *RTR_1*, the addressing, and the interfaces to the other MXs and the ISPs. The internal interfaces are carved from the 11.13.132.0/28 address space, while the external addresses are shown as 10. addresses; in reality, they are as defined by the ISPs.

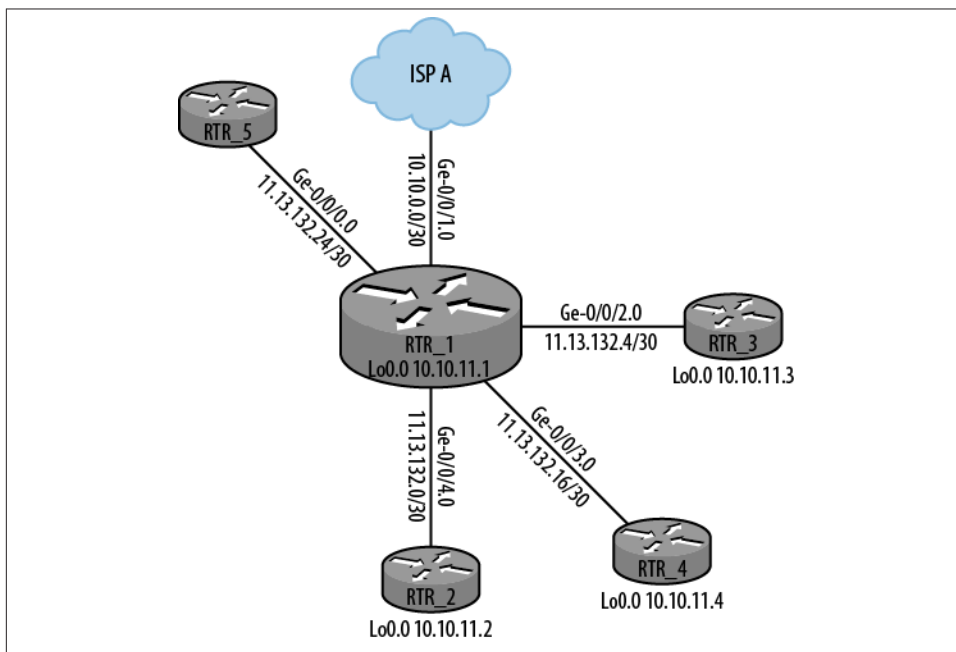


Figure 9-2. *RTR_1* addressing

OSPF

The OSPF protocol is used for internal propagation of routing information between the MX routers. In the future, though, other portions of the network might use this same backbone (administrative traffic, interoffice traffic, etc.). To prepare for this growth, the OSPF area 0 is used for the backbone. This allows other non-backbone areas to interface with the backbone without the need for a total redesign. The OSPF protocol supports authentication (the security requirement) as well.

The configuration for the OSPF protocol is:

```
protocols {
  ospf {
    area 0.0.0.0 {
      interface ge-0/0/2.0 {
        authentication {
          md5 1 key "$9$hWjyeM7-waZj8XZjHqQzhSrlWLxNd";
        }
      }
      interface ge-0/0/3.0 {
        authentication {
          md5 1 key "$9$A0TZuBRreW-VYhSVYgojiAp00EcyLK";
        }
      }
      interface se-1/0/3.0 {
        authentication {
          md5 1 key "$9$7Zdw2ZUH5Qn4aQn/CB17-VbgoJGD";
        }
      }
      interface lo0.0;
      interface ge-0/0/0.0 {
        passive;
      }
    }
  }
}
```

The passive interface is to the “customer” MX; this isn’t part of the OSPF area, but it’s nice to have the address in the other routers’ tables. The authentication protects the network from hackers.

Once the other MXs are programmed, OSPF can be verified as operational.

BGP

The ease of configuring OSPF is countered with the complexity of configuring BGP. Two versions of BGP are run in the network, external and internal. IBGP is used to communicate to the other MXs in the backbone, while EBGP is used to communicate to the customers and the ISPs.

The IBGP implementation that is used is a full mesh: each backbone router forms an adjacency to all the other backbone MXs. If the network were going to grow by leaps and bounds, the use of route reflectors would make sense. But because the growth is going to be in customer routers, a full mesh between the backbone MXs is not going to be a maintenance issue down the road.

The base configuration of the MX for IBGP is:

```
bgp {
  group internal {
    type internal;
    local-address 10.10.11.1;
    export [ NHS LOCAL-OUT ];
    bfd-liveness-detection {
      minimum-interval 200;
    }
    neighbor 10.10.11.2;
    neighbor 10.10.11.3;
    neighbor 10.10.11.4;
  }
}
```

The neighbor addresses are the loopback addresses of the other MXs. Bidirectional forwarding detection (BFD) is turned on for this group. This provides for quicker fail-over in the event of a link or node failure. There are two policies assigned to the group as export policies. The first was discussed earlier, for advertising the aggregate route into BGP. The other is a policy that modifies any routes received from the ISPs. The modification is to change the next hop of the external routes to be the address of the local router. The policy looks like:

```
policy-statement NHS {
  from {
    protocol bgp;
    route-type external;
  }
  then {
    next-hop self;
  }
}
```

The alternative to this policy is to include the ISP interfaces in the OSPF configuration as passive interfaces.

The configuration of the EBGP groups is not much different: the peer AS# is defined, and the neighbor address is the interface address of the directly connected router. Two different EBGP groups are defined—the first is for the ISP (*ISP_A*), and the other is for the customer (*RTR_5*). The configurations are:

```
bgp {
  group ISP_A {
    type external;
```

```

import ISP-PREF-110;
authentication-key "$9$XduNVsaZjP5F245Fn/00X7-dYgoJG";
export UPStream-ISP-Out;
peer-as 14;
neighbor 10.10.0.1;
}
group CUST-A {
type external;
import [ NO-PRIV CUST-A-IN ];
peer-as 24111;
neighbor 11.13.132.26;
}
}

```

The ISP requires authentication of the relationship, so an MD5 key is defined for authentication purposes. There is an import and an export policy for the ISP. The import policy allows both the default route and the full Internet route table and sets the local preference of these routes to be 110. This will force traffic out this ISP. The export policy sends the local and customer addresses to the ISP. To directionalize incoming traffic, the other ISPs will receive a prepended set of prefixes from the network.

The customer group is associated with two import policies. The first blocks all RFC 1918 prefixes, and the second limits the prefixes that have a mask of less than 25 bits (/0 through /24).

The only policy not described in a previous section is *ISP-PREF-110*. This is a combination of the local preference policy and the *INET-DEF* policy:

```

policy-statement ISP-PREF-110 {
term DEF {
from {
route-filter 0.0.0.0/0 exact;
}
then {
local-preference 110;
community add TAG-DEF;
accept;
}
}
term INET {
from {
protocol bgp;
route-filter 0.0.0.0/0 upto /24;
}
then {
local-preference 110;
community add TAG-PEER;
accept;
}
}
}

```

```

    term REJECT-ALL {
        then reject;
    }
}

```

The higher local preference is preferred over a lower, and the default is 100. Adding a local preference of 110 ensures that these prefixes are preferred over the same prefix learned from another source.

A note about the customer group: the lack of an export policy does not mean that the router will not receive BGP routes; on the contrary, the default operation for BGP is to forward all BGP routes to an EBGP peer. So, the customer MX will receive the full feed of routes. If a reduced load is required, an export policy can be created to block prefixes.

With OSPF and BGP configured, we have a template to configure the other backbone MXs. After updating the MX-specific components, *RTR_1*'s configuration can be loaded in the other backbone routers and tested.

Phase 2

The components that have to be updated for the other MXs are:

- The *fxp0* interface addresses
- The loopback interface address
- The internal interfaces (addresses and OSPF)
- The external interfaces (addresses and BGP)
- The exterior BGP groups

The policies and the firewall filters are all generic in nature, except for the customer-specific policies. The use of the apply lists eliminates the manual labor for the prefix lists. By using *show | display set* and Notepad, the configurations for the remaining backbone MXs can easily be created.

Once the configurations were loaded, we were able to verify that BGP was set up correctly with:

```

lab@RTR_A> show bgp summary
Groups: 10 Peers: 16 Down peers: 0
Table  Tot Paths  Act Paths Suppressed  History Damp State  Pending
inet.0      63         23         0         0         0         0
Peer      AS      InPkt   OutPkt   OutQ   Flaps  Last Up/Dwn
State| #Active/Received/Accepted/Damped...
10.10.11.2  24878      497      495       0         1    3:47:14
Establ
  inet.0: 0/1/1/0
10.10.11.3  24878      495      496       0         1    3:47:17
Establ

```

```

    inet.0: 0/0/0/0
10.10.11.4    24878          514          515          0          2          3:56:18
Establ
    inet.0: 0/0/0/0

```

This is a super troubleshooting tool; it shows that all the other components are up and running, including OSPF, interface addressing, authentication, and BGP.

The aggregate interface is being advertised by *RTR_2* as well as *RTR_1*, so the received prefix shown in the 10.10.11.2 line is showing that the policies are working as well.

The best OSPF verification is to look at the OSPF database. It shows the other routers and the Ethernet links in the networks. The command and output from *RTR_1* are:

```
lab@RTR_A> show ospf database
```

```

    OSPF database, Area 0.0.0.0
  Type      ID          Adv Rtr      Seq          Age  Opt  Cksum  Len
Router *10.10.11.1      10.10.11.1  0x80000020  1696  0x22  0xccd3  96
Router  10.10.11.2      10.10.11.2  0x8000001a  1531  0x22  0xa1a8  96
Router  10.10.11.3      10.10.11.3  0x8000001c  1531  0x22  0xa5a4  84
Router  10.10.11.4      10.10.11.4  0x80000022  1567  0x22  0x74c5  84
Network 11.13.132.1        10.10.11.2  0x8000000a  2173  0x22  0x2212  32
Network 11.13.132.6     10.10.11.3  0x8000000a  2173  0x22  0xf339  32
Network 11.13.132.10    10.10.11.4  0x8000000d   44  0x22  0xe53c  32
Network 11.13.132.14    10.10.11.4  0x8000000e  793  0x22  0xad70  32

```

This display shows that the four backbone routers are present and that the interconnecting links are up and running.

Prior to entering phase three and the proof of concept, we decided to test the other policies with a logical network. The fifth MX is logically divided into three ISPs and a customer router (*RTR_5*). The MX is interconnected to the other four MXs via Gigabit Ethernet interfaces. Each ISP offers some /32 prefixes, some /24 prefixes, and a default route. The customer offers a /32 prefix, a nonprivate /24 prefix, and a private /24 prefix.

This range of prefixes will test the effectiveness of the policies to these “external interfaces.” All prefixes are offered to the network as BGP prefixes. The backbone routers should reject these. Below is a shortened list of prefixes being sent by *ISP_A*:

```
lab@ISP_A> show route advertising-protocol bgp 10.10.0.2
```

```

ISP_A.inet.0: 67 destinations, 71 routes (67 active, 0 holddown,
0 hidden)
  Prefix          Nexthop      MED      Lclpref    AS path
* 0.0.0.0/0       Self                I
* 10.10.0.12/30    Self                13 I
* 10.10.12.2/32    Self                13 I
* 10.100.20.1/32   Self                13 I

```

```

* 10.100.30.1/32          Self          13 12 I
* 10.101.10.0/24         Self          I
* 10.111.10.0/24         Self          13 I
* 10.121.10.0/24         Self          13 12 I

```

If the policies are correct, the /24 prefixes should be accepted, tagged, and given a local preference of 110 at *RTR_1*. The default should be treated the same. The other prefixes should all be hidden. This can be verified by using the *show route detail x.x.x.x* command. The default route is verified by looking for a nonexistent route (1.1.1.1):

```
lab@RTR_1> show route 1.1.1.1 detail
```

```

inet.0: 82 destinations, 84 routes (43 active, 0 holddown, 39 hidden)
0.0.0.0/0 (1 entry, 1 announced)
  *BGP      Preference: 170/-111
            Next hop type: Router, Next hop index: 665
            Next-hop reference count: 102
            Source: 10.10.0.1
            Next hop: 10.10.0.1 via ge-0/0/1.0, selected
            State: <Active Ext>
            Local AS: 24878 Peer AS: 14
            Age: 5:14:01
            Task: BGP_14.10.10.0.1+63856
            Announcement bits (3): 0-KRT 4-BGP RT Background
              5-Resolve tree 2
            AS path: 14 I
Communities: 24878:4
            Accepted
Localpref: 110
            Router ID: 10.10.12.1

```

```
lab@RTR_A> show route 10.10.0.12/30
```

```
inet.0: 82 destinations, 84 routes (43 active, 0 holddown, 39 hidden)
```

```
lab@RTR_A> show route 10.10.0.12/30 hidden
```

```

inet.0: 82 destinations, 84 routes (43 active, 0 holddown, 39 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

10.10.0.12/30      [BGP ] 05:14:37, localpref 100
                   AS path: 14 13 I
                   > to 10.10.0.1 via ge-0/0/2.0

```

```
lab@RTR_A> show route 10.101.10.0/24 detail
```

```

inet.0: 82 destinations, 84 routes (43 active, 0 holddown, 39 hidden)
10.101.10.0/24 (1 entry, 1 announced)
  *BGP      Preference: 170/-111
            Next hop type: Router, Next hop index: 665
            Next-hop reference count: 102
            Source: 10.10.0.1

```



```

Next hop: 10.10.0.1 via ge-0/0/1.0, selected
State: <Active Ext>
Local AS: 24878 Peer AS: 14
Age: 5:14:55
Task: BGP_14.10.10.0.1+63856
Announcement bits (3): 0-KRT 4-BGP RT Background
  5-Resolve tree 2
AS path: 14 I
Communities: 24878:2
Accepted
Localpref: 110
Router ID: 10.10.12.1

```

The prefixes sent from the customer router are shown next:

```

lab@RTR_5> show route advertising-protocol bgp 11.13.132.25

RTR_5.inet.0: 32 destinations, 54 routes (32 active, 0 holddown,
0 hidden)
  Prefix                Nexthop      MED      Lclpref    AS path
* 10.0.8.0/24           Self                I
* 11.10.10.0/24         Self                I
* 11.10.10.45/32        Self                I
* 11.10.30.0/24         Self                12345 12356 I

```

These prefixes include a few that are not allowed in the network. The first (10.0.8.0/24) is a private address. The third has too long a subnet mask. The last is from a network that is a customer of a customer of a customer, and constitutes transit traffic. These three prefixes are all hidden on the backbone, as shown in the output of the following commands issued from *RTR_2*:

```

lab@RTR_2> show route hidden 10.0.8.0/24

inet.0: 82 destinations, 107 routes (43 active, 0 holddown, 39 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.8.0/24          [BGP ] 05:05:44, localpref 100
                    AS path: 24111 I
                    > to 11.13.132.30 via ge-0/0/1.0

lab@RTR_2> show route hidden 11.10.10.45

inet.0: 82 destinations, 107 routes (43 active, 0 holddown, 39 hidden)
+ = Active Route, - = Last Active, * = Both

11.10.10.45/32       [BGP ] 07:57:55, localpref 100
                    AS path: 24111 I
                    > to 11.13.132.30 via ge-0/0/1.0

lab@RTR_2> show route hidden 11.10.30.0/24

inet.0: 82 destinations, 107 routes (43 active, 0 holddown, 39 hidden)
+ = Active Route, - = Last Active, * = Both

```

```

11.10.30.0/24      [BGP ] 07:58:04, localpref 100
                  AS path: 24111 12345 12356 I
                  > to 11.13.132.30 via ge-0/0/1.0

```

This verifies that the import policies are working. By looking at the advertised prefixes from *RTR_1*, we can verify the outgoing policies:

```

lab@RTR_A> show route advertising-protocol bgp 10.10.0.1

inet.0: 82 destinations, 84 routes (43 active, 0 holddown, 39 hidden)
  Prefix            Nexthop          MED      Lclpref   AS path
* 11.10.10.0/24     Self              0

```

This shows that the customer prefix is being advertised as required, but what about the aggregate prefix of the local network?

Looking at the export policy shows:

```

lab@RTR_A> show configuration policy-options policy-statement
UPStream-ISP-Out
term Accept {
  from {
    protocol bgp;
    community ISP-OUT;
  }
  then accept;
}
term REJECT {
  then reject;
}

```

That looks right, as the community *ISP-OUT* is either tag 24878:1 (local) or 24878:3 (customer). The problem is not with the community string, but with the protocol definition. On *RTR_1*, the aggregate route is advertised to the internal group, but not to the ISP group.

There are two ways (aren't there always?) to fix this situation. The first is to modify the *UPStream-ISP-Out* policy to include aggregate routes. That seems like a pretty harsh solution, because it voids the generic policy guidelines. The second solution is to include the *LOCAL-OUT* policy for this ISP. The other locations' backbone MXs will see the aggregate as a BGP route and forward it with the basic policy.

Unfortunately, adding the *LOCAL-OUT* policy to the ISP group does not solve the problem:

```

[edit protocols bgp group ISP_A]
lab@RTR_A# show
type external;
import ISP-PREF-110;

```

```

authentication-key "$9$XduNVsaZjP5F245Fn/00X7-dYgoJG";
export [ UPStream-ISP-Out LOCAL-OUT ];
peer-as 14;
neighbor 10.10.0.1;

```

Looking at the *UPStream* policy again shows what the problem is—the policy contains the *reject-all* term. Reordering the policies with this command:

```

[edit protocols bgp group ISP_A]
lab@RTR_A# insert export LOCAL-OUT before UPStream-ISP-Out

```

gives us the solution that is required:

```

lab@RTR_A> show route advertising-protocol bgp 10.10.0.1

inet.0: 82 destinations, 84 routes (43 active, 0 holddown, 39 hidden)
  Prefix                Nexthop          MED      Lclpref   AS path
* 11.10.10.0/24         Self              24111    I
* 11.13.132.0/22        Self              I

```

Sometimes we really earn the title of Juniper Networks warrior—we have to attack a problem using all of our skills, until we find the required solution. When one approach does not bear fruit, we switch tactics and try again. Ninety-nine times out of a hundred, the problem can be resolved with a configuration change. The 1% of cases where this isn't possible often requires an escalation to JTAC for a resolution, but as warriors, we don't give up easily; we try our hardest to find a solution before taking that route.

Final Phases

The final phases of this project were also accomplished successfully with the assistance of this warrior team. Our time on the project provided a strong foundation for a successful deployment. The configuration is now in place, the KISS principle is in use, and the knowledge of how to modify the configuration for the various customers and ISPs is a known entity.

Conclusion

In this engagement, we went from dark boxes to a deployable ISP in a short time. Hopefully, this regional ISP will survive and provide superb service to its customer base. We warriors like a challenge, but we also like a success!

A Survivable Internet Solution for a Fully Distributed Network

Any engagement is a study in trial and error, or even trials and errors. Network warriors know that not every battle is won, and that when setbacks are encountered, you do not quit. For every option that fails, a lesson is learned that can be used to come to a successful conclusion. For this engagement, the tribe suffered its losses, but in the end combined portions of each of the failures were used to solve the client's problem.

The problem here was to provide survivable Internet access for a client that had deployed a fully distributed network infrastructure. The problem was solved after two OSPF-based solutions were explored (and discarded) and a static routed solution was tested and deployed. While each of the solutions was technically feasible, Layer 8 of the protocol stack ("politics") raised its ugly head and thwarted the first two options. In the end, the solution that was acceptable from a technical standpoint *and* a political standpoint was one of the most simple as well. It is amazing how often the KISS principle shows up.

Original Network Architecture

A simplification of the original network topology is shown in [Figure 10-1](#); it includes the addressing information and the major connectivity devices. For obvious security reasons, the actual network topology, site names, and addresses have been changed. I'll refer to the company as *King Capital*. King Capital operates a number of geographically dispersed service companies, and there are four core centers: one on the West Coast, two in New England, and one in the Mid-Atlantic region. Each center provides corporate support to a number of storefront locations in its area. Each core location is its own company presence on the Internet.

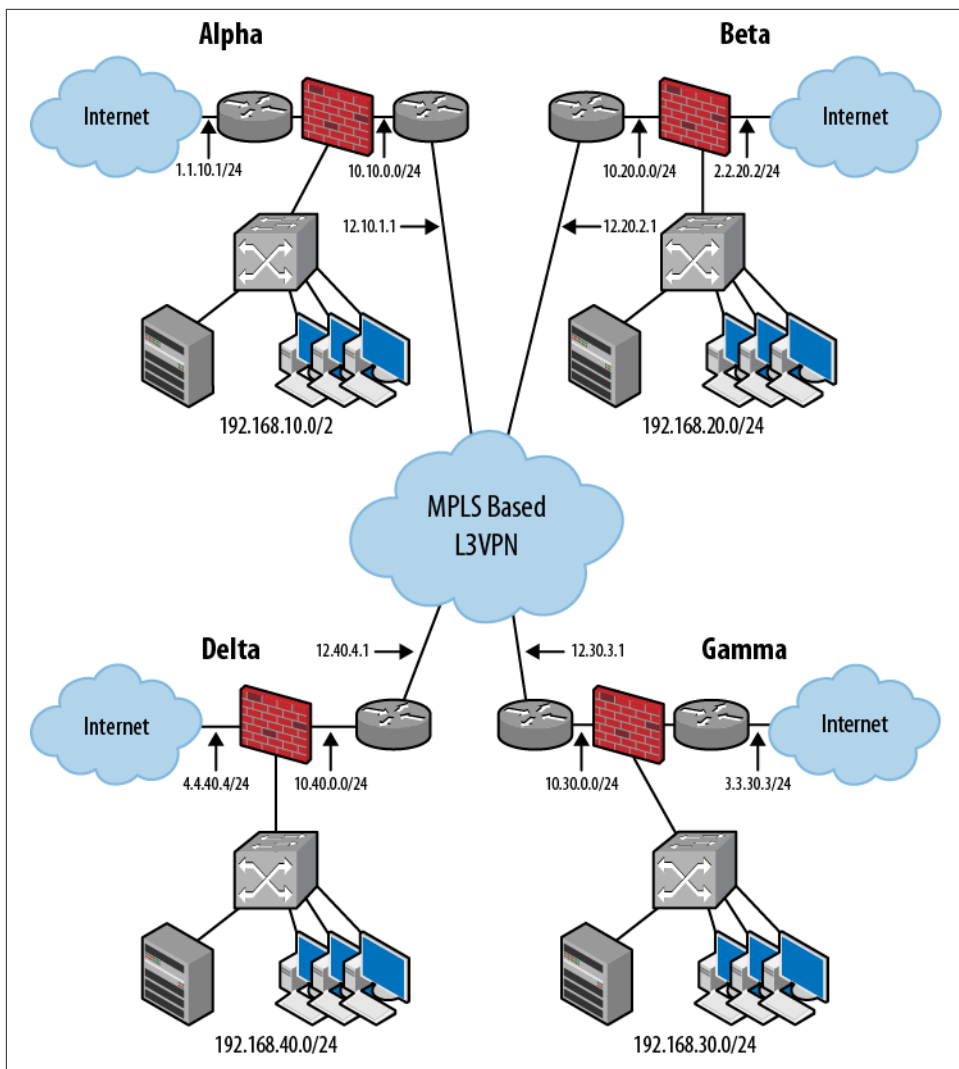


Figure 10-1. Existing network topology

WAN Connectivity

Each core location had corporate employees and enterprise servers (DNS, email, etc.), as well as Internet connectivity. The four core locations were interconnected via a provider-provisioned and -managed Layer 3 virtual private network (L3VPN). The service provider's MPLS-based wide area network provided IP connectivity between the sites at rates from 1.5 Mbps to 45 Mbps. Two of the sites housed data centers, with their

associated server farms. Each data center site also terminated IPSec VPNs from the storefront locations around North America. Each King Capital storefront location was configured with two VPN tunnels, one to each of the data centers. The connectivity between the storefront locations and the data centers was via the Internet.

Addressing

The IP addressing used in the network was from the RFC 1918 address groups (10.0.0.0 and 192.168.0.0). Considering the size of the enterprise (on the small side of the Small and Medium Business segment), there was no need to conserve address space. Each core location used a 10.0/24 address block for the WAN links to the L3VPN routers. The internal addresses were from the 192.168.0/24 address blocks. The private addresses were translated to public addresses by the firewall at each location for accessing the Internet. Incoming Internet traffic to the public servers was statically mapped to an internal private address.

Internal Connectivity

The internal communications at each core location used a combination of routing and switching (both wired and wireless). A Juniper Networks Netscreen firewall was used for Internet access services (network address translation and security policies). Cisco Systems routers were used for access to the L3VPN and, in two locations, access to the Internet. The routers were part of the WAN connectivity package and were managed by the service providers (either L3VPN or Internet).



King Capital had no access to the routers for configuration changes (see **“Solution 1 Issues”** (page 330) below). Due to the contracts that the net-working team had signed, this arrangement was a constraint that could not be resolved.

The L3VPN routers were advertising the local 10. and 192.168 addresses to the other locations over the L3VPN using BGP. This provided full interconnectivity between locations. Each L3VPN router was also part of the OSPF area of the local network. The BGP learned routes (from the L3VPN) were redistributed into OSPF by the L3VPN routers. The L3VPN routers also provided class of service processing for the WAN traffic.

The two managed Internet routers were not part of the OSPF network and provided static and default routes for Internet traffic. These routers did not perform NAT or have public addresses on either the inside or the outside interfaces. They relied on the Netscreen firewalls to provide translation and security services.

Firewalls

The Juniper Systems Netscreen firewalls were configured as screening devices for inter-location traffic as well as Internet traffic. They provided NAT and PAT (port address translation) for Internet-bound traffic and mapped IP addresses for inbound traffic for the public servers at the data center locations.

Each of the firewalls was configured with three security zones: the MPLS zone (L3VPN), the trust zone (users and servers), and the untrust zone (Internet). The basic firewall rules allowed unrestricted access between the trust zone and the MPLS zone, unrestricted outgoing access between the trust zone and the untrust zone, and restricted access to selected addresses (e.g., public servers) from the untrust zone.

Address translation was performed with either interface-based PAT or mapped IP (static) addresses. The interface-based PAT was defined for normal outgoing Internet users: their internal private addresses were translated to the public address of the firewall's Internet-facing interface. For incoming and outgoing Internet access for selected users, the static "mapped" address translation was used. The mapped NAT supported direct access to "public" servers and allowed selected users to keep a constant IP address for outgoing Internet access.



The source IP address is used as part of the authentication mechanism in certain environments in the financial industry, in a manner similar to your home phone number being used by credit card companies. The devices that use these services have to have a static public IP address.

Although not shown on **Figure 10-1**'s topology, the company utilized the services of an anti-spam application service provider (ASP). The anti-spam ASP monitored and stored email traffic destined to any of the four email servers found in the King Capital network. The public DNS MX records for King.com pointed to the IP address of the anti-spam server, and the anti-spam server forwarded appropriate traffic to the public addresses of the various King.com email servers. With this arrangement, all email was first filtered by the anti-spam server and then relayed to the appropriate King server.

King Capital maintained multiple email servers (part of the autonomous operation of each core location). When email arrived at the anti-spam server, a secondary DNS lookup was performed and the correct internal server was identified. The specific email server's public address was then used to route to the appropriate core location's Internet firewall, where the address was statically mapped to the internal private address of the email server.

Problem Definition

The employees of the companies of King Capital used the resources of the Internet on a daily basis. While each core location of the company had an independent network, only the Alpha site actually had an IT staff. The other locations had to fend for themselves for support. This situation is common in small businesses where communications is not the primary focus. This lack of allocation of IT resources led to the need to bring in experts to change the normal operation of the network.

The problem started one afternoon when one of the core locations lost Internet connectivity. This location was not one of the data centers, and it had no IT staff for support. As is normal when people are busy with their daily tasks, no one reported the incident to the IT manager. As fate would have it, one of the C-level officers of the company happened to be visiting the office that day. He attempted to access the Internet from his PC and found, much to his dismay, that the site was without Internet service. When he asked the people working at that location about the outage, they responded that they did not know how long the Internet had been down, but that it was a royal pain, and could he get it fixed? This interchange eventually led the IT manager to call in the Juniper Networks warriors to assist in cleaning up the mess.

Our task was to implement a survivable solution for Internet access across all the core sites.

The desired solution was to have each location use the Internet access of a neighbor location as a failover site. The requirements for Internet survivability were:

- The interconnectivity of the L3VPN WAN would provide the cross links to access the Internet.
- The failover should be as automated as possible.
- When Internet access was restored at the location, the traffic should return to using the local access.
- The switchover and switch back should be totally transparent to the users.
- The total time to detect a failure and switch over should be kept to a minimum.
- Each data center should provide backup access for one of the other locations and the other data center.

Figure 10-2 illustrates the desired failover scenario. Here, the Beta site is providing alternate access for the users of the Alpha site. When the Internet link at site Alpha goes south, Internet traffic will be routed across the L3VPN link to site Beta. The firewall at site Beta will be responsible for handling the NAT and security policies for not only the Beta traffic, but also the Alpha traffic that is present.

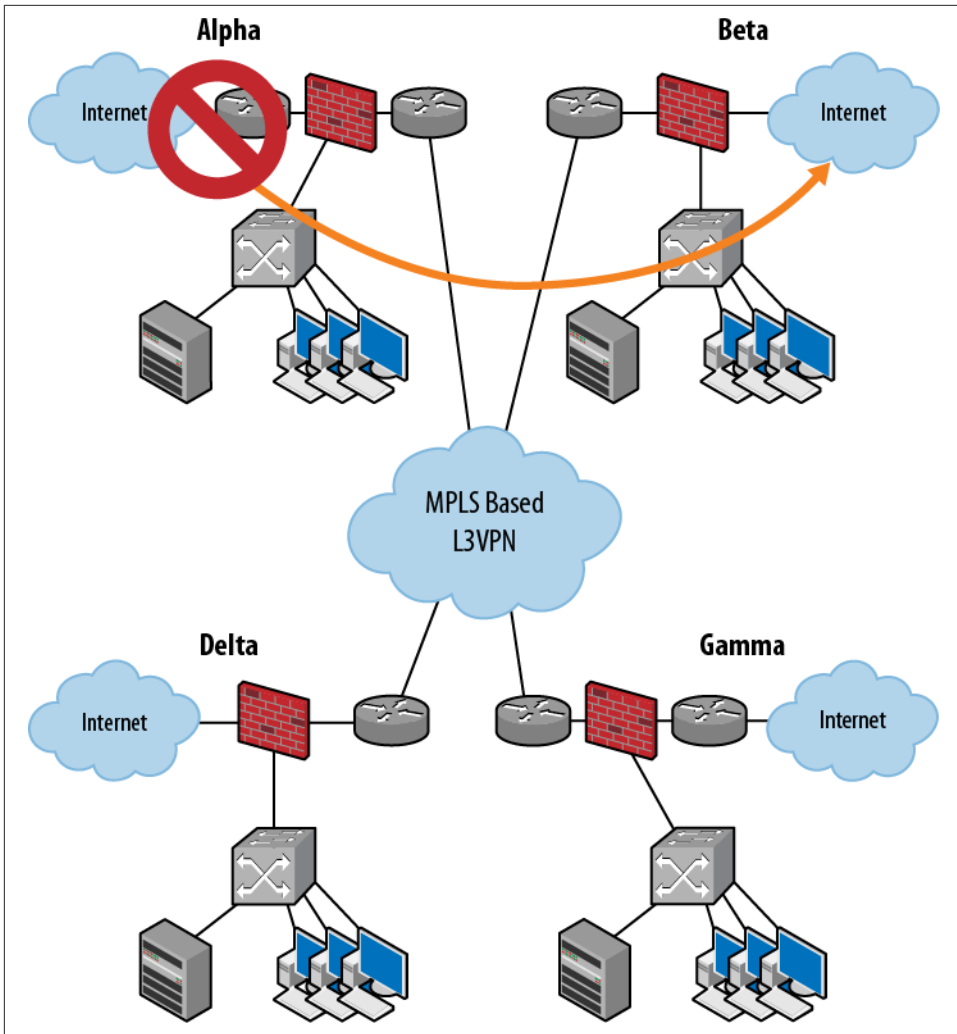


Figure 10-2. Failover scenario

A problem immediately became evident, though. Outgoing traffic to the Internet from the Alpha site and the Beta site used the firewall's interface address. The static mapped traffic from site Alpha was not supported at Beta, and no clean method could be found to restore these users without impacting the Internet routers (and that was not going to happen).

At first glance, the solution to the problem was a simple one: have each core site send a default route to all the other sites. Each site would pick the closest Internet access location. Sometimes warriors get arrogant and jump to solutions, ignoring the full ramifications of what they are proposing. This often leads to a half-baked solution that cannot be used in a production environment. Such was the case here, and I'll tell you why.

Proposed Solution 1

The initial proposed solution was to use the existing routing protocols to advertise the primary and alternate Internet access points. In general, the static routes can be redistributed into other routing protocols (OSPF and BGP). This redistribution allows the default routes (0.0.0.0/0) to be advertised to other routers in the network and between networks. The redistribution can also contain metrics that favor one advertised route over another.

In the case of King Capital, we proposed that the default routes from the Internet routers would be advertised to the local network via the OSPF protocol. At the two data center locations, this default route would also be redistributed into the L3VPN's BGP protocol, as shown in [Figure 10-3](#). The redistributed default route would be learned by all the other sites and would be used as the alternate Internet access in the event of the loss of the local Internet access.

Metrics were added to the default routes to create a predictable failover plan. The lower the metric, the more favorable the route, allowing the receiving site to choose the primary and secondary recovery Internet access sites. Each site would receive multiple default routes (one from each backup site and the local route); the routers would use the route with the lowest metric as the primary access, the next lowest as the failover site, and the next higher in the case where multiple failures occurred.

The routing table would look like this for each location (with the next hop and the addresses altered based on the sites):

IPv4 Dest-Routes for <trust-vr> (3 entries)

ID	IP-Prefix	Interface	Gateway	P	Pref	Mtr	Vsys
* 2	0.0.0.0/0	e0/2	2.2.2.3	0	0	5	Root
* 4	0.0.0.0/0	e0/1	10.1.1.2	0	0	15	Root
* 3	0.0.0.0/0	e0/1	10.2.2.2	0	0	25	Root

The first entry would be used until it was withdrawn because of a failure; in that case, the second entry would then be used. If that entry also had a failure and was withdrawn, then the third entry would be used for Internet access.

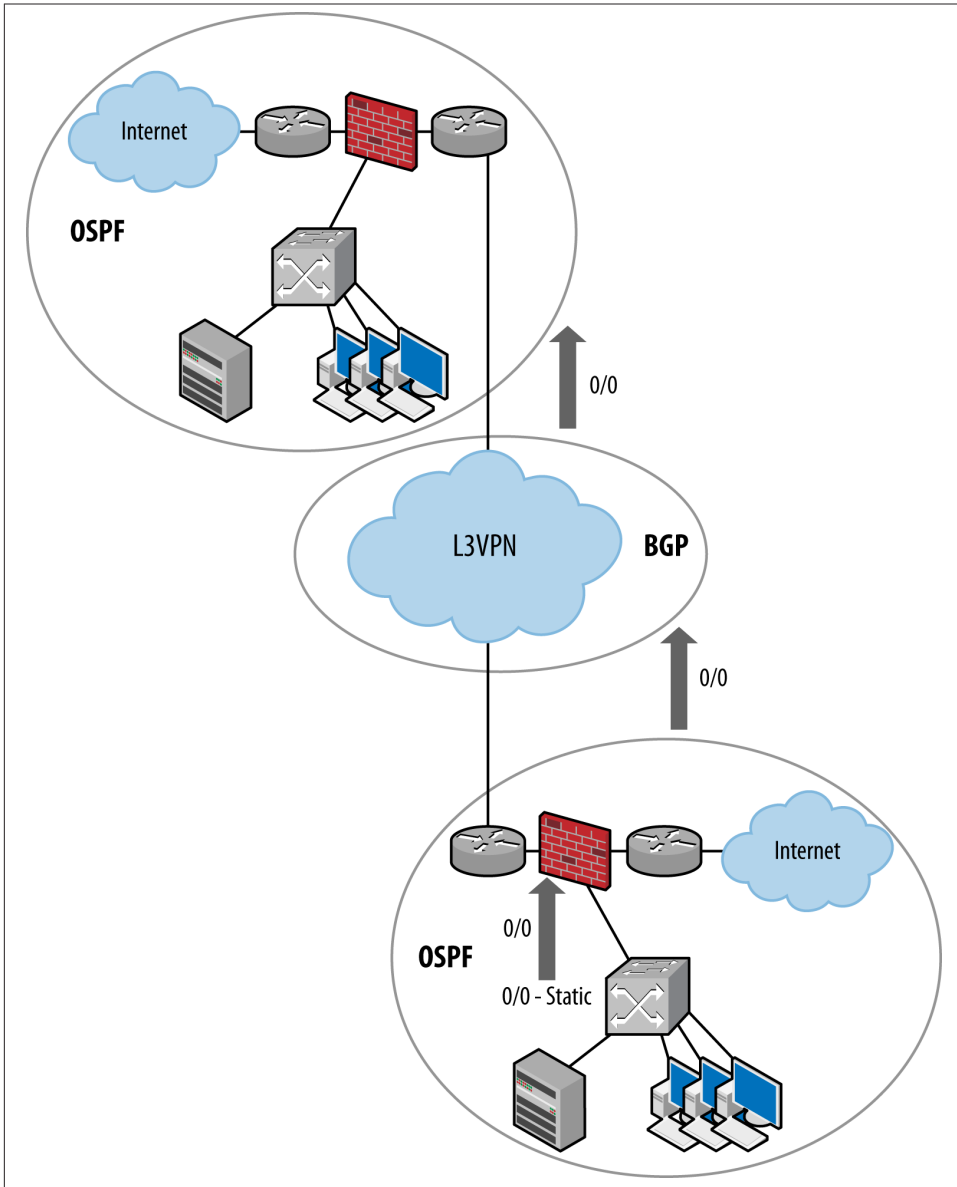


Figure 10-3. First solution—route redistribution

When the alternate Internet access was active, all traffic would use the same NAT rules and return routes as the traffic from the host site (e.g., Alpha traffic would use the Beta NAT rules, assuming that Beta was processing Alpha's traffic because Alpha's Internet link had failed).

This solution required a number of changes to the equipment at the sites:

- The static default route that was currently used on the Juniper Systems Netscreen firewalls would be deleted.
- All default routes would be learned.
- Firewall policies would be added to allow traffic to pass from the MPLS zones to the untrust (Internet) zones.
- The managed router would be added to the OSPF area.

Solution 1 Advantages

The initial proposed solution provided three advantages for King Capital:

- An automated alternate site selection capability that could be tuned to direct alternate access to predetermined sites.
- The ability to handle multiple Internet access failures. If all the sites advertise default routes and all are ranked properly, the failover process will find an alternate location even in the event of multiple failures.
- Alternate routing. This solution allows traffic to follow the existing routes for intersite communications as well as for failover Internet access.

Solution 1 Details

The configuration changes to the routers are shown below in the Cisco IOS syntax. The proposed configuration changes to the L3VPN routers were these additions:

```
Router ospf 1
  Default-information originate metric 10
Router bgp 65001
  Network 0.0.0.0
```

The additions proposed to the ISP managed router were:

```
Router ospf 1
  Area 0.0.0.0
  Default-information originate
  Redistribute static
  Network 192.168.10.0 0.0.0.255
```

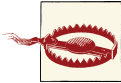
These changes would have allowed the routers to advertise the default route into the OSPF routing protocol. If the route became inactive, it would be withdrawn from OSPF.

At the firewalls, multiple default routes would be seen. Only the most attractive would be active; the others would not be feasible successors.

Solution 1 Issues

While technically sound, the initial solution had a large political flaw that caused it to be unusable for King Capital. The flaw was the realization that the WAN service provider was unwilling to modify the configuration of the managed router. There was no provision in the management contract to allow the router to be part of the OSPF area or to redistribute the default route.

Call it wild abandon or maybe a desperate desire to solve our client's problems, but we warriors had totally forgotten about the restrictions on the managed routers.



Be very careful around managed devices—the teams that manage them are very protective of the devices and the configurations within them. Even though we could have modified the configuration to work in our solution, we were told, in no uncertain terms, that we were not allowed to do anything—ANYTHING!!!!—to the device. Yes, we were actually yelled at and told not to solve the customer's problems.

After the fact, another tribe member proposed a solution to this issue. The Juniper Netscreen firewall has a monitoring capability for static routes. When a route is defined, an IP address associated with that route can be identified to be monitored. When the IP address is not reachable, the route is withdrawn.

In our scenario, each firewall would have had a local default route and a local IP address that determined whether the default route was still active. For example, if you refer to [Figure 10-1](#), the firewall at site Alpha would have a default route to the local Internet router, but monitor (send and receive pings to and from) 1.1.10.2 (the IP address of the other end of the Internet access link). When a failure occurred at the Internet router (link or node failure), the local static route would be withdrawn from the route table. This solution would allow OSPF to provide the alternate routes and not require changes to the managed routers.

Proposed Solution 2: OSPF over Tunnels

After discussing the first solution and having it shot down, we decided to revisit the design to eliminate changes to the managed routers. The next proposed solution was to use generic routing encapsulation (GRE) tunnels to connect the Juniper Systems Netscreen firewalls. The GRE tunnels would form a secondary mesh between the sites. This mesh would be used to pass the default route information between sites. A depiction of the proposed solution is shown in [Figure 10-4](#).

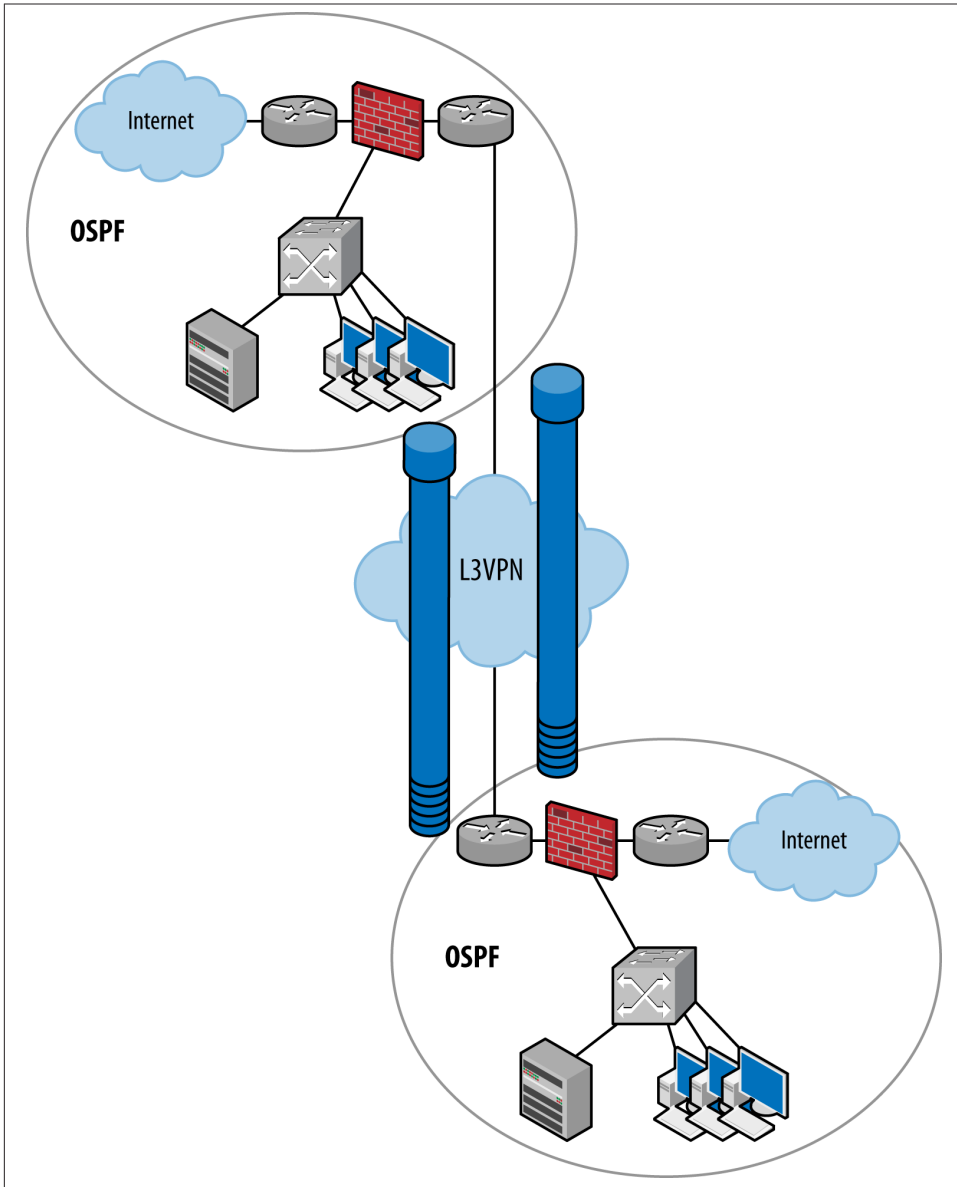


Figure 10-4. OSPF over tunnels

Early Death of Solution 2

When solution 2 was prototyped in the lab, the initial tests showed that it was feasible. But when the MPLS core was added to the prototype, some issues appeared that caused this solution to be shelved:

Dual routes

The first issue was that routes were being learned via the tunnels and via the L3VPN routers. This caused some traffic to be lost in routing loops and other traffic to be dropped at the firewall because of asymmetrical return paths.

Route preference

The solution to the dual route problem was to add route preferences to the learned routes. The L3VPN routes would have a higher preference than the tunnel routes, so only when the L3VPN routes were withdrawn would the tunnel be used. Unfortunately, the route preference modification had to be performed on the Netscreen firewalls as well as the managed routers. We were back to square one again.

Configuration for Solution 2

Shown here for completeness are the configuration details for the tunnel solution. The configuration was for changes to the Juniper Netscreen firewalls. The changes to the Cisco-managed ISP routers were the same as those seen in the previous solution.

On the firewalls, a GRE tunnel was created and included in the OSPF network. The syntax of this configuration is in ScreenOS, the operating system of the Netscreen firewalls:

```
set interface tunnel.3 zone trust
set interface tunnel.3 ip 192.168.23.2/24
set interface tunnel.3 tunnel encap gre
set interface tunnel.3 tunnel local-if ethernet0/1 dst-ip 192.168.20.1
set interface tunnel.3 protocol ospf area 0.0.0.0
set interface tunnel.3 protocol ospf cost 100
set interface tunnel.3 protocol ospf enable
```

A companion configuration was created at the far-end firewall pointing to this end. Each data center site had a tunnel to each of the other core locations, and one to the other data center. While this solution worked for the limited number of locations at King Capital, it is not a scalable solution for any larger enterprise, as the number of tunnels would be unmanageable.

Final Solution: Static Routes over Tunnels

After suffering two defeats, the warriors regrouped, strategized, and designed a third solution to the problem. This solution was an evolution of the previous solution, keeping the advantages and throwing away the disadvantages. The solution was implemented in the customer's network and worked as planned.



I know I've stated this before, but I want to emphasize it again: in our line of work, there are many ways to solve a problem. No one engineer has all the right answers, and that is why we work as a tribe, a group of like-minded individuals who combine forces to solve complex problems. In this case, the solution took three attempts to get it right. However, the focus should never be on the number of attempts, but rather on the "get it right" part of the equation.

For this client, the intermediate steps provided a means to arrive at the ultimate solution. They were not failures, but learning exercises that allowed us to come to a workable solution.

The implemented solution was a modification of the OSPF over tunnels solution discussed previously. In this third and final solution, each Netscreen firewall was programmed with a primary static default route and a secondary static default route. The secondary route had a higher metric than the primary. The secondary default route pointed to a GRE tunnel to the alternate ISP access site. In the event of a failure of the primary default route, all Internet traffic was passed over the GRE tunnel to the secondary location and used the default route at that location.

Multiple failures were to be handled by preplanning which sites would be used as a backup. If sites Alpha and Gamma were the data centers, Beta would alternate to Alpha and Delta to Gamma. If the Internet at either Alpha or Gamma were to also fail, then the traffic would be forwarded to the alternate Internet access providers for those sites. The topology of the final solution is shown in [Figure 10-5](#).

The configuration of the GRE tunnels for the Netscreen firewall was:

```
set interface tunnel.10 zone L3VPN
set interface tunnel.10 ip 192.168.23.2 255.255.255.0
set interface tunnel.10 tunnel encap gre
set interface tunnel.10 tunnel local-if ethernet0/1 dst-ip 10.10.0.1
```

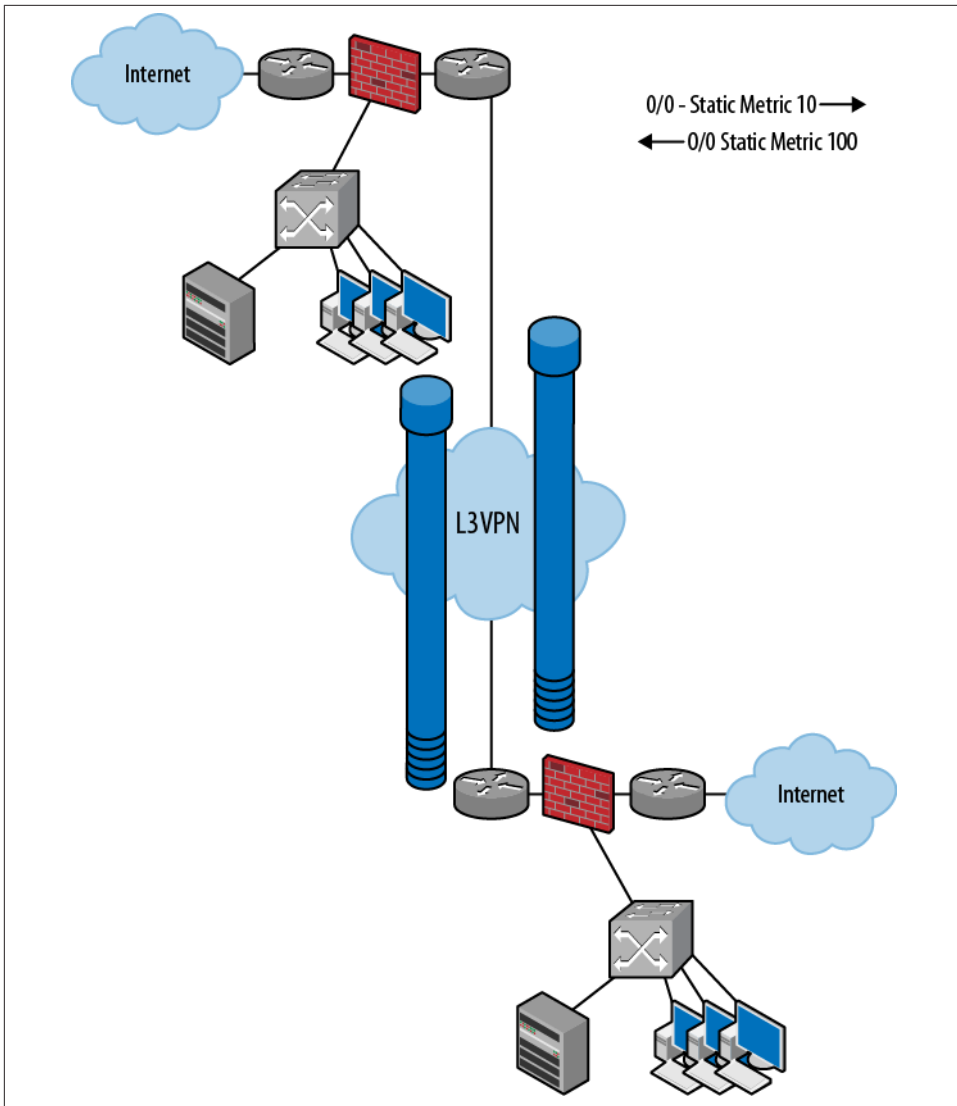



Figure 10-5. Final solution

Solution Advantages

The proposed solution provided a number of advantages for the client:

Predictable protection

With the use of static routes and GRE tunnels, the traffic patterns and failure scenarios could be preplanned and implemented in a manner that allowed the client to know where the traffic was going to be directed for each failure scenario.

Multiple failure protection

The solution did not look to resolve only a single point of failure, but was extended to include the possibility of multiple link failures.

Netscreen solution

The solution only required modification of the Juniper Netscreen firewalls. The managed routers for the Internet and the L3VPNs were not modified for this solution.

Quick convergence

The failover from the primary to the secondary default route took, on average, under 10 seconds. Most of this interval was due to the timeout of the monitoring capabilities of the firewall.

Solution Issues

The solution presented a number of issues that were able to be resolved with the extensive feature set of the Juniper Networks Netscreen firewall, including:

- Reverse path forwarding (RPF) checks
- Default gateway failure detection
- Email server address resolution

For each of the issues that arose, a solution set was devised and implemented.

RPF checks

The first issue was the ability to allow traffic to pass the reverse path forwarding checks that are part of the ScreenOS processing. In the normal operation of the Netscreen firewall, if a packet is received on an interface and the reverse path forwarding check shows that another interface is used to return to the source of the packet, the packet is dropped. This feature prohibits spoofed traffic from entering the network. The RPF check is performed on the source address of the packet. The source address is looked up in the routing table, and the interface of the next hop for that address is found. If the actual interface of the packet is different from the RPF check interface (from the routing table), then the packet could be spoofed and should be discarded.

In our solution, during a local Internet failure all Internet-bound traffic is sent over the GRE tunnel to the remote site, as designed. The remote Netscreen firewall performs the RPF check and sees that the source address should be seen on the interface to the L3VPN router, not the GRE tunnel. As per the normal processing, this traffic is dropped as spoofed traffic and a potential hazard.

To resolve the issue, the policy-based routing (PBR) feature of the firewall was used. A policy-based route supersedes the destination-based routes (and RPF checks) in

ScreenOS. A policy-based route is essentially an access control list (an *if/then*-type statement) that defines how traffic that matches the *if* statement is routed, in the *then* statement. A policy-based route can be applied to an interface, a zone, or a virtual router. In this case, the PBR is applied to the interface in the untrust zone that connects to the Internet. The policy queries the destination address of the incoming packets and directs the appropriate packets to the correct tunnels. This solves the reverse path lookup issue. A sample configuration for the policy-based route was:

```
set vrouter trust-vr access-list extended 1 dst-ip 192.168.40.0
255.255.255.0 entry 1
```

The above access list defines a destination address as a match condition. This address identifies the originating site of the traffic (the one that had the local Internet failure).

The access list was referenced in a match group. The match group forms the *if* portion of the *if/then* policy:

```
set match-group name Internet
set match-group Internet ext-acl 1 match-entry 1
```

The action group was the *then* portion of the policy:

```
set action-group name Internet
set action-group Internet next-interface tunnel.10 action-entry 1
```

Traffic that matched the *if* statement then used a next hop of the *tunnel.10* interface (our GRE tunnel to the originating site). Finally, the match group and the action group were combined into a policy. That policy (named *Internet*) was then applied to the Internet interface:

```
set pbr policy name Internet
set pbr policy Internet match-group Internet action-group Internet 1
set interface ethernet0/2 pbr Internet
```

Default gateway failure detection

Static routes provide a predictable routing method but have the shortcoming of being static. On most devices, if a static route points to a local interface, and that interface fails, the static route will be withdrawn from the route table. But if a static route points to another router and the link on the far side of that router fails, the static route remains active. [Figure 10-6](#) shows an example of this situation.

If router 1 has a static route to the Internet and the link between the Internet and router 1 fails, the static route will be withdrawn—no problem. The scenario that creates the issue here is that the firewall has the static route pointing to router 1 for Internet access. If the link between the firewall and router 1 fails, the static route will be withdrawn and

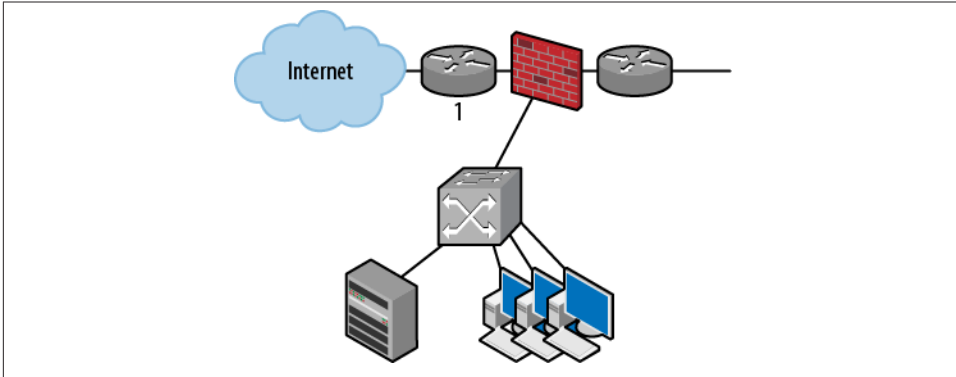


Figure 10-6. Static route monitoring

the alternate Internet route (GRE tunnel) will be used. But what if the link between the Internet and router 1 fails (a much more likely event)? The firewall will not see the link failure and will continue to process Internet traffic to router 1 (only for it to be discarded).

What is needed is a means to monitor the link to the Internet, or better yet, the full path to the Internet from the firewall. If the path, or any component along that path, fails, then the firewall can withdraw the local route and use the GRE tunnel.

ScreenOS incorporates such a monitoring feature, called *track-ip*. The feature sends ping traffic out the monitored interface to a destination. The responding ping packet is monitored by the firewall. If the ping response fails, all routes associated with the interface are withdrawn.

To prohibit flapping of an interface due to intermittent ping failures, timers and thresholds can be configured. These configurable parameters allow the administrator to determine when a failure is detected. The configuration of the *track-ip* monitor to a single IP address is:

```
set failover enable
set failover auto
set failover type track-ip
set interface eth0/2 monitor track-ip
set interface eth0/2 monitor track-ip ip 4.2.2.2 interval 5
set interface eth0/2 monitor track-ip ip 4.2.2.2 threshold 2
```

This example shows that pings will be sent every five seconds and that two successive pings must fail for the route to be withdrawn. With a default ping timeout of two seconds, that means the alternate default route would be active after about four seconds. Not a bad failover time—but what if Level3's server was down and the Internet connection was still good? This is considered a false positive and should be avoided. Why swap traffic when no failure has occurred?

To reduce the possibility of false positives, ScreenOS allows the tracking of up to four addresses for a single interface and the assignment of weights for those addresses. The four addresses are pinged at the same time. If all four, or for that matter, three of the four fail at once, it can be assumed that the path to the Internet is down. This is a more assured approach than monitoring a single address.

With multiple addresses, weights can be assigned to put more importance on some addresses. In this engagement the warrior tribe did not believe we needed that complexity, but it was nice to know it was there. The addresses we chose to monitor were:

4.2.2.2

The Level3 DNS server, as a global resource

75.75.75.75

The Comcast DNS server, as another global resource

1.1.10.2

The local ISP's public address (assigned per site)

132.43.12.101

The anti-spam email service address

For our implementation, the weights for each address were equal, and only three of the four had to fail for the Internet path to be declared inactive. The configuration for this monitoring was:

```
set failover enable
set failover auto
set failover type track-ip
set interface ethernet0/2 monitor track-ip ip 4.2.2.2 weight 1
set interface ethernet0/2 monitor track-ip ip 75.75.75.75 weight 1
set interface ethernet0/2 monitor track-ip ip 1.1.10.2 weight 1
set interface ethernet0/2 monitor track-ip ip 132.43.12.101 weight 1
set interface ethernet0/2 monitor track-ip threshold 3
set interface ethernet0/2 monitor track-ip
```

The monitoring capability on the firewall allows us to view the current status and the configured parameters with the *get interface* command for the monitored (Internet-facing) interface. An example output is:

```
get interface ethernet0/2 track-ip
```

ip address	interval	threshold	wei	gateway	fail-count	success-rate
4.2.2.2	1	1	1	1.1.2.3	0	100%
75.75.75.75	1	1	1	1.1.2.3	0	100%
1.1.10.2	1	1	1	1.1.2.3	0	100%
132.43.12.101	1	1	1	1.1.2.3	0	100%

threshold: 3, failed: 0 ip(s) failed, weighted sum = 0

As the tracked IP addresses fail, the failed count increases until the weighted sum equals the threshold. At that point, the interface is considered down for routing purposes, and

all routes that use the interface (static default route) are withdrawn. This arrangement gave us confidence that when the Internet path failed, the failure would be detected in a couple of seconds. This arrangement also lowered the false positive results, as three of the four addresses had to be down prior to the path being considered down.

In the event of a failure of the gateway (1.1.2.3 in the example above) or another component in the path, the firewall will continue to send pings to the interface. When the tracked addresses are again reachable, the interface will be considered up and the default route reinstated.



ScreenOS allows not only the monitoring of IP addresses, but also the monitoring of another interface. Consider the example shown in **Figure 10-7**, where a static Internet gateway route is being redistributed in a routing protocol and there are two interfaces to the core, a high-speed interface and a low-speed interface. If the high-speed interface goes down, the default route will still be advertised to the network, but now all the traffic to the Internet will pass through the low-speed interface.

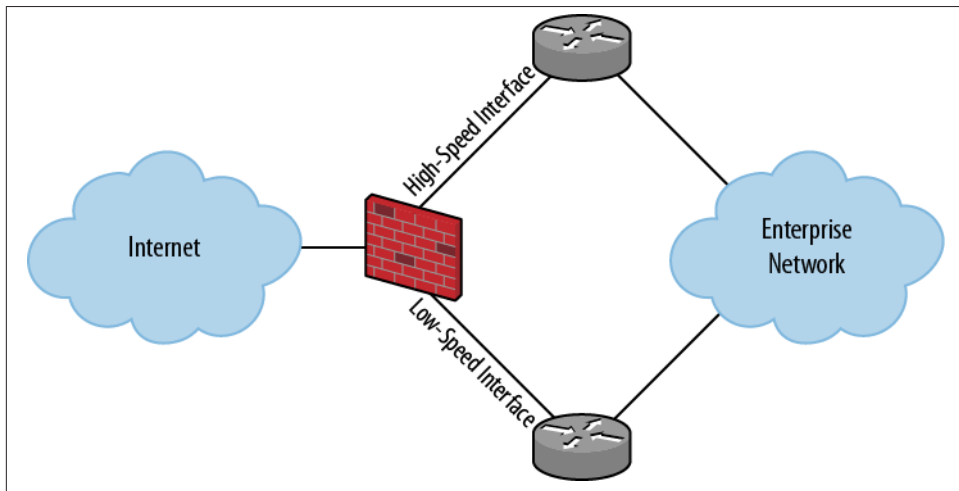


Figure 10-7. Interface monitoring



If an alternative gateway to the Internet exists, using that route would be a better choice than using the low-speed link to reach the Internet.

In this example, the gateway interface could monitor the high-speed interface. In the event of the failure of the high-speed interface, the static route to the Internet would be withdrawn, and the alternative gateway would be used until the high-speed link returned to service.

Email Server Address Resolution

The last issue to be resolved related to the email servers at each of the core locations. As stated previously, each core location had a local email server and local public web servers. When a core location lost its local Internet access, it was expected that the access to the local web server would also be lost. The machinations required to reroute the local public IP addresses and/or update the DNS records dynamically based on the status of the Internet links were not deemed worth the investment. The email servers were a different subject altogether.

Under normal circumstances, the local email server has a static IP address that is used for outgoing and incoming messaging. During a failure of the local Internet link, all that traffic is handled via a neighbor's firewall. During the initial failover testing, the email servers could send mail but could not receive mail. The outgoing traffic from the servers was being handled as normal traffic. The servers were receiving a public address from the alternate firewall's exterior interface. This allowed them to connect to an external email server and exchange messages. For incoming traffic, if the local email server established a session with an external server, the local server could receive the traffic from that external server (in an Internet link failure condition). This arrangement provided service, but at a reduced level. The time it took to receive an email depended on when outgoing traffic was being sent. This obviously was not the proper means of providing email service.

The resolution of the incoming mail delay problem was brought to the table by a drafted tribe member, the local IT manager. King Capital was using an Internet-based anti-spam service provider. The service intercepted mail for all of King Capital's email servers, filtered the spam, and forwarded the remaining traffic to the correct corporate email servers.

One of the value-added capabilities of this service provider was the ability to list multiple public IP addresses for each customer's email servers. If a connection to the initial entry timed out, the secondary entry was used.

We used this capability to allow the server to have both a public IP address from the primary site, and a different static public address at the alternative Internet access location. Mapped IP translations on the firewalls at each of these locations converted the public address to the single private address of the email server. A diagram of this resolution is shown in [Figure 10-8](#).

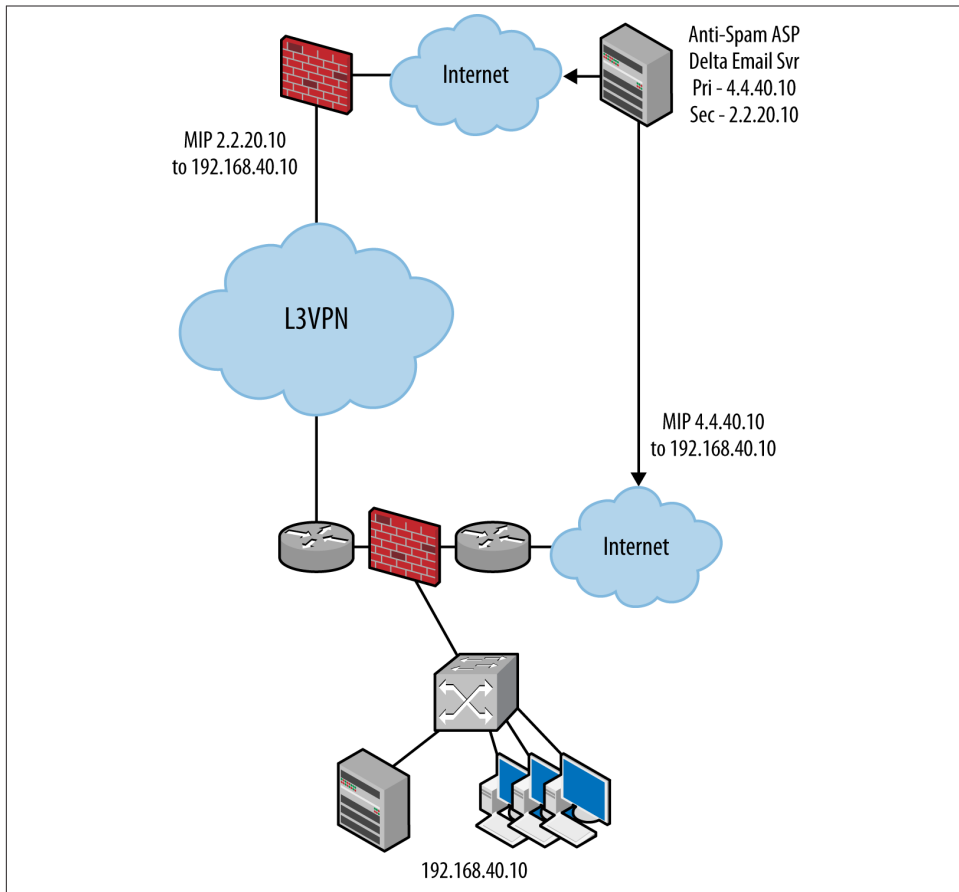


Figure 10-8. Email resolution

In this example, the email server at core site Delta has a private address of 192.168.40.10 and a mapped (static) public address of 4.4.40.10. During a failover, site Delta is backed up by site Beta. At site Beta, the firewall has a static address translation that maps the Delta email address (192.168.40.10) to a local public address (2.2.20.10). The anti-spam server lists both the primary address (4.4.40.10) and the secondary address (2.2.20.10)

for the Delta email server. When the anti-spam server tries to reach the Delta email server, it attempts the primary server address first. If that does not respond, the secondary address is attempted. If the server is active, then one of the two addresses responds and the mail is delivered.

This arrangement reduced the delay in receiving incoming email from 30 to 40 minutes to less than a minute. The amount of time to receive an incoming message was not changed during an Internet link failure. This was a better arrangement for the client and a better solution overall.

Once all the issues were resolved and the testing completed for the Internet failover, each of the locations was upgraded with the new configurations.

Firewall Configurations

The configuration structure of the Juniper Networks Netscreen routers is different than that of the Juniper Network routers and SRX firewall products. It is more aligned to the configuration of a Cisco architecture than the other Juniper products. For the purposes of this discussion, the original output of the *get config* command has been modified (rearranged). If you were to copy and paste these configuration snippets into a Netscreen firewall, I think it would work, but I haven't actually tried it.

If this is the first time you're looking at the configuration of a Netscreen device, what will stand out are the exit commands in the configuration. The Netscreen configuration, like the Junos configuration, is hierarchical (it is not shown in that manner here, but from a machine level, it is). The exit commands bring the device back to the global level prior to issuing the next configuration commands. For example, the following configuration assigns an OSPF area and enables the protocol on the device:

```
set vrouter "trust-vr"  
unset auto-route-export  
set protocol ospf  
set enable  
set area 0.0.0.1  
exit  
exit
```

The first hierarchical level is the *vrouter* (virtual router) level. The global device supports multiple virtual routers: the default is called the *trust-vr* (actually, there are two default vrouter in the device, the other is the *untrust-vr*). The *auto-export* command is disabled at the virtual router level. The next level of the configuration is the protocol itself. At this level, the area is defined (0.0.0.1) and the protocol is enabled (on that virtual router). The two exit commands are to return the device to the global level for the next configuration commands.

Now that you have the basic idea, let's get started. The first portion of the configuration deals with the administrative aspects of the device. These elements are set so that the device can be managed. The configuration contains NTP, DNS, SNMP, and user information:

```
set clock ntp
set clock timezone -5
set vrouter trust-vr sharable
set vrouter "untrust-vr"
exit
set ntp server "pool.ntp.org"
set ntp server backup1 "nist1-ny.ustiming.org"
set ntp interval 120
set ntp max-adjustment 120
```

The clock settings allow the use of the internal clock or an NTP server. We'll configure the NTP servers and enter the NTP specifications later; the commands are grouped here to allow you to see all the elements in a single section. The firewall allows the use of names or IP addresses for server addresses:

This next section of the configuration defines the authorized users for administrative purposes. The name of the root user (admin) is set, as well as the root password (the default is an imaginative *netscreen/netscreen*). The root authentication method (local) is also defined, as is the format for the root user (*dos*):

```
set auth-server "Local" id 0
set auth-server "Local" server-name "Local"
set auth-server "radius.kingco.com" id 1
set auth-server "radius.kingco.com" radius secret
  "lJLmcqlscNNeidAxyA=="
set auth default auth server "Local"
set auth radius accounting port 1646
set admin name "King-Admin"
set admin password "nKVUM2rwMUzPcrkG5swIHdCtqkAibn"
set admin user "read-write" password "nGqHKwrBPn7JcmMtcxHoJn"
  privilege "all"
set admin user "read-only" password "nC62LEQL9HLtNIBSwN" privilege
  "read-only"
set admin root access console
set admin auth timeout 10
set admin auth server "Local"
set admin format dos
```

Two nonroot users are defined here as well. These are given functional names rather than usernames, and these functional names are matched to the RADIUS roles that are used for authorization on the RADIUS server. The RADIUS server sends these names to the Netscreen during user authentication. A read-only user and a read/write user are defined (the only two options for nonroot users).

Once the users are defined, the access means are defined; administrators can access the device via the console, via a managed interface, and/or via a modem. SSH or Telnet has to be enabled on the device as well as allowed on an interface. In our case, only SSH is allowed. The modem can be used, and the settings are shown. For each admin access, a timeout is critical to assure that the access ports are not left open to nonauthorized users:

```
set ssh version v2
set ssh enable
set config lock timeout 5
set modem speed 115200
set modem retry 3
set modem interval 10
set modem idle-time 10
set console timeout 5
```

If you want to use server names rather than IP addresses to identify servers, this requires the use of a DNS server. This next section of configuration identifies the DNS servers and the local identifiers (hostname and domain). We used internal servers as the primary and secondary, and an external server as a final backup:

```
set domain kingco.com
set hostname King-Co-Alpha
set dns host dns1 192.168.10.210
set dns host dns2 192.168.20.210
set dns host dns3 4.2.2.2
set dns host schedule 00:00
```

In a later section, the access provisions for the interfaces that allow this management traffic to the processor are identified.

The Simple Network Management Protocol (SNMP) is used for monitoring purposes. The normal configuration items for monitoring (server, community string, and ports) are configured next. The trap configuration for the firewall is simplistic when compared to that of Junos—it is turned on with a version given, and there's no easy way to restrict or expand on what traps are sent:

```
set snmp community "King-c0-A15a" Read-Write Trap-on version v1
set snmp host "KING-CO-Alpha" 192.168.10.205 255.255.255.255 trap v1
set snmp location "2nd_floor_West_closet_Alpha"
set snmp contact "john_doe_101-555-1234"
set snmp name "KING-CO-Alpha"
set snmp port listen 161
set snmp port trap 162
exit
```

The management logs are best stored off the device. For our client, this meant a syslog server. The configuration for the syslog settings is:

```

set syslog config 192.168.10.233 facilities local0 local0
set syslog config ip address log all
set syslog src-interface ethernet0/0
set syslog enable

```

The syslog settings create log entries for traffic and events and forward them to the syslog server with the *local0* facility tag associated with them.

The next couple of sections of the configuration set up the security architecture of the firewall. As with SRX firewalls, the security architecture is made up of zones, interfaces, and addresses. First the zones are created, then the interfaces are assigned to the zones. Once that is done, IP addresses can be added to the interfaces. This sounds simple enough, but changing an interface from one zone to another requires the IP address to be removed from the interface and the interface deleted from the original zone and then added to the new zone. The IP address then can be added to the interface once again. (Sometimes I miss Junos!) In the following configuration snippet, the three zones are created, an interface is assigned to each, and IP addresses are assigned to the interfaces:

```

set zone "Trust" vrouter "trust-vr"
set zone "Untrust" vrouter "trust-vr"
set zone "MPLS" vrouter "trust-vr"

```

Associated with the zones are a number of protection schemes. The options for each zone are:

```

set zone trust ?
asymmetric-vpn      asymmetric vpn
block                intra-zone block
pbr                  Enable zone pbr-policy
reassembly-for-alg   IP/TCP reassembly for ALG on traffic from/to
                     this zone
screen               configure attack screen
tcp-rst              tcp non syn send reset back
vrouter              virtual router

```

Starting at the bottom, each zone is assigned to a virtual router—in our case, all the zones are in the *trust-vr* virtual router. The next option is to send a reset message for all TCP sessions that have reached the firewall in an asymmetrical fashion (e.g., the firewall has not seen the SYN packet). For our client, asymmetrical routing meant that there was a problem in the network, and such traffic should be blocked. The *screen* option allows the use of denial of service checks for traffic on that zone. The *screen* options are:

```

set zone trust screen ?
alarm-without-drop   Don't drop packet, only generate alarm
block-frag           enable ip fragment blocking
component-block       enable component block protection
fin-no-ack           enable Fin bit with no ACK bit in flags protection
icmp-flood           enable icmp flood protection
icmp-fragment         enable icmp fragment protection
icmp-id              enable icmp ping id zero protection
icmp-large            enable too large icmp packet (size > 1024)

```

	protection
ip-bad-option	enable ip with bad option detection
ip-filter-src	filter ip src route option
ip-loose-src-route	enable ip with loose source route option detection
ip-record-route	enable ip with record route option detection
ip-security-opt	enable ip with security option detection
ip-spoofing	enable address spoofing protection
ip-stream-opt	enable ip with stream option detection
ip-strict-src-route	enable ip with strict source route option detection
ip-sweep	enable address sweep protection
ip-timestamp-opt	enable ip with timestamp option detection
land	enable land protection
limit-session	limit sessions
mal-url	block malicious URL
ping-death	enable ping of death protection
port-scan	enable port scan protection
syn-ack-ack-proxy	enable syn-ack-ack proxy protection
syn-fin	enable SYN & FIN bits set attack protection
syn-flood	enable SYN flood protection
syn-frag	enable SYN frag packet detection
tcp-no-flag	enable TCP packet without flag protection
tear-drop	enable teardrop protection
udp-flood	enable udp flood protection
unknown-protocol	enable unknown protocol protection
winnuke	enable winnuke attack protection

The screen functions are self-explanatory, and only a few were deemed necessary for the Internet link; no screen functions were installed on the trust and MPLS zones.

The remaining zone options are also self-explanatory. The options that were assigned to the zones were:

```
set zone "Trust" tcp-rst
set zone "Untrust" tcp-rst
set zone "MPLS" tcp-rst
set zone "Untrust" screen tear-drop
set zone "Untrust" screen syn-flood
set zone "Untrust" screen ping-death
set zone "Untrust" screen ip-filter-src
set zone "Untrust" screen land
```

While not directly related to the zones, there are a couple of device-level configurations that relate to the *tcp-rst* commands shown above. These determine how traffic flows are handled on the device. The three configuration lines are:

```
set flow tcp-mss
set flow no-tcp-seq-check
set flow tcp-syn-check
```

The first allows the device to determine the TCP segment size for traffic passing through the device. This is helpful for TCP traffic that is traversing an IPSec tunnel. The second and third are similar to the *tcp-rst* commands in that they allow or block out-of-sequence TCP segments. In our case TCP sequence checking is turned on to block out-of-sequence TCP flows.

Now that the zones are configured, it is time to add the interfaces and the IP addresses to those interfaces. The following configuration sets these parameters:

```
set interface "ethernet0/0" zone "Trust"
set interface "ethernet0/1" zone "MPLS"
set interface "ethernet0/2" zone "Untrust"
set interface ethernet0/0 ip 10.10.0.1/29
set interface ethernet0/1 ip 192.168.10.1/24
set interface ethernet0/2 ip 1.1.11.2/24
```

At this point, the firewall can communicate with other devices (to pass traffic), but it cannot be managed from any port other than the console. To allow SSH to the device from the management stations, a number of things have to be configured. SSH has been enabled and the version set in the configurations shown above. Now the interfaces that will support SSH access have to be set up. In our case, two interfaces allow SSH access, Ethernet0/0 for local administrators and Ethernet0/1 for remote access over the MPLS network. All external (Internet) access is only through a Juniper Networks SA2000 that supports SSL VPNs (another story).

To allow remote management, two settings are created: a manage setting and a manager setting. The first allows the services to exist on the interface, and the second defines the IP addresses that can perform the management function. For our client, the management addresses were in the Alpha location and were in the address block 192.169.10.192/26. The granularity for the manage setting is:

```
set interface ethernet0/0 manage ?
ident-reset      turn ident reset manageability of interface on/off
mtrace           turn mtrace manageability of interface on/off
ping             turn interface ping on/off
snmp             turn snmp manageability of interface on/off
ssh             turn SSH manageability of interface on/off
ssl             turn SSL manageability of interface on/off
telnet          turn telnet manageability of interface on/off
web             turn web manageability of interface on/off
```

For our client, the following settings were activated:

```
set interface ethernet0/0 ip manage ping
set interface ethernet0/0 ip manage snmp
set interface ethernet0/0 ip manage ssh
set interface ethernet0/1 ip manage ping
set interface ethernet0/1 ip manage snmp
set interface ethernet0/1 ip manage ssh
```

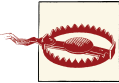
These settings allow ping traffic in and out from any source, SNMP traffic from the management servers, and SSH traffic from the management workstations. The management IP restrictions are defined by:

```
set admin manager-ip 192.168.10.192/26
set admin manager-ip 192.168.20.224/27
set admin manager-ip 192.168.30.224/27
set admin manager-ip 192.168.30.224/27
```

The local management stations are allowed in via Ethernet0/0 while the remote administrators are allowed in via the Ethernet0/1 interface. These configurations provided access for most of the administrators of the company and access to all the local servers.

ScreenOS supports a management address for each interface. This address is not used for routing and will never be used to originate traffic. It is set aside for management functions and is sort of “hidden” for that purpose. For our client, we assigned management addresses for both the Ethernet0/0 and Ethernet0/1 interfaces:

```
set interface ethernet0/0 manage-ip 192.168.10.2
set interface ethernet0/1 manage-ip 10.10.0.2
```



In ScreenOS, three commands are used, and they all have a very similar syntax. Care should be taken to understand where each is used:

- There is a *manage* command that allows services on an interface.
- There is a *manage-ip* command that gives a management IP address to an interface.
- There is a *manager-ip* command that defines the valid addresses that can access the device for management purposes.

These are all used together to manage the device.

In this deployment, local users in the trust zone needed DHCP service. Rather than spin up another server, the Netscreen was put to that use. The following configuration snippet illustrates:

```
set interface ethernet0/0 dhcp server service
set interface ethernet0/0 dhcp server auto
set interface ethernet0/0 dhcp server option gateway 192.168.10.1
set interface ethernet0/0 dhcp server option netmask 255.255.255.0
set interface ethernet0/0 dhcp server ip 192.168.10.3 to 192.168.10.190
```

Here, the local network users were assigned addresses that did not conflict with the management stations or the static servers.

The next portion of the configuration was the network address translation (NAT) configuration. The client used two different NAT methodologies. The first was interface-based NAT, which translated Internet-bound traffic from the rank-and-file King Capital users. The second was a static NAT mapping that was used by certain users and the public-facing servers:

```
set interface ethernet0/0 nat
set interface ethernet0/1 nat
set interface ethernet0/2 route
```

The interface-based NAT is set up on the interfaces involved. The interfaces can be set in one of two modes, NAT or route. Traffic entering the firewall on a NAT-mode interface and exiting the firewall on a route-mode interface is source address NATed. In our configuration, traffic arriving on either the MPLS interface (Ethernet0/1) or the local LAN interface (Ethernet0/0) will be NATed when it is headed for the Internet interface (Ethernet0/2).

The static NAT is called *mapped IP* (MIP) translation. This is a bidirectional NAT for outgoing and incoming traffic. The MIP addresses are actually used as the address book entries in the security policies. These addresses are associated with the outgoing interface where the translation should occur. In our case, internal traffic will use the private address, while Internet traffic will see the public addresses (which are translated at the firewall on the Internet interface). The MIP configuration is:

```
set interface "ethernet0/2" mip 1.1.10.100 host 192.169.10.200 netmask
255.255.255.255 vr "trust-vr"
set interface "ethernet0/2" mip 1.1.10.101 host 192.169.10.201 netmask
255.255.255.255 vr "trust-vr"
set interface "ethernet0/2" mip 1.1.10.102 host 192.169.10.202 netmask
255.255.255.255 vr "trust-vr"
```

The IPSec tunnels from the local stores are the next item to be configured. Each store creates a tunnel to a core location for access to the corporate servers. At the firewall, all of the tunnels are terminated on a single tunnel interface (*tunnel.1*). They are route-based tunnels. The configuration for the tunnel interface and the IPSec components is:

```
set interface "tunnel.1" zone "Trust"
set interface tunnel.1 ip 10.10.101.1/24
```

Setting the tunnel interface assigns it an address and installs it in the trust zone (the use of the trust zone allows all internal connectivity for these users, and the same Internet connectivity as for a local user). Next, the Internet key exchange (IKE) gateway configuration defines the remote device and the methods for creating a session key for the tunnel:

```
set ike gateway "Store_Alpha_1" address 1.1.3.1 Main
outgoing-interface "ethernet0/2" preshare
"P+ZHRAbwNUu5bVsyxi9nG/9j0rQ==" proposal "pre-g2-3des-sha"
set ike gateway "Store_Alpha_2" address 1.1.4.1 Main
outgoing-interface "ethernet0/2" preshare
```



```
"w0Z8s5USNZ4+C8N+06FnwrzjKJw==" proposal "pre-g2-3des-sha"
set ike gateway "Store_Alpha_3" address 1.1.5.1 Main
outgoing-interface "ethernet0/2" preshare
"VFoy5IdlNy4X+VWwCkbnOK4LWrQ==" proposal "pre-g2-3des-sha"
```

The VPN settings link the gateway to the tunnel interface and the actual encryption mechanism used:

```
set vpn "core-to-Alpha_1" gateway "Store_Alpha_1" no-replay tunnel
idletime 0 sec-level standard
set vpn "core-to-Alpha_1" id 1 bind interface tunnel.1
set vpn "core-to-Alpha_2" gateway "Store_Alpha_2" no-replay tunnel
idletime 0 sec-level standard
set vpn "core-to-Alpha_2" id 2 bind interface tunnel.1
set vpn "core-to-Alpha_3" gateway "Store_Alpha_3" no-replay tunnel
idletime 0 sec-level standard
set vpn "core-to-Alpha_3" id 3 bind interface tunnel.1
exit
set vpn "core-to-Alpha_1" proxy-id local-ip 192.168.10.0/24 remote-ip
192.168.100.0/24 "ANY"
set vpn "core-to-Alpha_2" proxy-id local-ip 192.168.10.0/24 remote-ip
192.168.101.0/24 "ANY"
set vpn "core-to-Alpha_3" proxy-id local-ip 192.168.10.0/24 remote-ip
192.168.102.0/24 "ANY"
```

The proxy identifier is a means to verify that the tunnel is actually being used by the proper traffic. It identifies the local IP addresses, the remote IP addresses, and the applications that are to be used on the tunnel. This information has to match on both ends of the tunnel.

Associated with the tunnels is a set of static routes that send traffic to the store locations. The routes are:

```
set route 192.168.100.0/24 gateway 10.10.101.2
set route 192.168.101.0/24 gateway 10.10.101.3
set route 192.168.102.0/24 gateway 10.10.101.4
```

With the IPSec tunnels in place, the actual security portion of the firewall is installed. Our client had a very simple firewall plan: company users were given free access to the Internet, while all incoming Internet traffic was limited to return traffic from local users or specific access to the public servers. The policies are:

```
set policy id 1 from "Trust" to "Untrust" "Any" "Any" "ANY" permit
set policy id 1
exit
set policy id 2 from "MPLS" to "Untrust" "Any" "Any" "ANY" permit
set policy id 2
exit
set policy id 3 from "Untrust" to "Trust" "20.100.23.0/24"
"MIP(1.1.10.100)" "HTTP" permit
set policy id 3
exit
```

```

set policy id 4 from "Untrust" to "Trust" "20.100.23.0/24"
  "MIP(1.1.10.101)" "DNS" permit
set policy id 4
exit
set policy id 5 from "Untrust" to "Trust" "20.100.23.0/24"
  "MIP(1.1.10.102)" "SMTP" permit
set policy id 5
exit
set policy id 6 from "Trust" to "MPLS" "Any" "Any" "ANY" permit
set policy id 6
exit
set policy id 7 from "MPLS" to "Trust" "Any" "Any" "ANY" permit
set policy id 7
exit

```

Considering that each site had Internet access, there is no need for untrust to MPLS policies. When we cover the configuration to add the redundancy, these policies will be added for the remote servers (MIPed to the local IP address).

The next portion of the configuration is the routing that allows the traffic to find the proper port on the firewall. The routing is a combination of static routes and OSPF. The OSPF configuration is:

```

set vrouter "trust-vr"
set protocol ospf
set enable
set area 0.0.0.1
exit
exit
set vrouter "trust-vr"
set router-id 0.0.0.1
unset add-default-route
set interface ethernet0/1 protocol ospf area 0.0.0.1
set interface ethernet0/1 protocol ospf enable
set interface ethernet0/0 protocol ospf area 0.0.0.0
set interface ethernet0/0 protocol ospf enable
exit

```

In this configuration snippet, the OSPF routing protocol is enabled, the router ID is set, and then the interfaces that are running OSPF are defined, as are the areas in which the interfaces reside. This device is an area border router (ABR) in OSPF terms, but for the purposes of ScreenOS, it is on area 1 with interfaces in area 0.

The static route to the Internet provides a connection directly to the Internet border router. The default costs and metrics are assigned to this route:

```

set route 0.0.0.0/0 gateway 1.1.11.1

```

The final portion of the configuration that I'll present here is the components that we added to provide the distributed Internet access that was the focus of the engagement. Only a single location is shown in this configuration, but in the final solution, each core location was connected to two other core locations to provide an increased level of survivability. The solution included four principal components:

- A GRE tunnel to bypass the MPLS routers
- The routing pointing to the remote location
- A policy-based routing scheme that made sure traffic was not trashed due to reverse path forwarding policies
- A set of policies and MIPs that allowed return email traffic to hit the correct servers

The first element of the solution is the GRE tunnel. The components of that are:

```
set interface "tunnel.10" zone "Trust"  
set interface tunnel.10 ip 192.168.23.2/24  
set interface tunnel.10 tunnel encap gre  
set interface tunnel.10 tunnel local-if ethernet0/1 dst-ip 10.10.0.2
```

The routing for the local traffic to reach the remote Internet access and the remote traffic to return uses a combination of static (outgoing) and OSPF (incoming) routing.

The following configuration sets the outgoing static route that will send traffic to the tunnel interface to reach the remote core site for Internet access:

```
set route 0.0.0.0/0 gateway 192.168.23.1 metric 100
```

The route carries a higher metric (100) than the default static route. This route will reside in the routing table but will not be used until the local default route is removed from the table.

When the local Internet interface was determined to be down, the secondary default gateway was used. To guard against a link failure that would not be seen by the local interface, IP monitoring was added to the local Internet link:

```
set interface ethernet0/2 monitor track-ip ip  
set interface ethernet0/2 monitor track-ip threshold 3  
set interface ethernet0/2 monitor track-ip weight 2  
set interface ethernet0/2 monitor track-ip ip 4.2.2.2  
set interface ethernet0/2 monitor track-ip ip 75.75.75.75  
set interface ethernet0/2 monitor track-ip ip 1.1.11.1  
set interface ethernet0/2 monitor track-ip ip 134.43.12.101
```

The monitor configuration protects against a nonlocal failure (i.e., not the interface between the firewall and the Internet border router). The monitor pings addresses and looks for responses. A lack of responses causes the routing table to pull all routes associated with the interface that is monitored.

The OSPF protocol allows the traffic to be returned to the remote locations. To avoid the situation where the traffic arrives on the tunnel and returns on the MPLS (and gets dropped by the remote firewall), the cost of the tunnel is extended to be well above that of all the local OSPF routes:

```
set interface tunnel.10 protocol ospf area 0.0.0.0
set interface tunnel.10 protocol ospf cost 1000
```

Also included in this area is the policy based routing (PBR—no not Pabst Blue Ribbon, but adding beer isn't a bad idea when you're trying to solve routing problems):

```
set access-list extended 1 dst-ip 192.168.20.0/24 entry 1
set match-group name Internet
set match-group Internet ext-acl 1 match-entry 1
set action-group name Internet
set action-group Internet next-interface tunnel.10 action-entry 1
set pbr policy name Internet
set pbr policy Internet action-group Internet 1
exit
set interface ethernet0/2 pbr Internet
```

The PBR looks for returning traffic from the Internet and forwards it back to the tunnel. This routing mechanism has precedence over normal destination routing.

The final configuration item is probably the most important, as without it, no traffic could use the tunnel. The MIPs and the policies for the redundant Internet access are presented next:

```
set interface "ethernet0/2" mip 1.1.10.200 host 192.169.20.200
netmask 255.255.255.255 vr "trust-vr"
set interface "ethernet0/2" mip 1.1.10.201 host 192.169.20.201
netmask 255.255.255.255 vr "trust-vr"
set interface "ethernet0/2" mip 1.1.10.202 host 192.169.20.202
netmask 255.255.255.255 vr "trust-vr"
```

A new set of MIPs were created that used a secondary address at the spam server. These addresses route the traffic to the alternate site (Alpha). Once there, the alternate address is translated to the server's actual address, and the traffic is routed over the GRE tunnel. The policies that support those flows are:

```
set policy id 8 from "Untrust" to "MPLS" "20.100.23.0/24"
"MIP(1.1.10.200)" "HTTP" permit
set policy id 8
exit
set policy id 9 from "Untrust" to "MPLS" "20.100.23.0/24"
"MIP(1.1.10.201)" "DNS" permit
set policy id 9
exit
set policy id 10 from "Untrust" to "MPLS" "20.100.23.0/24"
"MIP(1.1.10.202)" "SMTP" permit
set policy id 10
```

To restrict access to all but the spam server, its address is entered as an address book entry, and the policies are restricted to that address as a source address:

```
set address "Untrust" "20.100.23.0/24" 20.100.23.0 255.255.255.0
```

With the configuration complete, the routing table displays the alternate path to the Internet (the second 0.0.0.0 entry) and the return routes for the remote locations (next hop of *tun.10*). Due to the increased cost, these routes will not be used until the alternate routes are active:

get route

IPv4 Dest-Routes for <untrust-vr> (0 entries)

H: Host C: Connected S: Static A: Auto-Exported
I: Imported R: RIP P: Permanent D: Auto-Discovered
iB: IBGP eB: EBGP O: OSPF E1: OSPF external type 1
E2: OSPF external type 2

IPv4 Dest-Routes for <trust-vr> (19 entries)

	ID	IP-Prefix	Interface	Gateway	P	Pref	Mtr	Vsys
*	38	0.0.0.0/0	ethernet0/2	1.1.11.1	S	20	1	Root
	29	0.0.0.0/0	tun.10	192.168.23.1	S	20	100	Root
*	10	10.10.0.1/32	ethernet0/0	0.0.0.0	H	0	0	Root
*	9	10.10.0.0/29	ethernet0/1	0.0.0.0	C	0	0	Root
*	20	1.1.11.2/32	ethernet0/2	0.0.0.0	H	0	0	Root
*	8	192.168.10.1/32	ethernet0/0	0.0.0.0	H	0	0	Root
*	23	10.10.101.1/32	tun.1	0.0.0.0	H	0	0	Root
*	22	10.10.101.0/30	tun.1	0.0.0.0	C	0	0	Root
*	25	192.168.23.2/32	tun.10	0.0.0.0	H	0	0	Root
*	24	192.168.23.0/24	tun.10	0.0.0.0	C	0	0	Root
	26	192.168.23.0/24	ethernet0/1	10.10.0.2	O	60	101	Root
*	16	192.168.20.0/24	ethernet0/1	10.10.0.2	E2	200	0	Root
*	15	192.168.30.0/24	ethernet0/1	10.10.0.2	E2	200	0	Root
	7	192.168.10.0/24	ethernet0/0	0.0.0.0	C	0	0	Root
*	14	192.168.40.0/24	ethernet0/0	10.10.0.2	E2	200	0	Root
*	13	10.40.0.0/24	ethernet0/0	10.10.0.2	E2	200	0	Root
*	12	10.20.0.0/24	ethernet0/0	10.10.0.2	E2	200	0	Root
*	11	10.30.0.0/24	ethernet0/0	10.10.0.2	E2	200	0	Root
*	19	1.1.11.0/24	ethernet0/2	0.0.0.0	C	0	0	Root

Conclusion

The value of the Internet to most companies continues to grow. With this growth come security concerns and reliability issues. It is no longer allowable to have Internet access

be “down” for any length of time. The Juniper Networks Netscreen firewall family of devices provides the capabilities and the security that corporations need. The ScreenOS provides a mature set of advanced features that allow an IT professional to create innovative solutions to real-life problems.

Internet Access Rebuild

While I was writing this book, the company that I have worked with and for the last six years—Proteus Networks—was acquired by another company. That company approaches projects with a different methodology than the one Proteus Networks used. This methodology starts with a discovery phase, followed by testing and finally implementation phases—a much improved methodology from what we were used to. It is much better to have a specific phase of the project dedicated to investigating the client's environment prior to jumping in and getting to the meat of the project.

In this engagement, the discovery phase was an eye-opener. It allowed us to see the internal operations of the client's network prior to a major upgrade.

It is said that we're better off not knowing how some things are made or what goes into them—sausages and hot dogs are commonly listed in this category. This client's network could also easily make the list. When we looked into the network configurations and discovered what was actually happening, we were really surprised that the network operated as well as it did.



While out with a group of warriors, I once heard a story about a client's network that was overburdened and at its maximum. It was determined that the principal router needed to be replaced. The client purchased a new, high-powered router to replace the tired old device. While the new device was being installed, the old device was examined and its configuration was cleaned up. After the cleanup, the old device was no longer overburdened and actually ran easily with the corporate load. When the new device replaced the old one, it was so underutilized that the sales team was truly embarrassed—the upgrade had not really been needed. So be the power of a good discovery.

Sad to say, the same was not the case here. The devices were running well; the configurations and the topology were where the problems existed. The issue was that the client's company was growing too fast for the old topology to keep up—things needed to change.

The client is a hosting service with a single 10 Gbps Internet feed. Due to the growth of the business, it was determined that the home-grown (eBay-procured) core needed to be upgraded. This led to the call for us network warriors to get involved. The tribe for this engagement included the usual suspects: me as the architect for the project, a fellow router engineer, and the CTO from the client organization. It was a small tribe, but we got the job done.

Requirements

A good discovery starts with identifying the customer's requirements and their business drivers. In our case, the customer requirements were:

Reliable Internet access

While the 10 Gbps Internet feed had performed well to this point, having only one access point was a failure waiting to happen. There was an arrangement for backup Internet access that did work, but it was not reliable and needed to be swapped out for a true secondary service provider.

DOS attack protection

The customers of the client occasionally got blocked due to denial of service attacks. There needed to be a means to mitigate these attacks with minimal impact on the other customers.

Scalability

The client's growth pattern was such that 200% growth was expected in the next year.

High-bandwidth user support

The need to provide access speeds greater than 1 Gbps for hosting customers.

Existing Network

Did I mention that the existing network was a home-grown construction? Well, the architecture proved it. All traffic entered the network from a dark fiber link that ran between hosting company and a peering point across the street. The fiber was terminated in a 10 Gbps port on a Layer 2 switch. From this point, the circuits fanned out to the Internet border routers and to the server farms. A loose interpretation of the network is shown in [Figure 11-1](#).

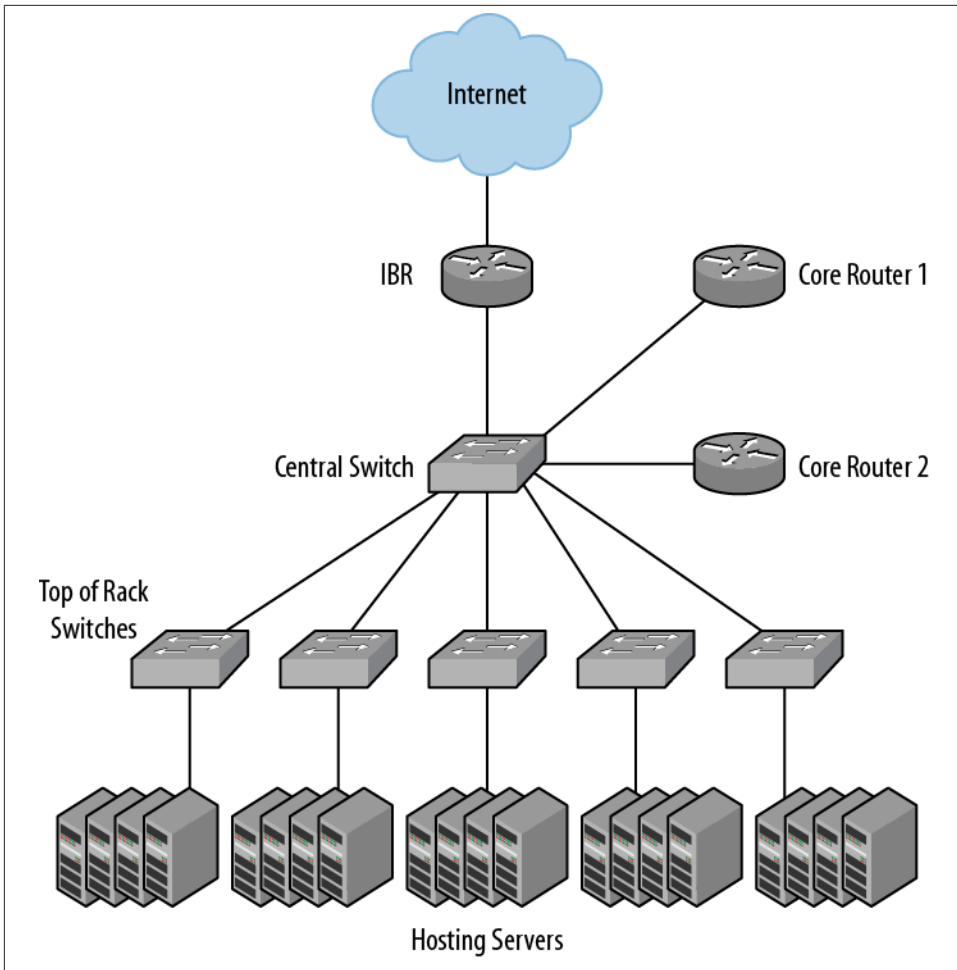


Figure 11-1. Existing hosting network

The traffic from the servers passed through a top-of-rack (ToR) switch to the central switch, and then to the router, back to the central switch, and off to the Internet.

The network was wide open to attacks; it had single points of failure and traffic patterns that would make a cloverleaf look straightforward.

Routing Protocols

The routing in the network was a combination of static routes, OSPF, and BGP. We were told that the original concept was to use OSPF to talk to the ToR switches, but that transitioned to static routes (and the OSPF was never taken out). There was even a BGP peering to one of the ToR switches, which was never explained.

The BGP peering to the Internet provider was straightforward enough, but lacked any type of protection. The client received a default route (0/0) and forwarded its own routes to the ISP. The configuration was:

```
protocol bgp;
group transit {
    description "TRANSIT CONNECTIONS";
    neighbor 1.2.1.7 {
        description "ISP Link";
        local-preference 100;
        import as2-in;
        export as2-out;
        peer-as 2;
    }
}
```



As in all of the examples in this book, the IP addresses here have been sanitized and the names of the actual client companies have been changed.

The BGP peering prompted a few questions. Why was it called “transit,” and who or what were the downstream users? Why was the local preference set to the default value at this point? The answers to these and many more questions were ours to determine, because the original administrator was not available to talk to us.

The configurations for the previously referenced policies were:

```
policy-statement as2-out {
    term block-prefixes {
        from {
            prefix-list dont-re-advertise;
        }
        then reject;
    }
    term advertise-net {
        from {
            prefix-list Hostel-nets;
        }
        then accept;
    }
    term reject {
        then reject;
    }
    then {
        local-preference 5;
    }
}
policy-statement as2-in {
    then {
```

```

        metric 0;
        local-preference 100;
        as-path-prepend 2;
    }
}

```

The use of a prefix list for the export command was correct, so at least that part of the configuration could be used again. But the import policy made us think that the other Juniper router, *cr2*, might have a different local preference and metric (not so—the configurations on both routers were identical).

The export policy had a very common error, the hanging *then* statement after the *reject* term. This is typical of an administrator adding the statement as an afterthought, but forgetting to assign it to a term. We did not find any need for this statement in any other configurations.

The use of two prefix lists was actually a neat trick. The entire inventory of addresses that were owned by the client were listed in the *Hostel-nets* prefix list, while those that were not in use or had to be restricted for whatever reason (e.g., causing a DOS attack) were included in the *dont-re-advertise* list. The one would stay relatively static, while the other could be dynamic. The tribe decided to store that trinket away for future reference.

The client owned a number of /22 and /21 address ranges. The *Hostel-nets* prefix list was a little confusing because it contained the /21 range and the longer variations of the prefixes as well. I have seen this approach taken by other clients when multiple ISPs are used and preferences are given to portions of the address range for one ISP versus the other. Since there was a single ISP here, though, this breakdown did not make sense (so many questions and so few answers!). An example of the confusion in the prefix list was:

```

policy-options {
    prefix-list Hostel-nets {
        3.9.90.0/23;
        3.9.90.0/24;
        142.2.152.0/22;
        142.2.152.0/23;
        142.2.152.0/24;
    }
}

```

In the above snippet, the 3.9.90.0/23 and 3.9.90.0/24 prefixes are being advertised to the same ISP. There is no preference given to one over the other. The same was true with the 142.2.152.0 prefixes: here, the /22, /23, and /24 of the same subnet are advertised. What about the other /24s of the range? Totally weird. This was an area where the tribe could clean things up and offer options for going forward.

The final section of the routing configuration that needed attention was the area of static routes. A few routes were there to create a summary route that could be used in the BGP advertisements (although it would have been better to use aggregate routes), and several were there for access to the customer applications. A few were totally confusing, and a few were just wrong. The routing options were:

```
routing-options {  
  static {  
    route 3.9.89.0/24 discard;  
    route 3.9.90.0/23 discard;  
    route 142.2.152.0/22 discard;  
    route 142.6.220.0/22 discard;  
    route 142.3.232.0/21 discard;  
    route 142.5.16.0/21 discard;  
    route 7.219.96.0/20 discard;  
    route 143.22.64.0/20 discard;
```

This was a good list of the prefixes that are owned by the client. They represented a static entry in the routing table. As such, they would always be advertised to the Internet as part of BGP advertisements. An alternative would be to make this a list of aggregate routes that are advertised only when the prefixes are actually used in the network. When a prefix is not active, it will not be advertised and will not be subject to scans.

The next set of routes was less clear—these are owned prefixes, and a few that are not owned, but recognized. The next-hop addresses were outside of this network:

```
route 143.22.66.0/24 next-hop 143.22.96.4;  
route 16.238.88.0/23 next-hop 143.22.96.12;  
route 16.238.126.0/23 next-hop 143.22.96.16;  
route 7.4.211.0/24 next-hop 142.76.221.62;  
route 7.4.212.0/24 next-hop 142.76.221.62;
```

This traffic was the elusive transit traffic that was identified earlier. The local tribe member had no idea why it was present, so we just added this to the list of questions that would be asked prior to updating (read, deleting) them.

The last set of static routes provided access to the top-of-rack switches for customer applications. These static routes were augmented by the OSPF routes to provide full coverage between the Internet and the hosting servers:

```
route 142.76.22.104/30 next-hop 142.72.152.98;  
route 142.76.22.108/30 next-hop 142.72.152.98;  
route 143.22.110.0/29 next-hop 142.72.152.98;  
route 142.76.22.16/28 next-hop 142.72.152.98;  
route 142.76.22.128/28 next-hop 142.72.152.201;  
route 142.76.22.160/27 next-hop 142.72.152.201;  
route 7.219.109.0/27 next-hop 142.72.152.201;  
  }  
}
```

When a denial of service attack was launched, a static route was added to the routers to blackhole the traffic to the attacked address. Often when an attack happened, access to the routers was very slow because the access path was the same as that taken by the attack traffic. The links between the central switch and the routers were saturated, and they carried both the management traffic and the customers' traffic. The tribe had to build an out-of-band management network to resolve this issue.

Solution Options

In some engagements, the optimal solutions are obvious, and in others, the solutions are restricted by what hardware the client has on hand. In this engagement, the solution was ours to design and install, but the client wanted to review different options, weigh the pros and cons of each, and make an educated decision.

We always aim to support the client, so our little tribe created three variations on a theme that all met the client's requirements, with slightly different hardware. The options can be differentiated by the number of layers in the design: we proposed a three-layer, a two-layer, and a one-layer design.

Three-Layer Design

The three-layer design used the most hardware, but adhered to the “not all in one basket” survivability rules. The design used an MX as an IBR, an SRX as a screening device, and an EX4200 virtual chassis as an access switch. The design is shown in [Figure 11-2](#).

All traffic entered the system through the MX; only traffic for customers that paid for DOS protection was shunted to the SRX1400. Both the MX and the SRX fed traffic to the EX2400s that were in an aggregation position for all the top-of-rack switches.

The connectivity for all the links was provided by 10 Gbps interfaces. The links between the EX4200s and the SRX were redundant Ethernet (Reth) interfaces.

The design was graded against the client's requirements:

Reliable Internet access

This requirement was not met with this design. The single point of failure of the MX40, regardless of the number of ISP links, made this requirement hard to meet.

DOS attack protection

The design allowed DOS protection via two means. The first is routing traffic for customers that pay for DOS protection through the SRX. The *screen* feature of the SRX blocks DOS attacks as they are happening. The second means of protection is the use of BGP blackhole communities by the ISP. When a DOS attack is in progress, the logs can be analyzed and the MX can be updated to include the destination addresses in the blackhole list.

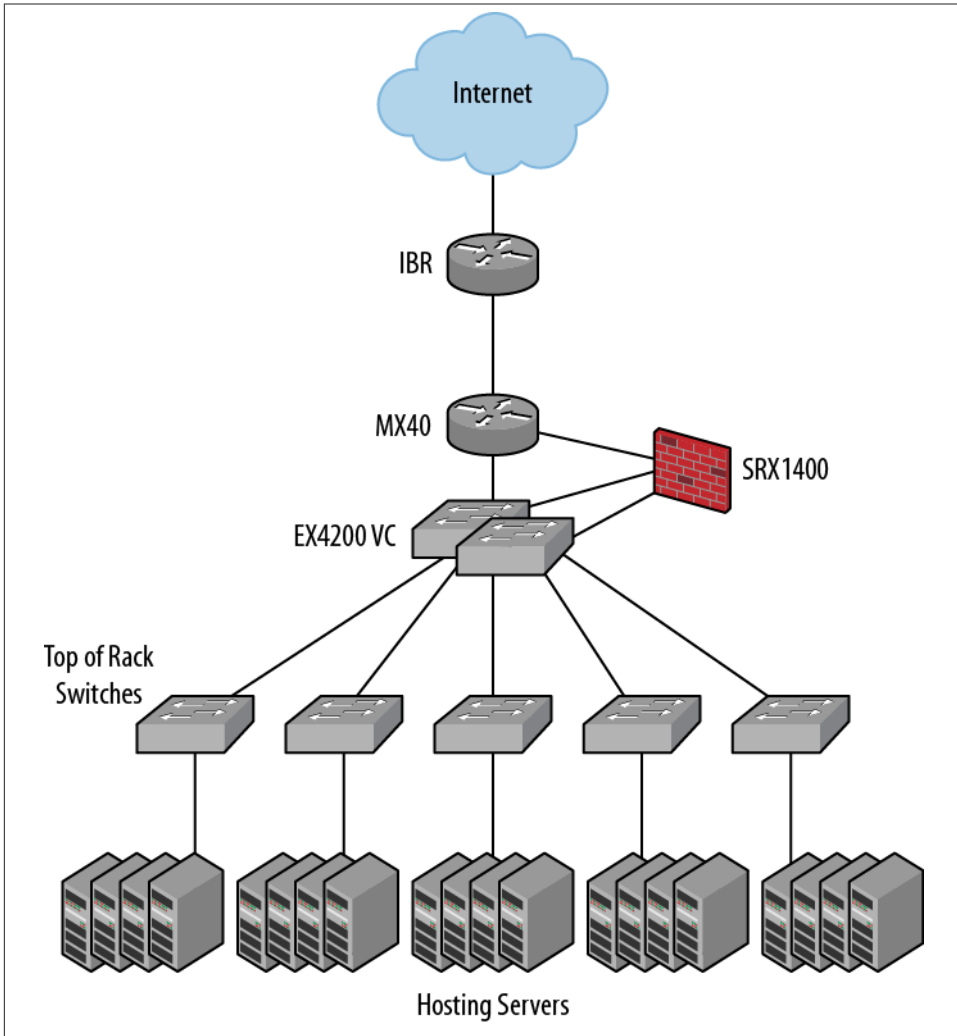


Figure 11-2. Three-tier design

Scalability

This solution had one of the best scalability capabilities. The MX40 can be upgraded to an MX80, adding bandwidth and interfaces as needed. The SRX1400 can handle 10 Gbps of firewalled traffic. Finally, the EX4200 virtual chassis can be increased to handle up to 10 switches (over 450 top-of-rack interfaces).

High-bandwidth user support

This solution also allowed the mixture of MX4500s into the aggregation switch virtual chassis. The EX4200 supports a limited number of 10 Gbps interfaces; with the addition of the EX4500, 40 10 Gbps interfaces are supported per chassis.

All in all, the design met most of the requirements set out by the client. However, its downsides made the client wary of going with this design. The perceived and actual downsides of the design were:

Multiple devices to configure, manage, and maintain

While the use of Junos throughout was an advantage, this was a small shop: the fewer the devices, the better. The three-tier design added two more levels of complexity, which the CTO was trying to eliminate rather than add on to the design.

Single point of failure

The single point of failure was the main killer for this design. The MX40 does not support redundant routing engines, and MX virtual clusters were not yet proven enough for this customer.

Due to these two critical failures, the design was shelved and the tribe went back to the drawing board for design number two.

Two-Layer Design

While the three-layer design took a divide-and-conquer approach, with the two-layer design we looked at the requirements and aimed to more closely align with them while keeping the goal of economy in sight. The design had two components: the aggregation layer was the same as in the previous design, but the security and the border router were consolidated into a single device, an SRX1400 cluster. This approach looked like that depicted in [Figure 11-3](#).

Not shown in the diagram are the interconnection links between the SRX1400s: these two devices are clustered to form a single firewall with twice the capacity and throughput. Also not shown are the connections to the IBR; a pair of links connected each side of the SRX1400 to the Internet. This added survivability to the design and allowed the use of an alternate ISP. The use of redundant Ethernet links between the SRX1400s and the EX4200 virtual chassis (VC) avoided the need for the virtual router redundancy protocol (VRRP). To achieve load balancing between the servers, the SRX1400s operated in an active/active operational mode, with each side terminating different VLANs assigned to the ToR switches.

This design more closely matched the stated requirements for the client:

Reliable Internet access

The design allowed the failure of any Internet-related element: links, routing engines, interfaces, or chassis. Each element was redundant and could handle the full traffic load.

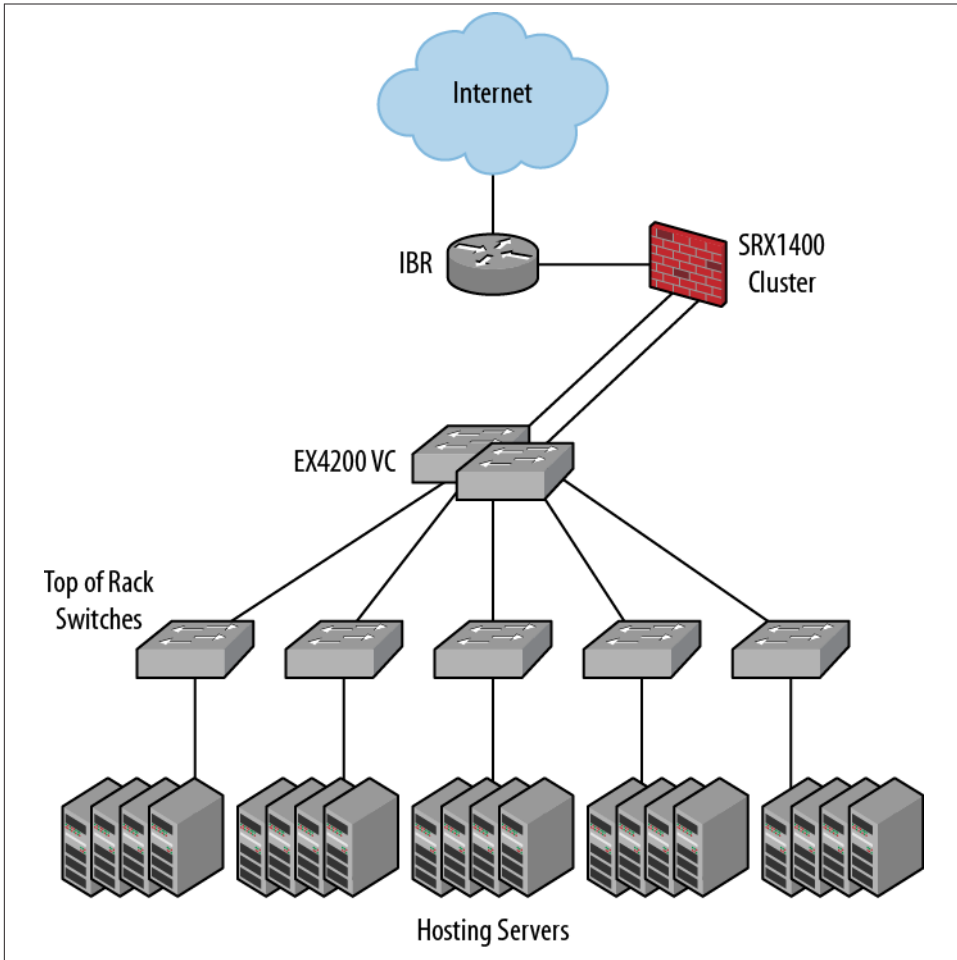


Figure 11-3. Two-tier design

DOS attack protection

As all traffic was fronted by the SRX1400, all clients received the same DOS protection. Screen functions could be tuned to match the traffic profiles for normal traffic and for DOS attacks. The logging capability of the SRX1400 allows the same type of analysis performed by J-flow without the need for a different license. The attacking addresses as well as the attacked addresses can be determined quickly and actions taken to thwart the attack.

Scalability

The design had the same scalability as the three-tier design: the SRX1400s could handle 10 Gbps for each chassis, and the EX4200 VC offers expanded capabilities as well. The existing loads could be quadrupled prior to the design being overloaded.

High-bandwidth user support

Like the previous design, this solution allowed the mixture of MX4500s into the aggregation switch virtual chassis. The EX4200 supports a limited number of 10 Gbps interfaces, and with the addition of the EX4500, 40 10 Gbps interfaces are supported per chassis.

While closer to a winner than the three-tier design, the client was still not happy with a few elements of this design.

The number of devices remained a major issue. We took an MX out but added an SRX, so the point was still being made that more is not better. The added reliability was a winner, but the complexity of the clustering and the virtual chassis was too much.

Casey was at bat, two outs, two strikes in the ninth inning. He needed a hit to win this game. Our third design had to meet the design requirements, and meet the bigger manageability requirements of the CTO.



Dealing with the issues that keep the client awake at night—no matter how minor they may seem—is key to the success of any design. If all the technical requirements are met, yet the client cannot sleep soundly, the design will inevitably fail.

As warriors, we are trained to see these weaknesses and take action. Like other warriors, our actions allow folks to sleep at night. It's just that we have a different job to do, and it is not at all dangerous.

In this client's case the overriding requirement was not at all technical, but rather one of manageability. He wanted a solution that would require him to learn one box.

One-Tier Design

The data center design guidelines presented by Juniper Networks are to take a traditional three-tier design (core-aggregation-access) and compress it to a two-tier design, and ultimately to a one-tier design. The MX universal edge devices and the SRX service gateways allow this compression of functions to a single tier. This was the approach that the tribe took for the third and final design we presented to the client. We had a feeling that we were close, but time was running out for us to find a winning combination. Like Casey, we did not have any strikes left, let alone outs.

For the one-tier design, we needed to upgrade the devices from mid-range devices that would handle the traffic loads to high-end devices that would handle the scalability and the interface load of the aggregation layer. The two devices that are ideal for this were the MX480 and the SRX5600. Both offered the port densities needed (multiple 40-port 1 Gbps cards) and the redundancy (multiple 10 Gbps cards) for the Internet access. The SRX5600 offered the DOS protection, while the MX480 offered the redundant routing engines.

The deciding factor was the logging capabilities of the SRX5600. While the MX480 can perform J-flow logging, a slot had to be dedicated to the multiservices dense PIC concentrator (MS-DPC) that is required for the flows. The SRX5600 offers logging of all traffic to an external server for all policies on the device. We traded off the reliability of the routing engine for the capability of stopping DOS attacks from disrupting customer's traffic (back to what keeps the client up at night).

A diagram is really not necessary to show the one-tier design, but I would be remiss to skip it at this point. [Figure 11-4](#) shows the location of the SRX.

The other trade-off between the SRX5600 and the MX480 was one of the power of the routing engine for Internet-based routing. As we saw in the discovery phase, the client did not expect to receive the full routing table from the ISPs. We therefore decided that either device could satisfy the routing engine requirements.

When we talked to the client about the redundancy of the routing engine, we were told that the existing M20s had not had a routing engine failure *ever*, and that these boxes were already ancient when they were purchased on eBay.

The CTO also stated that if the growth pattern of the company continued as it was, a second SRX5600 could be added to the design to resolve the reliability issue.

So, the new design was compared to the requirements again:

Reliable Internet access

The one-tier design allowed the failure of most of the Internet-related elements: links, ports, and cards. Each redundant element could handle the full traffic load.

DOS attack protection

The SRX5600 offers the industry-leading DOS screen protections and reporting capabilities.

Scalability

The SRX5600 has room for five I/O slots that can handle up to 40 Gigabit Ethernet interfaces (and a total capacity of 160 Gbps), while also supporting four 10 Gbps interfaces for Internet access.

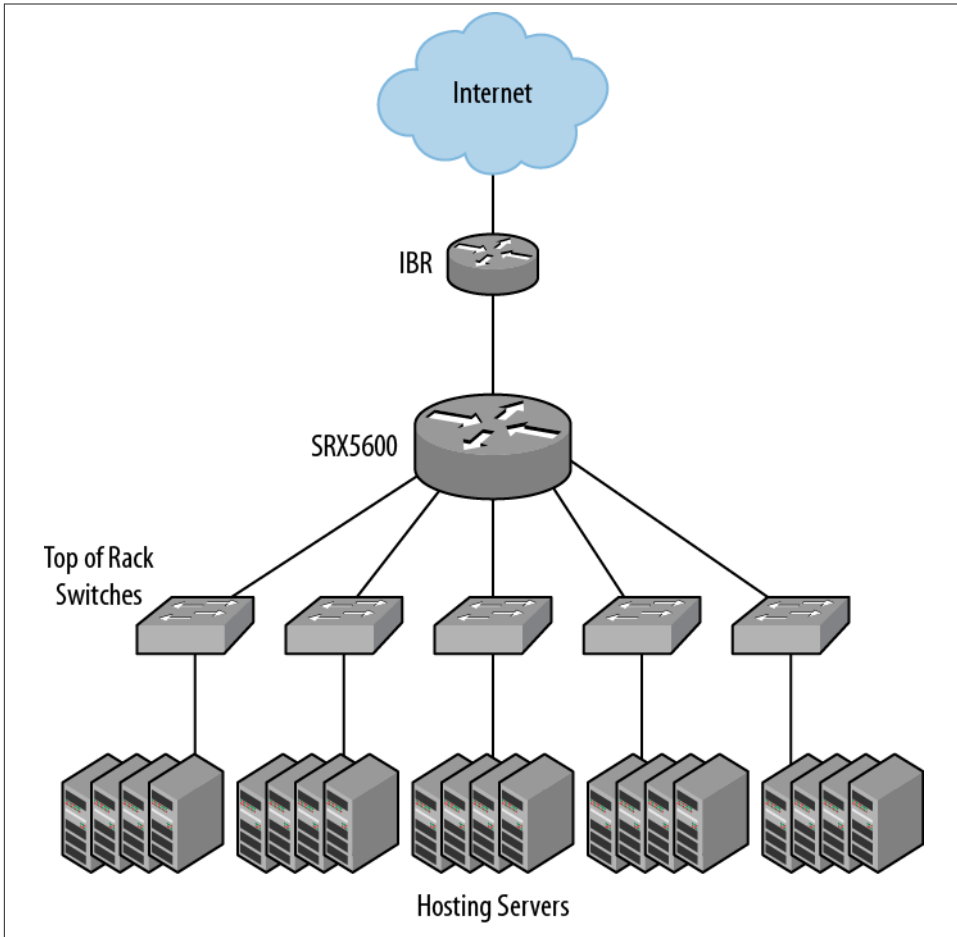


Figure 11-4. One-tier design

High-bandwidth user support

The SRX5600 supports a flex I/O card that can handle two modules; each module can handle 4 10 Gbps ports or 16 Gigabit Ethernet ports. As 10 Gbps customers are added, additional cards can be added.

The only problem with this design was the redundancy for the Internet access. The use of different cards and ports offered a level of reliability, while the added features of the SRX made the design, while not a home run, a definite hit.

The proposed SRX5600 configuration is shown in [Figure 11-5](#) and [Figure 11-6](#).

The card complement for the initial design was two 40-port 1 Gbps cards for the ToR switches and two 10 Gbps cards for the Internet access. Each ToR switch was connected with two uplinks to the identical port on each of the 40-port cards. The links from the ISP were initially connected to one of the 10 Gbps cards. Two SPCs (slot 2 and slot 3 in [Figure 11-5](#)) were added to the mix to handle the traffic and provide redundancy.

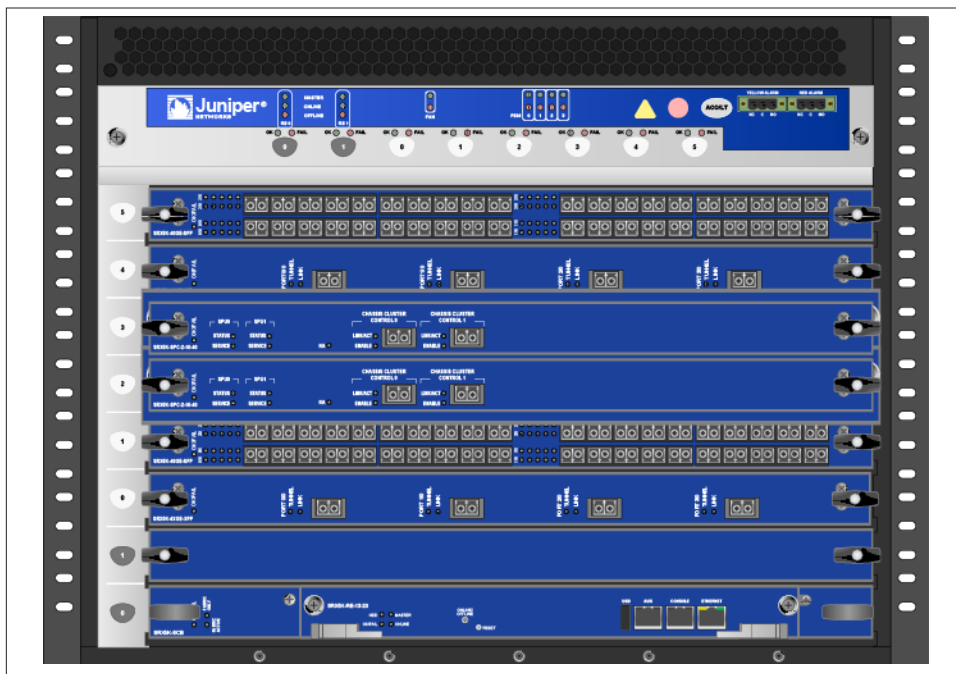


Figure 11-5. Initial SRX5600 configuration

The SRX was positioned between the ISP's IBR and the top-of-rack switches. Each ToR switch was connected to two different I/O cards in the chassis. The initial ISP link was connected to one of the cards, and the same port on the other 10 Gbps card was reserved for the future 10 Gbps ISP link (this link was actually connected to the SRX prior to the completion of the engagement). The cabling is shown in [Figure 11-6](#). While not discussed previously, the management network of the client was connected to the management ports of the top-of-rack switches as well as the management port of the SRX. This out-of-band connection allowed connections to the devices out of the path of the customer traffic (always a good idea if at all possible).

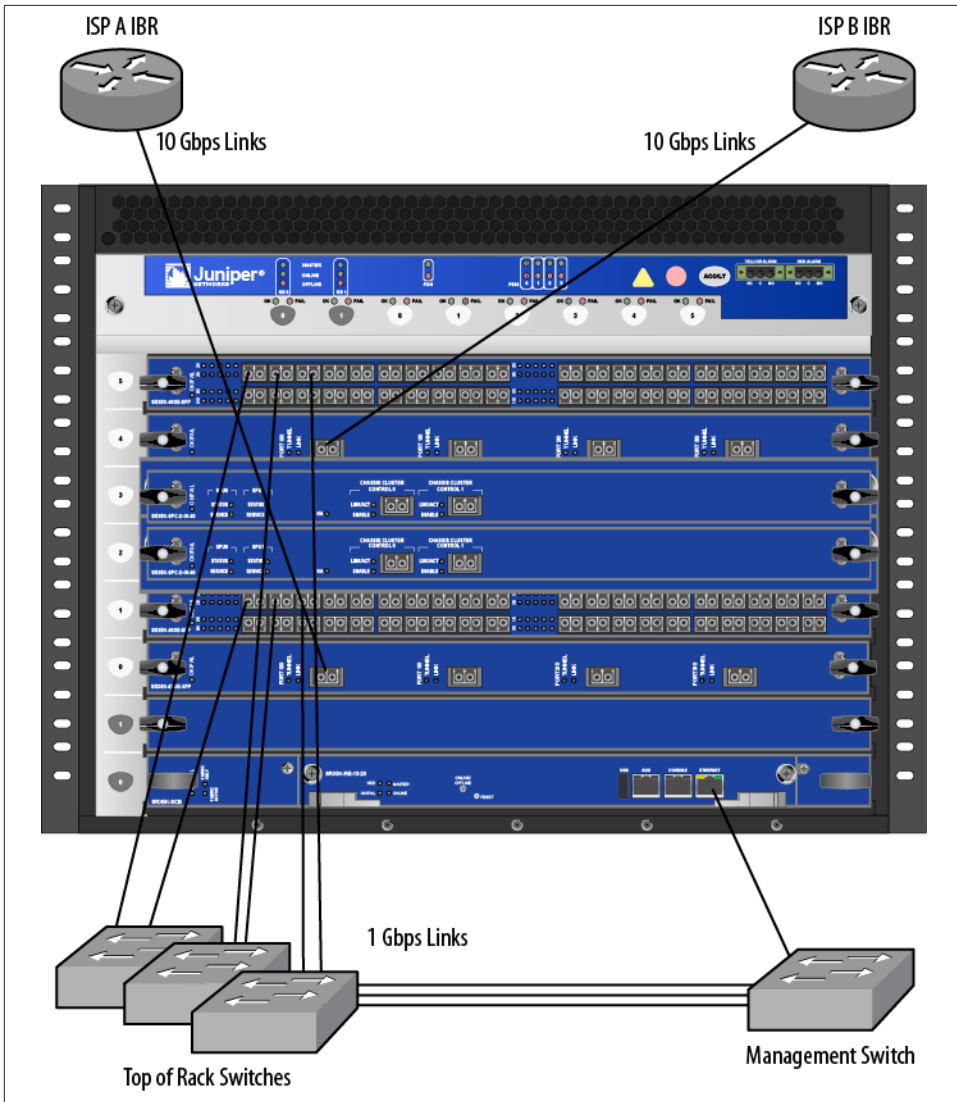


Figure 11-6. Connectivity design

While this design did not cover *all* aspects of *all* the requirements, the areas that were not covered were of the lowest importance to the client. The art of negotiation is key in any engagement and in any proposed solutions: recognizing what will keep the client happy is crucial.

This might seem strange because there is only one SRX (not clustered), but the dual links to the ToR switches were configured as redundant Ethernet links. The operation of the Reth is the same as for a clustered SRX, only providing redundancy between cards, not chassis.

Configurations

The setup of the SRX for this engagement was broken into multiple sessions. Initially the management services were set up, followed by the local connections to the top-of-rack switches, followed by the ISP configurations. The security aspects of the SRX were initially configured as wide open, providing DOS screening only. Server-to-server connectivity between the ToR switches was provided on a case-by-case basis.

Deployment Scenario

The deployment of the SRX in the network could have been a traumatic experience for the client. Initially the CTO foresaw a weekend project that would have his customers yelling and screaming for connectivity to their applications. However, we planned the deployment in a series of steps that allowed us to minimize the impact on the network. The steps used during this deployment were a logical progression of staging and testing:

1. The initial management staging and testing step allowed us to rack and power the SRX in its production environment, establish connectivity between the SRX and the management servers, and test the management functions.
2. The next step was to create a ToR switch template for adding customers to the SRX. The initial configuration was for a “test” switch that was connected to the SRX. For this step, the SRX was given access to the production network for testing purposes only.
3. During the installation, the new ISP link became available. This allowed us to use this link as a test for BGP and the routing system. We were also able to verify the denial of service (DOS) settings at this time. This configuration step allowed us to perform traffic tests to Internet-based servers without disrupting customer traffic.
4. Next was the production configuration step. This stage allowed us to add the full configuration for the top-of-rack switches and the configuration for the actual ISP links.
5. Finally, the SRX was installed to replace the existing equipment (the cut-over). The production configuration was tweaked to resolve some anomalies, and we were done.

In the following sections, we’ll explore more fully what was done in each of these steps.

Management Staging and Testing

The management functions of the design were as simple as the overall design: a Remote Authentication Dial In User Service (RADIUS) server was used to authenticate the users of the system, an SNMP server polled for traffic information, a syslog server captured events, and the usual DNS (one internal and one external) and NTP servers provided support for name resolution and time synchronization. The allowed system services were restricted to SSH only.

The source addresses for each of the servers was set to the loopback interface. This allowed access to the servers from multiple interfaces while still associating the SRX with the servers. The configuration of the system services was:

```
system {
    host-name SRX5600-Core;
    domain-name hosting.com;
    time-zone America/New_York;
    root-authentication {
        encrypted-password "$1$rBQ/asdgjui35bv/";
    }
    authentication-order {
        radius;
        password
    }
    name-server {
        142.76.155.26;
        4.2.2.2;
    }
    radius-server {
        142.76.155.20 {
            port 49;
            secret "$9$OHGM1IKW88oGji5Tp0BErv";
            source-address 142.76.22.200;
        }
        142.76.155.43 {
            port 49;
            secret "$9$aWZGj.PTQn95Q1hLXgoJDqf";
            source-address 142.76.22.200;
        }
    }
    login {
        user ADMIN {
            uid 2017;
            class super-user;
            authentication {
                encrypted-password "$1$pAPUtngdf.";
            }
        }
        user READ-ONLY {
            uid 2019;
            class read-only;
        }
    }
}
```



```

        authentication {
            encrypted-password "$1$ly/MsMFNmoy0";
        }
    }
}
services {
    ssh {
        root-login deny;
        protocol-version v2;
        connection-limit 20;
    }
}
syslog {
    user * {
        any alert;
    }
    host 142.76.155.26 {
        any any;
    }
    file messages {
        any any;
    }
}
ntp {
    boot-server tick.nac.net;
    server ntp.amnic.net;
}
archival {
    configuration {
        transfer-on-commit;
        archive-sites {
            "ftp://admin:admin123@142.76.155.26";
        }
    }
}
}
snmp {
    name SRX5600-CORE;
    location "RK1/RW1/RU0-15";
    contact "help@hosting.com";
    community wrdfsdftw4r3 {
        authorization read-only;
    }
}
}

```

A few notes are in order to explain the management elements of the configuration. The use of a RADIUS authentication service makes administrating access to the device easier. Only a single device is actually touched when admins are added or removed (e.g., if they leave the company). Each device has a set of generic users that are roles rather than

users. The *ADMIN* role and the *READ-ONLY* role are added to the RADIUS profiles of the actual administrative names. When an admin logs in, the RADIUS server queries the SRX for the role attributes and assigns these attributes to the user. In this client's system, this arrangement allows a standard configuration for all devices.

The authentication order of RADIUS, then password assures that if the RADIUS server fails, the devices can still process local passwords for the users. We debated including a local-only user with a common password for each device, but it was decided that the root password was sufficient. We also blocked root access from SSH—if a major failure occurred, console access was going to be necessary.

Another piece of the administrative configuration was the archive-on-commit setting. This allowed off-SRX storage of the configurations each time a change was performed. The setting increased the comfort level of the CTO, reassuring him that if things went really bad, a backup of the most recent configuration would be available to rebuild the SRX.

The SRX is a closed device; from a security standpoint, these management functions could not be tested without some interface and security configuration. To allow testing, a base interface (loopback interface, *fxp0* port, and one ToR switch) was configured as well as a test-only security configuration. This arrangement allowed the SRX to be staged and tested for management functions prior to accepting customer traffic.



The installation of the SRX for customer traffic was to be performed during maintenance windows. To limit the amount of time required for those windows, as much non-customer affecting configuration as possible was performed ahead of time.

This allowed us to focus on the customer connections during the maintenance windows, decreasing the chances of interruptions and loss of customer connectivity.

The initial interface and security configurations were:

```
interfaces {
  ge-1/0/0 {
    unit 0 {
      description "Management ToR Switch";
      family inet {
        address 142.76.155.130/24;
      }
    }
  }
  fxp0 {
    unit 0 {
      description "OOB Management Switch";
      family inet {
```

```

        address 142.76.156.1/24;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 142.76.22.200/32;
        }
    }
}
}
security {
    zones {
        security-zone trust {
            host-inbound-traffic {
                system-services {
                    ping;
                    snmp;
                    traceroute;
                    ssh;
                }
            }
            interfaces {
                ge-1/0/0.0;
            }
        }
    }
}
policies {
    from-zone trust to-zone trust {
        policy permit_all {
            match {
                source-address any;
                destination-address any;
                application any;
            }
            then {
                permit;
                count;
            }
        }
    }
    default-policy {
        deny-all;
    }
}
}

```

Although most of the initial security configuration would eventually be overwritten, its purpose for now was to allow testing. Once the management functions were verified, the next stage of the deployment could begin. The verification tests performed at this point were:

- Console access for administrative users (read/write and read-only) and root
- SSH access for administrative users (with and without RADIUS access), to the *fxp0* port and the *ge-* port
- Polling from the SNMP platform
- DNS name resolving
- NTP synchronization
- Logging to the syslog server
- Archiving of the configuration
- *ping* and *traceroute* operation

The only glitch was the need to add static routes to point traffic to the correct interfaces. Sometimes, it's the little things that cause angst.

Top-of-Rack Switch Testing

Once the management functions of the installation were verified, the operations required for attaching customers to the SRX had to be ironed out. This was done using a test switch and server. The address range for these tests was currently not in use in the production network, so there was very little chance of interrupting the service of actual customers. Two different scenarios were created for this test. The first was for a customer that was hosting an application on a single server, and the other for a customer that had applications on multiple servers and needed communications between these servers as well as out to the Internet.

The connections from the ToR switches to the SRX were redundant Ethernet (Reth) connections that required a single IP address for each pair of links to the switch. This was a departure from the existing design, which used the virtual router redundancy protocol (VRRP) to connect the switches to the two core routers. For each of the /29 address blocks that were assigned for the VRRP addressing, only a /30 was necessary for the new arrangement. The interconnecting links were addressed from a common set of /24 addresses, and we used an unused block for the testing.

At each ToR switch, either a /24 or a /23 was used to serve customers on that switch. The routing at the switch level was via direct routes or via a default route pointing to the SRX. At the SRX, static routes were used to access the correct switch. The logical diagram for the ToR tests was as shown in [Figure 11-7](#).

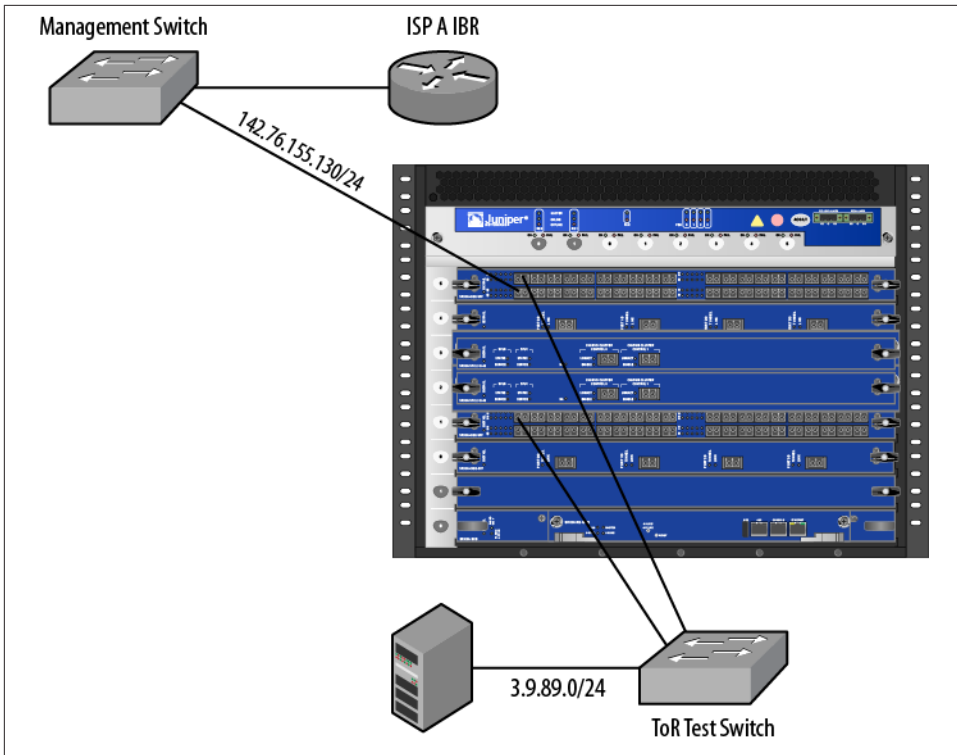


Figure 11-7. ToR test configuration

Because of the rapid growth of the company, new switches were being added all the time. To reduce the configuration hassles, a group template was developed that contained all the elements for deploying a single switch. The group configuration was:

```
groups {
  ToR-Switch-TEST {
    interfaces {
      ge-1/2/0 {
        gether-options {
          redundant-parent reth1;
        }
      }
      ge-5/2/0 {
        gether-options {
          redundant-parent reth1;
        }
      }
    }
    reth0 {
      vlan-tagging;
      redundant-ether-options {
        redundancy-group 1;
      }
    }
  }
}
```

```

    }
    unit 0 {
        vlan-id 201;
        family inet {
            address 143.22.96.1/29;
        }
    }
}
routing-options {
    static {
        route {
            3.9.89.0/24 {
                next-hop 143.22.96.2
            }
        }
    }
}
security {
    zones {
        security-zone Switch_TEST {
            host-inbound-traffic {
                system-services {
                    ping;
                    traceroute;
                }
            }
            interfaces {
                reth0.0;
            }
        }
    }
    address-book {
        address net-3.9.89.0-24 3.9.89.0/24;
    }
    policies {
        from-zone Switch_TEST to-zone Internet {
            policy Internet-Access {
                match {
                    source-address net-3.9.89.0-24;
                    destination-address any;
                    application any;
                }
                then {
                    permit;
                    count;
                    log {
                        session-init;
                    }
                }
            }
        }
    }
    from-zone Internet to-zone Switch_TEST {

```



```

        ping;
        snmp;
        traceroute;
        ssh
    }
}
interfaces {
    ge-1/0/0.0;
}
}
policies {
    from-zone Internet to-zone Internet {
        policy permit_all {
            match {
                source-address any;
                destination-address any;
                application any;
            }
            then {
                permit;
                count;
            }
        }
    }
}
}
routing-options {
    static {
        route {
            0/0 {
                next-hop 142.76.155.131;
            }
        }
    }
}
}

```

The second test case was to allow communications between two switches. This test case was accomplished with the addition of another ToR switch group and the addition of policies to both the groups. The additional policies allowed traffic to traverse between the server prefixes on each switch. The configuration for the policies is shown here without the other ToR switch group. For inter-switch traffic, each group configuration was modified with the originating traffic policy, so from the example below, the *from-zone Switch_TEST* policy was found in the *ToR-Switch-TEST* group configuration and the *from-zone Switch_TEST2* policy was found in the *ToR-Switch-TEST2* group configuration:

```

security {
    policies {
        from-zone Switch_TEST to-zone Switch_TEST2 {
            policy SW-SW-Access {

```



```

        match {
            source-address net-3.9.89.0-24;
            destination-address net-3.9.90.0-24;
            application any;
        }
        then {
            permit;
            count;
            log {
                session-init;
            }
        }
    }
}
from-zone Switch_Test2 to-zone Switch_TEST {
    policy Interent-Access {
        match {
            source-address net-3.9.90.0-24;
            destination-address net-3.9.89.0-24;
            application any;
        }
        then {
            permit;
            count;
            log {
                session-init;
            }
        }
    }
}
}
}
}

```

Once these configurations were established and the second test switch was in place, inter-switch communications were tested successfully. The tests performed for this stage of the deployment included:

- Server access to the Internet
- Management access to the server
- Management access to the switch via the management network
- Server access with a link failure
- Server speed tests to the management servers (FTP downloads)
- Server-to-server testing (connectivity and speed)

What we found in the testing was that inter-server communications for servers that resided on the same switch were not controlled by the SRX and thus were allowed by default. This situation was not optimal, but the investigation of a solution was postponed until after the initial deployment.

ISP Link Testing

Once the tribe was confident that the customer side of the SRX was operational and that we could add switches in a simple and consistent manner, the next battle was the testing of the ISP configurations. As luck would have it, the new ISP service was installed as we were testing the SRX. This made testing to an actual BGP connection (rather than a simulated virtual router on the SRX, which was the original plan) possible and allowed us to verify the new ISP link in the process. As a starting point for this test, we used the old BGP policy and static routing configuration. Most of the configuration elements used in this test were going to be used in the actual deployment, so we wanted to make the configuration as structured as possible. The first elements to be moved over were the static routes that formed the route summaries in the old configuration. The use of aggregate routes rather than static routes minimized the route summaries that were advertised to only those that were in use. We also decided to advertise only a single shorter prefix rather than the mix of short and long. The aggregate route list that remained was:

```
routing-options {
  aggregate {
    route 3.9.90.0/23;
    route 142.2.152.0/22;
    route 142.6.220.0/22;
    route 142.3.232.0/21;
    route 142.5.16.0/21;
    route 7.219.96.0/20;
    route 143.22.64.0/20;
  }
}
```

The full set of routes were added to the aggregate routes, and we determined that we would restrict the advertised routes during the tests by using a test BGP export policy. The existing design had the static routes and a *hosting.net* prefix list of routes. The prefix list was a subset of the static routes and was referenced in the BGP export policy. The tribe simplified this chain by only putting the proper prefixes in the aggregate route list, then referencing this list in the BGP export policy rather than the prefix list. When addresses were added to or subtracted from the system, they only had to be modified in the aggregate route location rather in multiple places. The export policy then was:

```
policy-options {
  prefix-list BLACK-HOLE {
  }
  policy-statement ISP-B-out {
    term BLACK-HOLE-PREFIXES {
      from {
        prefix-list BLACK-HOLE;
      }
      then community add ISP-B-BLACKHOLE;
    }
  }
}
```

```

    term ADVERTISE-HOSTING-NET {
        from {
            protocol aggregate;
        }
        then accept;
    }
    term REJECT {
        then reject;
    }
}
community ISP-B-BLACKHOLE members 1234:666;
}

```

The new policy was very simple: there were three terms, one for blackholing prefixes that were being attacked, one for advertising the active aggregate routes (only those that were being used), and a reject that was a safety valve to prevent any other BGP routes from being sent to the Internet. For the test, the term *from protocol aggregate* was replaced with the term *from route-filter 3.9.89.0/24 exact* (our test address). The initial thought was to have a single policy for both ISPs, but then we saw that the community string for blackholing a prefix was different for each ISP, so we ended up with a pair of export policies, one for ISP A and one for ISP B.

Blackholing prefixes is a means for stopping a DOS attack at the ISP rather than at your network. You identify the address that is being attacked and ask the ISP to block all traffic to that address. Most ISPs have a community string that, when received tagged to a prefix, will discard any traffic to that address. When the attack is over, the prefix is withdrawn and the route is back in service. It's a shame that the response to a DOS attack is to take the destination address out of service (as that seems like the purpose of the DOS attack in the first place), but by sacrificing one customer, you can avoid all your customers suffering from an attack. Most ISPs will accept any length prefix for this feature, so as to minimize the impact of the remedy. In our case, when an attack is seen, two entries are needed in the SRX: a static route to the attacked prefix and the addition of the prefix to the prefix list called *BLACK-HOLE*. The static route is needed to make the route active (to allow BGP to advertise it). An example addition for stopping a DOS attack on address 123.123.123.0/30 would be:

```

set routing-options static route 123.123.123.0/30 reject
set policy-options prefix-list BLACK-HOLE 123.123.123.0/30
commit

```

Within 30 seconds, the traffic to 123.123.123.1 would stop being received at the client's network. With the old system, this was the only response to DOS attacks; with the SRX, the screen feature protects against the DOS attacks, but we kept the policy in place just to be on the safe side.

The agreement with the ISPs was that they were only going to advertise a default route to our client. That made the import policy a work of simplicity. The same policy was used for both ISPs. The policy was:

```

policy-options {
  policy-statement ISP-IN {
    term DEFAULT {
      from {
        route-filter 0.0.0.0/0 exact;
      }
      then accept;
    }
    term REJECT {
      then reject;
    }
  }
}

```

The next portion of the configuration was the actual BGP policy to connect to the ISP. This involved a number of steps. The ISP offered two 10 Gbps links to the location, each with a different address. These were installed on the SRX and tested for connectivity:

```

interfaces {
  xe-4/0/0 {
    unit 0 {
      description "ISP-B-XE-0/0/9";
      family inet {
        address 6.29.19.10/30;
      }
    }
  }
  xe-0/1/0 {
    unit 0 {
      description "ISP-B-XE-1/0/9";
      family inet {
        address 6.29.19.14/30;
      }
    }
  }
}

```

These interfaces needed to be added to the security zone, and the zone had to be updated to allow the incoming BGP traffic as well. The existing interface in the Internet zone was deleted and these were added. We also deleted a couple of the services allowed on the zone. The security zone was now close to the final configuration:

```

security {
  zones {
    security-zone Internet {
      host-inbound-traffic {
        system-services {
          ping;
          traceroute;
        }
        protocols {
          bgp;
        }
      }
    }
  }
}

```

```

    }
  }
  interfaces {
    xe-0/1/0.0;
    xe-4/0/0.0;
  }
}

```

The peer addresses for these links were 6.29.19.9 and 6.29.19.13 (oh joy, lucky 13!). With this information and the AS# (1234), we created the BGP group and configuration for this ISP:

```

protocols {
  bgp {
    log-updown;
    graceful-restart;
    group ISP-B {
      export ISP-B-OUT;
      import ISP-IN
      description "ISP contact mel-banc@800.555.1212";
      neighbor 6.29.19.9 {
        peer-as 1234;
      }
      neighbor 6.29.19.13 {
        peer-as 1234;
      }
    }
  }
}

```

The use of graceful restart allows the devices to commit or restart the routing process without causing a loss of the peering. Once this configuration was committed, we received our first good news of the tests—the ISP peer was up. However, we could not see our test prefix in any looking-glass site:

```

admin@SRX5600> show bgp summary
Groups: 2 Peers: 2 Down peers: 0
Table Tot Paths Act Paths Suppressed History Damp State Pending
inet.0 1 1 0 0 0
Peer AS InPkt OutPkt OutQ Flaps
6.29.19.13 1234 13 14 0 0
Last Up/Dwn State|#Active/Received/Accepted/Damped...
10:32
Establ
inet.0: 1/1/1/0
6.29.19.9 1234 13 14 0 0
10:32
Establ
inet.0: 1/1/1/0

```

The BGP peer was up and established, and the route summary command showed that a single route was received. The route that was received was displayed as:

```
admin@SRX5600> show route protocol bgp

inet.0: 55 destinations, 55 routes (38 active, 0 holddown, 17 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          [BGP/170] 10:45, localpref 100, from 6.29.19.13
                   AS path: 1234 I
```

This was our default route from the ISP. The issue was not in the incoming side but in the outgoing side. We did our normal troubleshooting and looked at the advertising routes:

```
admin@SRX5600> show route advertising-protocol bgp 6.29.19.13

inet.0: 55 destinations, 55 routes (38 active, 0 holddown, 17 hidden)
+ = Active Route, - = Last Active, * = Both
   Prefix                Nexthop          MED      Lclpref    AS path
* 3.9.89.0/24           Self                               I
```

This showed that we were at least sending the proper route to the ISP. But what was happening to the prefix? We called the ISP's help line and spoke to a kindred warrior who understood our dilemma and knew how to resolve the issue. The import policy on the ISP's router had an *exact* statement associated with the route filters. We had him change that to an *or-longer* statement.

Accessing our regular looking-glass site showed that our prefix was now being advertised as expected:

```
inet.0: 424675 destinations, 3230661 routes (423304 active, 6 holddown,
847 hidden)
3.9.89.0/24        (13 entries, 1 announced)
  *BGP             Preference: 170/-81
                   Next hop type: Indirect
                   Address: 0x9178c28
                   Next-hop reference count: 5358
                   Next hop type: Router, Next hop index: 1041
                   Next hop: 198.32.160.137 via xe-0/3/0.0, selected
                   Protocol next hop: 198.32.160.137
                   Indirect next hop: 9175b9c 1048583
                   State:
                   Local AS: 1273 Peer AS: 19151
                   Age: 6d 6:34:37      Metric: 0      Metric2: 0
                   AS path: 19151 80 1234 65001 I
                   AS path: Recorded
                   Communities: 19151:2000 19151:62001 19151:65020
                   Accepted
                   Localpref: 80
                   Router ID: 6.29.19.13
```

This showed that the route was being accepted by the ISP and advertised to the world; our AS# (65001) and router ID (6.29.19.13) showed in the output.



For those of you who do not have a regular looking-glass site, we use the [Cable&Wireless Worldwide Utilities site](#).

With the advertisement of the address to the Internet, we could test connectivity and speed to Internet sites from our test server. The policies used earlier in the testing allowed inbound and outbound traffic, so all testing was permitted.

The next item was testing against a DOS attack—the old mechanism was tested and shown to work as expected, but the screen feature had to be configured and applied to the Internet zone.

The attacks that the client received were destination-based DOS attacks: a single address was bombarded by traffic from the Internet. These attacks seemed to be primarily on servers that were hosting games—I guess it's the modern way of taking your ball and bat and going home (“If I can't play, no one can!”).

To protect the servers from this type of DOS attack, session limits were configured for the servers. The SRX monitors the number of simultaneous sessions and blocks all sessions above the threshold. Two session limits were added to the screen feature, one for the number of incoming (destination) sessions that were allowed for any one server in the network, and one for the number of outgoing (source) sessions that were allowed. The outgoing limit was considerably lower, as the majority of the traffic is incoming in nature. These session limits are for concurrent sessions.

Two session limits were arranged for each direction, which allowed the client to start with a lower limit and up the limit to a higher value if their customers were hitting the lower limit.

The screen configurations and the assignment to the Internet zone were as follows:

```
security {
  ids-option 100-SOURCE-LIMIT-SESSION {
    limit-session {
      source-ip-based 100;
    }
  }
  ids-option 1K-SOURCE-LIMIT-SESSION {
    limit-session {
      source-ip-based 1000;
    }
  }
  ids-option 1K-DEST-LIMIT-SESSION {
    limit-session {
```

```

        destination-ip-based 1000;
    }
}
ids-option 10K-DEST-LIMIT-SESSION {
    limit-session {
        destination-ip-based 10000;
    }
}
zones {
    security-zone Internet {
        screen 100-SOURCE-LIMIT-SESSION;
        screen 1K-DEST-LIMIT-SESSION;
    }
}
}

```

These screens blocked attacks to specific servers, but what if all servers or the entire network was being attacked? Additional screens were defined to block these other types of attacks as well. The full set of screens that were assigned to the Internet zone was:

```

security {
    screen {
        ids-option INTERNET-SCREEN {
            icmp {
                flood {
                    threshold 1000;
                }
                fragment;
                ip-sweep {
                    threshold 100;
                }
                large;
                ping-death;
            }
            ip {
                bad-option;
                block-frag;
                loose-source-route-option;
                record-route-option;
                security-option;
                source-route-option;
                spoofing;
                strict-source-route-option;
                tear-drop;
                unknown-protocol;
            }
            tcp {
                fin-no-ack;
                land;
                port-scan {
                    threshold 100;
                }
            }
        }
    }
}

```



```

        syn-ack-ack-proxy {
            threshold 1000;
        }
        syn-fin;
        syn-flood {
            alarm-threshold 1024;
            attack-threshold 200;
            source-threshold 1024;
            destination-threshold 2048;
            timeout 20;
        }
        udp {
            flood {
                threshold 100;
            }
        }
    }
}

```

These screens include all the bells and whistles required by this client. They are broken into four areas: ICMP, IP, TCP, and UDP.

The ICMP screens are looking for floods, scans, and malformed ICMP packets. The IP screens are looking for any option fields (we deleted the stream and timestamp options, as they are found in the client's environment), as well as IP spoofing and unknown protocols. The TCP screens include session limits for synchronization floods and port scans, and block common TCP attacks. The UDP screens only look for flooding from UDP packets.

Since only a single screen can be applied to a zone, the contents of the session-limiting screens were copied to the *Internet-screen* and applied to the Internet zone.

Once this configuration was committed, the CTO used some questionable Internet-based tools to scan and “attack” the SRX. These attacks were verified with the counters on the SRX via the command *show security screen statistics zone Internet*. The output of the command listed the screen items that were exercised:

```
user@host> show security screen statistics zone Internet
```

Screen statistics:	
IDS attack type	Statistics
ICMP flood	10
UDP flood	2
TCP winnuke	0
TCP port scan	91
ICMP address sweep	14
IP tear drop	0
TCP SYN flood	8
IP spoofing	34

ICMP ping of death	0
IP source route option	0
TCP land attack	0
TCP SYN fragment	14
TCP no flag	0
IP unknown protocol	6
IP bad options	0
IP record route option	0
IP timestamp option	0
IP security option	0
IP loose source route option	0
IP strict source route option	0
IP stream option	0
ICMP fragment	0
ICMP large packet	0
TCP SYN FIN	0
TCP FIN no ACK	0
Source session limit	0
TCP SYN-ACK-ACK proxy	0
IP block fragment	0
Destination session limit	3

At this point, we had a good BGP connection, good routing to and from the Internet, and a secure device. The tests that were performed at this stage were:

- Route advertisement
- Default route receipt
- Blackhole activation
- Failover between interfaces
- Load balancing between interfaces
- DOS attack mitigation

These tests, combined with the previous testing, allowed us to progress with confidence to the actual installation phase of the engagement. With the maintenance window phase, either luck was with us again or Murphy had taken a vacation, but in any case, we had only a few issues and all were solvable.

Production Configuration

The remainder of the configuration was created prior to actually cutting over the device to the production network.

The first task was configuring the groups for the ToR switches. We disabled the ISP links and copied and edited the test group configuration. As defined earlier in this chapter, each switch is placed in a group configuration and is assigned to a security zone of its own. Once all the switches had been configured and the configurations had been applied and committed, the zones look like this:

```
admin@SRX5600> show security zones terse
```

Zone	Type
Internet	Security
Switch_201	Security
Switch_202	Security
Switch_203	Security
Switch_204	Security
Switch_205	Security
...	...
Switch_231	Security
Switch_232	Security
Switch_TEST	Security

Each configuration group also added one or more static routes for the address space on the switch. These static addresses activated the aggregate routes in the routing table. The same table at this point showed:

```
admin@SRX5600> show route protocol aggregate
```

```
inet.0: 55 destinations, 55 routes (53 active, 0 holddown, 2 hidden)
+ = Active Route, - = Last Active, * = Both
```

3.9.90.0/23	*[Aggregate/130] 10:45
	reject
142.2.152.0/22	*[Aggregate/130] 10:45
	reject
142.6.220.0/22	*[Aggregate/130] 10:45
	reject
142.3.232.0/21	*[Aggregate/130] 10:45
	reject
142.5.16.0/21	*[Aggregate/130] 10:45
	reject
7.219.96.0/20	*[Aggregate/130] 10:45
	reject
143.22.64.0/20	*[Aggregate/130] 10:45
	reject

There are still a few hidden routes at this point, and those are the routes that have not been allocated to customers yet. The detailed display of these routes shows that they are hidden because of the lack of a child:

```
admin@SRX5600> show route hidden extensive
```

```
inet.0: 55 destinations, 55 routes (53 active, 0 holddown, 2 hidden)
3.9.89.0/23 (1 entry, 0 announced)
```

```

Aggregate
  Next hop type: Reject
  Next-hop reference count:
  State: <Hidden Int Ext>
  Local AS 65000
  Age: 3d 12:1:46
  Task: Aggregate
  AS path: I
  Flage:                               Depth: 0      Inactive

```

Once the switches were added, the current BGP connection was copied over from the legacy routers. This was the final portion of the configuration to be added to the SRX. The interfaces to *ISP-A* were also added to the Internet security zone at this point.

Upon review of the configuration, and looking at our best practices, one of the tribe saw that we had left a large hole open in our architecture: we had not closed down access to the routing engine. Yes, the client has a trustworthy set of customers, but security is security, and we had to add a firewall filter to protect the routing engine. There are many examples to choose from, but once again we went for simplicity. We designed a firewall filter to block incoming traffic to the loopback interface. The main components of the firewall were:

```

firewall {
  policer DOS-PROTECTION {
    if-exceeding {
      bandwidth-limit 32k;
      burst-size-limit 2k;
    }
    then discard;
  }
  family inet {
    filter LOCKDOWN {
      term ICMP {
        from {
          protocol icmp;
          icmp-type [ echo-reply echo-request time-exceeded
                     unreachable ];
        }
        then {
          policer DOS-PROTECTION;
          accept;
        }
      }
    }
    term TRACEROUTE {
      from {
        packet-length [ 82 40 ];
        protocol udp;
        port 33435-33523;
      }
      then {
        policer DOS-PROTECTION;
      }
    }
  }
}

```

```

        accept;
    }
}
term SSH {
    from {
        prefix-list {
            SECURE-HOSTING;
        }
        protocol tcp;
        port ssh;
    }
    then accept;
}
term RADIUS {
    from {
        prefix-list {
            SECURE-HOSTING;
        }
        protocol [ tcp udp ];
        port radius;
    }
    then accept;
}
term DNS {
    from {
        protocol udp;
        port domain;
    }
    then accept;
}
term SYSLOG {
    from {
        prefix-list {
            SECURE-HOSTING;
        }
        protocol udp;
        port syslog;
    }
    then accept;
}
term SNMP {
    from {
        prefix-list {
            SECURE-HOSTING;
        }
        protocol [ udp tcp ];
        port snmp;
    }
    then accept;
}
term FTP {
    from {

```

```

        prefix-list {
            SECURE-HOSTING;
        }
        protocol tcp;
        port [ ftp ftp-data ];
    }
    then accept;
}
term NTP {
    from {
        prefix-list {
            SECURE-HOSTING;
        }
        protocol udp;
        port ntp;
    }
    then accept;
}
term OSPF {
    from {
        protocol [ ospf igmp ];
    }
    then accept;
}
term BGP {
    from {
        protocol tcp;
        port bgp;
    }
    then accept;
}
term DISCARD-ALL {
    then {
        count COUNTER;
        discard;
    }
}
}
}
}
}

```

The filter is applied as an input filter to the loopback interface. The filter requires the creation of a prefix list for those addresses that can reach the routing engine and those that the routing engine should be able to reach. The entries in this prefix list were the management networks of the client's network:

```

policy-options {
    prefix-list SECURE-HOSTING {
        142.76.156.0/24;
    }
}

```

```
        142.76.155.0/24;  
        142.76.22.0/24;  
    }  
}
```

Finally, we had a complete, secure configuration that we were confident in taking to the maintenance window.

Cut-Over

After all the testing and preparations, the cut-over was something of an anticlimax—it all went smoothly, and as such, it will serve as a fitting end to this chapter and to the book.

On the early morning of the maintenance window, the tribe met, fully loaded with caffeine and doughnuts. We had created a maintenance plan that included the following steps:

1. Create a cut-over export policy for the SRX. This policy referenced an access list that included each of the switch-level prefixes (/24s).
2. Update the existing routers' prefix list with the same prefixes as above.
3. Test that all prefixes were advertised on the Internet.
4. Create a patching list for swapping the switches from the central switch to the SRX.
5. Create a cut sheet that associated a switch | interfaces | prefixes | sequence number.
6. Create a set of patch files on all the routers (old cores and SRX) that installed the appropriate switch group, new prefix list, and old prefix list.
7. Verify that the SRX BGP peering was operational.
8. Perform the patches (cable and configurations).
9. Move the old ISP to the SRX.
10. Monitor the operation.
11. Drink beer or sleep, depending on the time.

The plan was designed to minimize the customers' downtime (hopefully). With the cut-over plan in hand, we started the process. We began by repatching a single switch at a time, loading the appropriate patch file in all the devices. Once the commit was complete, we tested using the looking-glass site and pinged to the servers from the Internet, and vice versa.

After a couple of switches, we got bolder and started repatching multiple switches at a time and entering the appropriate patch files.

Once all the switches were attached to the SRX, we retested server access and verified that all prefixes were pointing to the new ISP and the SRX. When this was verified, the old ISP was moved to the SRX and the peering was verified. All went as planned, with only a couple of mispatched interfaces and one very upset customer (for some reason, the two-week notice given was not enough time to prepare for the maintenance).

The dual ISPs were monitored for incoming and outgoing traffic. We noticed that all outgoing traffic was leaving on one of the links of the old ISP, and none on the new ISP's links; the return traffic was coming in on a nearly equal basis between the two ISPs, but only on one link for each ISP.

What we found was that we needed to use load balancing between the outgoing paths. We added the typical load balancing policy to the forwarding table and the *multipath multiple-as* command on the ISP group. This resolved the load balancing need for the outgoing traffic, but the incoming traffic was still being sent on a single ISP link per ISP. We asked the CTO to send a request to each of the ISPs to request load balancing between the links for that ISP. Of course, this was not accomplished during the maintenance window.

We monitored the traffic and observed that the data rates returned to normal levels during the morning. The customers were all notified of the successful change and were requested to report any issues ASAP.

Conclusion

This engagement came to an end not with a bang, but a whimper. What looked like a disaster in the making turned into a secure, scalable network improvement. The discovery process and the use of simple configuration steps allowed the client to feel comfortable with the upgrade, and it served their customers well.

The tribe relaxed around a few beers, and we all slept without worries that night—including the CTO.

A

- access, 137
 - (see also Internet access)
 - example, 70
 - remote access VPNs, 142
 - routers, 113–117
- active/active
 - clustering, 146
 - with one-legged Reths, 88
 - without Reths, 87
- active/passive mode, 146
- address-persistent command, 158
- addresses
 - distributed network, 323
 - email server address resolution, 340
 - IP addressing, xiv
 - MX network configuration, 303
 - PCI-compliant data center, 226
- adjacency points, loopback interfaces, 201
- administrative domains, 138
- anti-spam ASP, 324
- AT&T class of service coding, 19
- attacks
 - signatures, 56
 - updating, 62
- authentication
 - BGP links, 304
 - RADIUS, 374

- source IP address, 324
- availability, 288

B

- backup site BNA, enterprise VPN example, 37
- bandwidth
 - about, 289
 - low bandwidth, 139
- banners, 115
- BGP (Border Gateway Protocol)
 - about, 15
 - authentication on BGP links, 304
 - configuration, 119
 - Internet access routing, 159–165
 - MX network configuration, 312–315
 - peering, 359
 - service provider engagement trade-offs, 182
- BGP/VPLS deployments, 182
- blackholing prefixes, 384

C

- class of service
 - enterprise VPN example
 - company profile, 4, 8
 - implementation, 18–31
- CLI commands, 42
- clustering
 - data center security configuration, 76–87

We'd like to hear your suggestions for improving our indexes. Send email to index@oreilly.com.

- Internet access configuration, 147–150
 - Internet access trade-offs, 145
 - command-line interface, xiv, 40–43
 - company profile, 175–184
 - design approach, 178–181
 - deployment, 180
 - management network, 181
 - MX connectivity, 179, 180
 - design trade-offs, 181
 - physical network topology, 176
 - services, 178
 - configuration, 184–218
 - data center security, 75–90
 - active/active with one-legged Reths, 88
 - active/active without Reths, 87
 - clustering, 76–87
 - testing, 89
 - EX4500 transit switch, 276–281
 - DCB, 279–281
 - interfaces and VLANs, 276–279
 - firewalls in a survivable Internet solution for a distributed network, 342–354
 - Internet access, 147–168
 - clustering, 147–150
 - routing, 159–168
 - security, 150–159
 - Internet access rebuild, 372–397
 - cut-over, 396
 - deployment scenario, 372
 - ISP link testing, 383–391
 - management staging and testing, 373–377
 - production, 391–396
 - top-of-rack switch testing, 377–382
 - layer 3 to layer conversion, 106–134
 - core routers, 123–129
 - CPE switch, 134
 - distribution router, 131
 - distribution switch, 129
 - management, 108–117
 - protocols, 118–123
 - rate control, 133
 - MX network deployment, 291–315
 - management, 291
 - policy, 303–311
 - protocols, 311–315
 - routing engine protection, 293–302
 - PCI-compliant data center, 233–251
 - EX4200, 233–238
 - firewall, 245–251
 - MX240, 239–245
 - QF3500 Fibre Channel gateway, 264–275
 - DCB, 272
 - interface, 270–272
 - management, 264–270
 - service provider engagement
 - boilerplate configuration, 184
 - EX boilerplate and interfaces, 193–199
 - Layer 3 VPN, 207–214
 - MBGP, 201
 - MPLS, 202
 - MX interfaces, 187–193
 - OBM, 217
 - OSPF, 199
 - RSVP, 204–207
 - VPLS, 214–217
 - connectivity, PCI-compliant data center, 251
 - content filtering, 139, 142
 - core routers, configuration, 123–129
 - core tier, 70
 - CPE switch, configuration, 134
 - CRM traffic, 5
 - cut-over, enterprise VPN example, 31
- ## D
- dark fiber replacement, 255–285
 - EX4500 transit switch configuration, 276–281
 - DCB, 279–281
 - interfaces and VLANs, 276–279
 - existing design, 255–259
 - proposed design, 259–264
 - advantages and benefits, 263
 - naming, 259
 - network quality, 260
 - network upgrade, 261
 - QF3500 Fibre Channel gateway configuration, 264–275
 - DCB, 272
 - interface, 270–272
 - management, 264–270
 - verification, 282–285
 - data center security, 67–91
 - about, 68–75
 - configuration, 75–90
 - active/active with one-legged Reths, 88
 - active/active without Reths, 87
 - clustering, 76–87

- testing, 89
- DCB (data center bridging)
 - about, 260
 - EX4500 transit switch configuration, 279–281
 - QF3500 Fibre Channel gateway configuration, 272
- DCBX (Data Center Bridge Exchange), 260
- defaults
 - gateway failure detection, 336
 - Internet access routing configuration, 166
 - route in MX network configuration, 306
 - VR, 155
- denial of service attacks, 363
- deployment
 - PCI-compliant data center, 251
 - service provider engagement, 180
- detector engine, updating, 59
- device count, 238
- distribution
 - router, 131
 - switch, 129
 - tier, 70
- DOS attacks, 363

E

- email server address resolution, 340
- Enhanced Transmission Selection (ETS), 260
- enterprise VPN example, 1–37
 - company profile, 2–10
 - class of service, 4
 - design trade-offs, 6–10
 - need for change, 4
 - network, 2
 - traffic flow, 3
 - implementation, 10–37
 - backup site BNA, 37
 - class of service, 18–31
 - cut-over, 31
 - main site, 32
 - prototype phase, 10–18
 - remote site JAX, 32–36
 - remote sites PHL and IAD, 36
- Ethernet, xiv
- ethtool commands, 42
- ETS (Enhanced Transmission Selection), 260
- EX connectivity, 180
- EX4200, PCI-compliant data center EX4, 233–238

- EX4500 transit switch configuration, 276–281
 - DCB, 279–281
 - interfaces and VLANs, 276–279
- export policies, OSPF, 164
- external routes, OSPF, 166

F

- failover, 80
- FCoE transit switch, 278
- feature interactions, 170
- Fibre Channel
 - about, 257
 - QFX, 263
- filters
 - example, 301
 - filter-based forwarding, 145
 - firewall, 296
- firewalls
 - distributed network, 324
 - filters, 296
 - managing, 138
 - PCI-compliant data center
 - configuration, 245–251
 - design, 231
 - trade-offs, 224
 - survivable Internet solution for a distributed network, 342–354
 - virtual structure, 150
- four-nines availability, 226
- functions, 138

G

- gateway failure detection, 336

H

- hot standby mode, 146

I

- IBR (Internet border router), Internet access
 - trade-offs, 144
- ICMP (Internet Control Message Protocol), xiv
- IDP systems, 39–65
 - IDP800 background, 40–47
 - command-line interface, 40–43
 - NSM, 45
 - web management interface, 43
 - Internet access trade-offs, 145

- support tasks, 47–63
 - daily tasks, 47–54
 - policies, 54–58
 - rulebase optimization, 58
 - updating attacks, 62
 - updating IDP appliance OS, 60
 - updating the detector engine, 59
- IDP800 background, 40–47
 - command-line interface, 40–43
 - NSM, 45
 - web management interface, 43
- ifconfig command, 42
- inter-silo traffic, 71
- inter-VLAN connections, 236
- interactions
 - features, 170
 - network interactions, 171
- interfaces
 - Internet access security, 151–156
 - mapping, 89
- Internet access, 137–173
 - configuration, 147–168
 - clustering, 147–150
 - routing, 150, 159–168
 - security, 150–159
 - design, 140–146
 - clustering, 145
 - filter-based forwarding, 145
 - IBR integration, 144
 - IDP systems, 145
 - routing, 144
 - implementation, 169–173
 - administrative issues, 173
 - feature interactions, 170
 - network interactions, 171
 - objective, 138
 - rebuild, 357–397
 - configuration, 372–397
 - existing network, 358–363
 - requirements, 358
 - solution options, 363–372
- Internet border router (IBR), Internet access trade-offs, 144
- Internet Control Message Protocol (ICMP), xiv
- Internet traffic, 5
- Internet VR, 155
- interoperability, OSPF, 172
- inventory control traffic, 5
- IP addressing, xiv

- ISPs (Internet Service Providers)
 - link testing, 383–391
 - MX network configuration, 305–311
 - working with, 171

K

- KISS (Keep It Super Simple), 179, 295

L

- label distribution protocol (LDP), paths, 183
- LAN (local area networking), compared to Fibre Channel, 257
- layer 3 to layer conversion, 93–135
 - configuration, 106–134
 - core routers, 123–129
 - CPE switch, 134
 - distribution router, 131
 - distribution switch, 129
 - management, 108–117
 - protocols, 118–123
 - rate control, 133
 - problem, 96–104
 - Q-in-Q framing, 99
 - VPLS overhead, 99–104
 - solutions, 104–106
- Layer 3 VPN, service provider engagement configuration, 207–214
- LDP (label distribution protocol), paths, 183
- link testing, ISPs, 383–391
- lo0.0, 108–113
- local area networking (LAN), compared to Fibre Channel, 257
- logging
 - Internet access security, 159
 - and monitoring, 139
- logical systems, 231
- loopback interfaces
 - adjacency points, 201
 - filters, 294

M

- maintenance
 - IDP systems, 39–65
 - IDP800 background, 40–47
 - support tasks, 47–63
 - PCI-compliant data center, 252

- manage, manage-ip and manager-ip commands, 348
- managed devices, 330
- management network, service provider engagement, 181
- management systems, 138
- management, MX configuration, 291
- manual failover, 138
- MBGP (multiprotocol BGP), service provider engagement configuration, 201
- MEDs (Multi-exit discriminators), MX network configuration, 305
- MIP translation, 349
- monitoring and logging, 139
- MPLS (multiprotocol label switching)
 - configuration, 119
 - reflector BGP deployment, 183
 - service provider engagement configuration, 202
 - service provider engagement trade-offs, 183
- MTU (maximum transmission unit)
 - moving, 105
 - restrictions, 105
- multimedia traffic, 5
- multipath multiple-as, 162
- multivendor networks, 68
- MX connectivity, 179
- MX interfaces, service provider engagement configuration, 187–193
- MX network deployment, 287–320
 - configuration, 291–315
 - management, 291
 - policy, 303–311
 - protocols, 311–315
 - routing engine protection, 293–302
 - final phases, 320
 - phase 1, 289
 - phase 2, 315–320
 - plans and topology, 288
- MX240 3D Universal Edge Router, 229
- MX240, PCI-compliant data center configuration, 239–245

N

- naming, Fibre Channel network, 259
- NAT (network address translation), Internet access security, 156
- next-hop self statement, 122
- nonrevertive LSP, 205

- NSM (Network and Security Management)
 - about, 45
 - versions, 62

O

- OBM (out-of-band management), service provider engagement configuration, 217
- office automation traffic, 5
- OOB network, Internet access routing configuration, 167
- OS, updating IDP appliance OS, 60
- OSI model, xiii
- OSPF (open shortest path first), 123
 - about, 6
 - Internet access routing, 166
 - interoperability, 172
 - MX network configuration, 312
 - service provider engagement configuration, 199
 - service provider engagement trade-offs, 181
 - survivable Internet solution for a distributed network, 330
- overhead, VPLS, 99–104

P

- paths, LDP versus RSVP, 183
- PBR (policy-based routing), 335
- PCI-compliant data center, 221–254
 - client goals, 222
 - configuration, 233–251
 - EX4200, 233–238
 - firewall, 245–251
 - MX240, 239–245
 - deployment, 251
 - design, 227–233
 - firewall layer, 231
 - routing layer, 229
 - switching layer, 227
 - trade-offs, 224
 - virtualization, 232
- PFC (Priority Flow Control), 260
- physical network topology, 176
- policers, 296
- policies
 - example, 82
 - IDP systems, 54–58
 - Internet access security, 151–156

- MX network deployment, 303–311
 - authentication on BGP links, 304
 - default route, 306
 - local preference, 305
 - locally assigned addresses, 303
 - MEDs, 305
 - prepending, 307
 - RFC 1918 prefixes, 304
 - subnets, 304
 - transit network, 308
- policy-based routing (PBR), 335
- port mirroring, 103
- PPVPN (provider-provisioned VPN), 4
- prepending, MX network configuration, 307
- prepositioning, 233
- production, configuration, 391–396
- profiles, class of service, 4
- protocols, 118–123
 - BGP, 119
 - MPLS, 119
 - OSPF, 123
- prototype phase, enterprise VPN example, 10–18

Q

- Q-in-Q framing, 99
- QCN (Quantized Congestion Notification), 261
- QF3500 Fibre Channel gateway configuration, 264–275
 - DCB, 272
 - interface, 270–272
 - management, 264–270
- QFX, Fibre Channel, 263

R

- RADIUS authentication service, 374
- rate control, configuration, 133
- reliability
 - MX connectivity, 179
 - SRX, 231
- remote access VPNs, 142
- remote locations, routing and survivability, 7
- remote management, 347
- remote site JAX, enterprise VPN example, 32–36
- remote sites PHL and IAD, enterprise VPN example, 36
- remove-private command, 162

- resilience, 139
- resource reservation protocol (see RSVP)
- Reths
 - active/active with one-legged Reths, 88
 - active/active without Reths, 87
- RFC 1918 prefixes, MX network configuration, 304
- RFC 4623, layer 3 to layer conversion, 104
- route preference manipulation, 165
- routers
 - about, xiv
 - access, 113–117
- routing
 - enterprise VPN example, 6
 - example, 71
 - Internet access, 159–168
 - BGP, 159–165
 - default, 166
 - OOB network, 167
 - OSPF, 166
 - security, 150
 - trade-offs, 144
 - PCI-compliant data center
 - design, 229
 - trade-offs, 225
 - protocols, 359–363
- routing engine protection, MX configuration, 293–302
- RPF checks, 335
- RSVP (resource reservation protocol)
 - paths, 183
 - service provider engagement configuration, 204–207
- rulebase optimization, IDP systems, 58

S

- scalability
 - examples, 139, 201, 364
 - PCI compliance, 223
- SCIO commands, 40
- sctop command, 42
- security, 67
 - (see also data center security)
 - firewall policies, 156
 - Internet access configuration
 - interfaces, zones and policies, 151–156
 - logging, 159
 - NAT, 156
 - routing instances, 150

- PCI compliance, 222
- security policy logs, 253
- service provider engagement, 175–219
 - company profile, 175–184
 - design approach, 178–181
 - design trade-offs, 181
 - physical network topology, 176
 - services, 178
 - configuration, 184–218
 - boilerplate configuration, 184
 - EX boilerplate and interfaces, 193–199
 - Layer 3 VPN, 207–214
 - MBGP, 201
 - MPLS, 202
 - MX interfaces, 187–193
 - OBM, 217
 - OSPF, 199
 - RSVP, 204–207
 - VPLS, 214–217
- session flows, 253
- single point of failure, 365
- SNMP (Simple Network Management Protocol)
 - about, 344
 - traffic and remote locations, 353
- stacking, 233
- static routes, 138, 162, 336
- STP (spanning tree protocol), 178
- subnets
 - IP addressing, xiv
 - MX network configuration, 304
- survivability
 - about, 288
 - enterprise VPN example, 6
 - PCI-compliant data center, 226
- survivable Internet solution for a distributed network, 321–355
 - firewall configurations, 342–354
 - original network architecture, 321–324
 - addressing, 323
 - firewalls, 324
 - internal connectivity, 323
 - WAN connectivity, 322
 - problem definition, 325
 - solution using existing routing protocols, 327–330
 - solution using OSPF over tunnels, 330
 - solution using static routes over tunnels, 333–342
- switches, xiv

- switching layer, PCI-compliant data center, 227
- symmetrical routing, 71

T

- TCP (Transmission Control Protocol), xiv
- testing
 - data center security configuration, 89
 - Internet access rebuild, 373–391
 - ISP link testing, 383–391
 - management staging and testing, 373–377
 - top-of-rack switch testing, 377–382
 - top-of-rack switch testing, 377–382
- track-ip, 337
- trade-offs
 - data center security, 72
 - Internet access, 143–146
 - clustering, 145
 - filter-based forwarding, 145
 - IBR integration, 144
 - IDP systems, 145
 - routing, 144
 - PCI-compliant data center, 224
 - service provider engagement, 181
- traffic
 - classes of, 5
 - flow, 3, 140
 - mapping, 10
- transit network, MX network configuration, 308
- transparent mode, 225
- trusted traffic, 140
- type 1 and type 2 OSPF external routes, 166

U

- UDP (User Datagram Protocol), xiv
- updating
 - appliance OS, 60
 - attacks, 62
 - detector engine, 59
- upgrading network for dark fiber replacement, 261

V

- verification, dark fiber replacement, 282–285
- virtual router configuration, 44
- virtual systems, 224

- virtualization
 - example, 68
 - PCI-compliant data center, 232
 - SRX, 231
- VLANs, EX4500 transit switch configuration, 276–279
- VPLS (virtual private LAN service)
 - overhead, 99–104
 - service provider engagement configuration, 214–217
 - service provider engagement trade-offs, 182
- VPN (virtual private network), 1
 - remote access, 142

- traffic, 140
- vrf-table-label command, 210
- vrouter, 342
- VSYS of Netscreen, 246

W

- WAN connectivity, 322
- web management interface, 43

Z

- zones, Internet access security, 151–156

About the Author

Peter Southwick is a senior network engineer at Proteus Networks, providing both professional services support and training for Proteus Networks customers. He is a JNCI, holds JNCIE-M #473, and other Juniper certifications in routing and security. He is a co-author of *Junos Enterprise Routing, Second Edition* (O'Reilly); co-author of the JNCIE-SP Workbook (Proteus Press); author of *Telecommunications: A Beginner's Guide*; co-author of *ISDN: Concepts, Facilities and Services* (both published by McGraw Hill); and contributing author to *The Handbook of Local Area Networks* (CRC Press). He lives with his wife and two daughters in the Champlain valley of Northern Vermont.

Colophon

The animal on the cover of *Juniper Networks Warrior* is a seawolf, or Atlantic wolffish (*Anarhichas lupus*). The seawolf is known by many names: Atlantic catfish, ocean catfish, devil fish, wolf eel, or sea cat. This marine fish is the largest of the wolffish family *Anarhichas*. Despite its frightening appearance, the seawolf poses a threat only when it feels threatened, often when caught out of water. The National Oceanic and Atmospheric Administration's National Marine Fisheries Service has named the seawolf a Species of Concern due to its rapidly depleting numbers—the result of bycatch and overfishing.

The wolffish can live close to a mile below sea level in temperatures below 40 degrees Fahrenheit. To survive in such low temperatures, the seawolf produces a natural anti-freeze that keeps its blood moving fluidly. The seawolf plays an important role in its environment, keeping other populations—sea urchin, green crabs, and other bottom-dwelling species such as cod—in check.

The most notable feature of the seawolf is its large teeth and strong jaw, used to eat hardshell mollusks, crustaceans, and echinoderms. Other fish are safe from the jaws of the seawolf, but cockles, sea clams, large hermit crabs, starfish, sea urchin, and green crabs are not.

The cover image is from *Wood's Animate Creatures*. The cover font is Adobe ITC Garamond. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSansMonoCondensed.