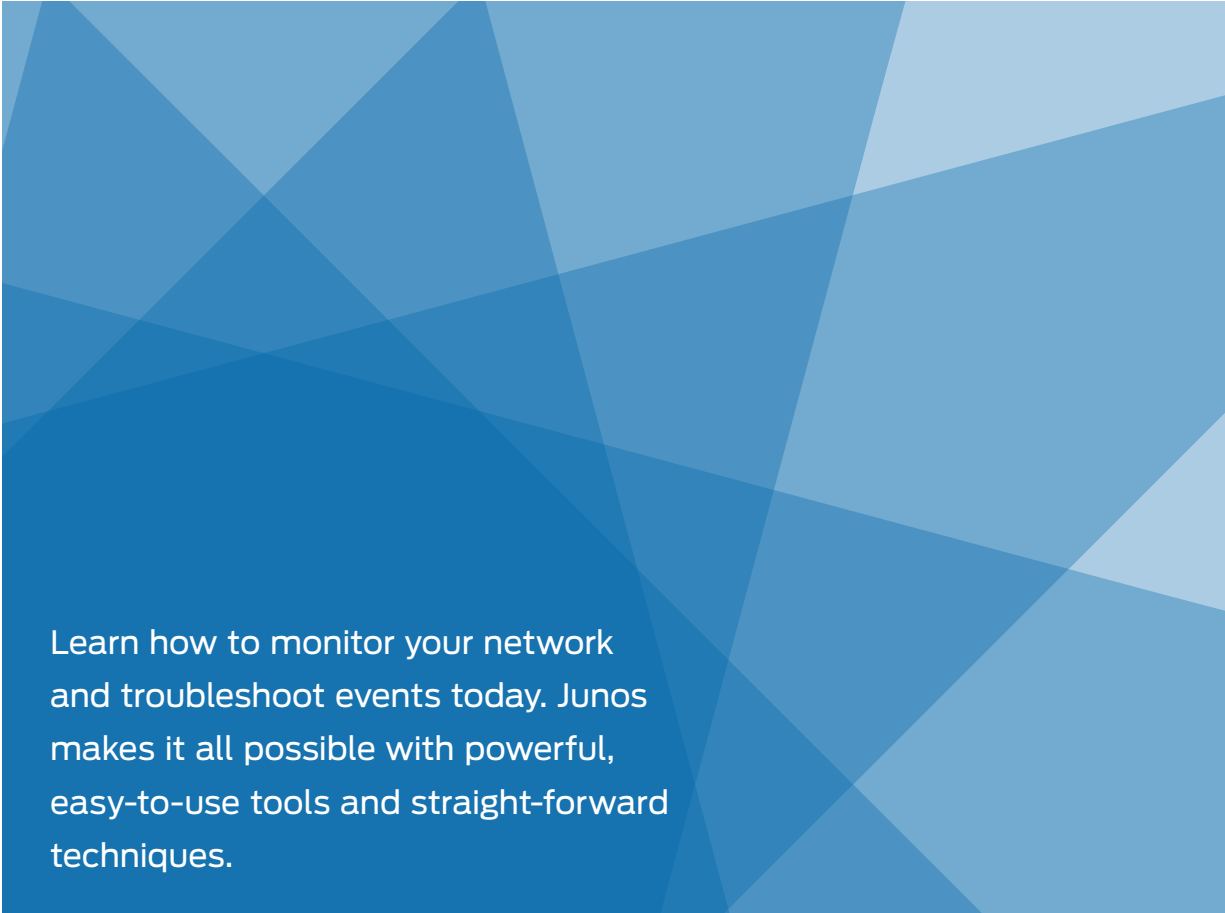


DAY ONE: JUNOS MONITORING AND TROUBLESHOOTING



Learn how to monitor your network and troubleshoot events today. Junos makes it all possible with powerful, easy-to-use tools and straight-forward techniques.

By Jamie Panagos and Albert Statti

DAY ONE: JUNOS MONITORING AND TROUBLESHOOTING

This Day One book advocates a process for monitoring and troubleshooting your network. The goal is to give you an idea of what to look for before ever typing a **show** command, so by book's end, you should know not only *what* to look for, but *where to look*.

Day One: Junos Monitoring and Troubleshooting shows you how to identify the root causes of a variety of problems and advocates a common approach to isolate the problems with a best practice set of questions and tests. Moreover, it includes the instrumentation to assist in root cause identification and the configuration know-how to solve both common and severe problems before they ever begin.

"This Day One book for configuring SRX Series with J-Web makes configuring, troubleshooting, and maintaining the SRX Series devices a breeze for any user who is new to the wonderful world of Junos, or who just likes to use its GUI interface rather than the CLI."

Alpana Nangpal, Security Engineer, Bravo Health

IT'S DAY ONE AND YOU HAVE A JOB TO DO, SO LEARN HOW TO:

- Anticipate the causes and locations of network problems before ever logging in to a device.
- Develop a standard monitoring and troubleshooting template providing technicians and monitoring systems with all they need to operate your network.
- Utilize the OSI model for quick and effective troubleshooting across different protocols and technologies.
- Use the power of Junos to monitor device and network health and reduce network downtime.
- Develop your own test for checking the suitability of a network fix.

Juniper Networks Day One books provide just the information you need to know on day one. That's because they are written by subject matter experts who specialize in getting networks up and running. Visit www.juniper.net/dayone to peruse the complete library.

Published by Juniper Networks Books

ISBN 978-1-936779-04-8



9 781936 779048



7100 1241

JUNIPER
NETWORKS®

Junos® Fundamentals

Day One: Junos Monitoring and Troubleshooting

By Jamie Panagos & Albert Statti

<i>Chapter 1: Root Cause Identification</i>	<i>5</i>
<i>Chapter 2: Putting the Fix Test to work</i>	<i>17</i>
<i>Chapter 3: CLI Instrumentation</i>	<i>29</i>
<i>Chapter 4: System Monitoring and Troubleshooting</i>	<i>39</i>
<i>Chapter 5: Layer 1 and Layer 2 Troubleshooting</i>	<i>55</i>
<i>Chapter 6: Layer 3 Monitoring</i>	<i>75</i>
<i>Chapter 7: Layer 3 Troubleshooting</i>	<i>99</i>

© 2011 by Juniper Networks, Inc. All rights reserved. Juniper Networks, the Juniper Networks logo, Junos, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. Junose is a trademark of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice. Products made or sold by Juniper Networks or components thereof might be covered by one or more of the following patents that are owned by or licensed to Juniper Networks: U.S. Patent Nos. 5,473,599, 5,905,725, 5,909,440, 6,192,051, 6,333,650, 6,359,479, 6,406,312, 6,429,706, 6,459,579, 6,493,347, 6,538,518, 6,538,899, 6,552,918, 6,567,902, 6,578,186, and 6,590,785.

Published by Juniper Networks Books
 Writers: Jamie Panagos and Albert Statti
 Editor in Chief: Patrick Ames
 Copyediting and Proofing: Nancy Koerbel
 Junos Program Manager: Cathy Gadecki

ISBN: 978-1-936779-04-8 (print)
 Printed in the USA by Vervante Corporation.
 ISBN: 978-1-936779-05-5 (ebook)

Version History: v3 January 2011
 4 5 6 7 8 9 10 #7100124

About the Authors

Jamie Panagos is a Senior Network Consultant for Juniper Networks and specializes in the design, implementation, and operation of datacenter, enterprise, and service provider networks. Jamie has over 10 years of experience on some of the largest networks in the world and has participated in several influential industry communities including NANOG, ARIN and RIPE. He holds JNCIE-MT #445 and JNCIE-ER #50.

Albert Statti is a Senior Technical Writer for Juniper Networks and has produced documentation for the Junos operating system for the past nine years. Albert has developed documentation for numerous networking features and protocols, including MPLS, VPNs, VPLS, and Multicast.

Authors Acknowledgments

The authors would like to take this opportunity to thank Patrick Ames, whose direction and guidance was indispensable. To Nathan Alger, Lionel Ruggeri, and Zach Gibbs, who provided valuable technical feedback several times during the development of this booklet, your assistance was greatly appreciated. Finally, thank you to Cathy Gadecki for helping turn this idea into a booklet, helping with the formative stages of the booklet, and contributing feedback throughout the process. There are certainly others who helped in many different ways and we thank you all.

This book is available in a variety of formats at: www.juniper.net/dayone.
 Send your suggestions, comments, and critiques by email to dayone@juniper.net.
 Follow the Day One series on Twitter: @Day1Junos

What you need to know before reading this booklet.

- ✓ A solid understanding of the topology, traffic flows, and protocols used on your network.
- ✓ A familiarity with the Junos CLI.
- ✓ Experience with network monitoring protocols such as syslog and SNMP.
- ✓ An understanding of Network Management Systems, what they do, and how they do it.
- ✓ Awareness of the OSI model and how it applies to network protocols and elements.

After reading this booklet, you'll be able to:

- ✓ Anticipate the causes and locations of network problems before ever logging in to a device.
- ✓ Develop a standard monitoring and troubleshooting template providing technicians and monitoring systems with all they need to operate your network.
- ✓ Utilize the OSI model for quick and effective troubleshooting across different protocols and technologies.
- ✓ Use the power of Junos to monitor device and network health and reduce network downtime.
- ✓ Develop your own test for checking the suitability of a network fix.

NOTE We'd like to hear your comments and critiques. Please send us your suggestions by email at dayone@juniper.net.

About Junos

Junos® is a reliable, high-performance network operating system for routing, switching, and security. It reduces the time necessary to deploy new services and decreases network operation costs by up to 41%. Junos offers secure programming interfaces and the Junos SDK for developing applications that can unlock more value from the network.

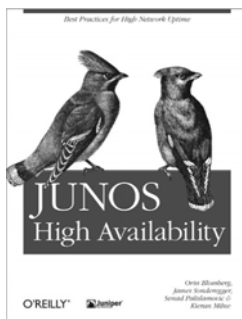
- ✓ Junos is one system, designed to completely rethink the way the network works.
- ✓ One operating system: Reduces time and effort to plan, deploy, and operate network infrastructure.
- ✓ One release train: Provides stable delivery of new functionality in a steady, time-tested cadence.
- ✓ One modular software architecture: Provides highly available and scalable software that keeps up with changing needs.

Running Junos in a network improves the reliability, performance, and security of existing applications. It automates network operations on a streamlined system, allowing more time to focus on deploying new applications and services. And it's scalable both up and down – providing a consistent, reliable, stable system for developers and operators. Which, in turn, means a more cost-effective solution for your business.

About the Junos Fundamentals Series

This Day One series introduces the Junos OS to new users, one day at a time, beginning with the practical steps and knowledge to set up and operate any device running Junos. For more info, as well as access to all the Day One titles, see www.juniper.net/dayone.

Special Offer on Junos High Availability



Whether your network is a complex carrier or just a few machines supporting a small enterprise, *Junos High Availability* will help you build reliable and resilient networks that include Juniper Networks devices. With this book's valuable advice on software upgrades, scalability, remote network monitoring and management, high-availability protocols such as VRRP, and more, you'll have your network uptime at the five, six, or even seven nines – or 99.99999% of the time.

- ✓ www.oreilly.com. Use promo code **JHAVT** for a 35% discount.

Chapter 1

Root Cause Identification

The Fix Test11

This Book’s Network..... 13

Summary 15

The primary goal when troubleshooting any issue is root cause identification. This section discusses an approach for using clues and tools to quickly identify the root cause of network problems. This approach should help novice network engineers all the way to senior network engineers to reduce their investigation time, thus reducing downtime, and in the end, cost.

Before you ever log into a device or a network management system, it's possible to anticipate the nature of the problem, where you need to look, and what to look for. That's because you've been asking yourself a set of basic questions to understand the characteristics of the problem.

NOTE You still might not find a resolution to your problem, but you should be able to narrow down the options and choices by adhering to a set of questions that don't necessarily change over time. Their application is as much about consistency in your network monitoring and troubleshooting as it is about the answers they may reveal.

TIP Due to the complexity of computer networks, the equipment you suspect to be the cause might be functioning normally and the real root of the difficulty is in equipment in a different layer of the network. Routers can cause Layer 4 problems; switches can cause problems that would normally appear to be Layer 3 problems. In short, you might simply be looking in the wrong place. So don't throw out your suspicions, just apply them three dimensionally.

Figures 1-1a, 1-1b, 1-1c, and 1-1d illustrate how to approach monitoring and troubleshooting Juniper Networks devices and networks. Each figure begins with a problem scope. For example, Figure 1-1a illustrates how to approach a networking problem in which a single user is having difficulty connecting to a single destination. You can then narrow the problem down to the type of traffic affected, to whether the problem is constant or sporadic, and finally to where you can focus your troubleshooting efforts.

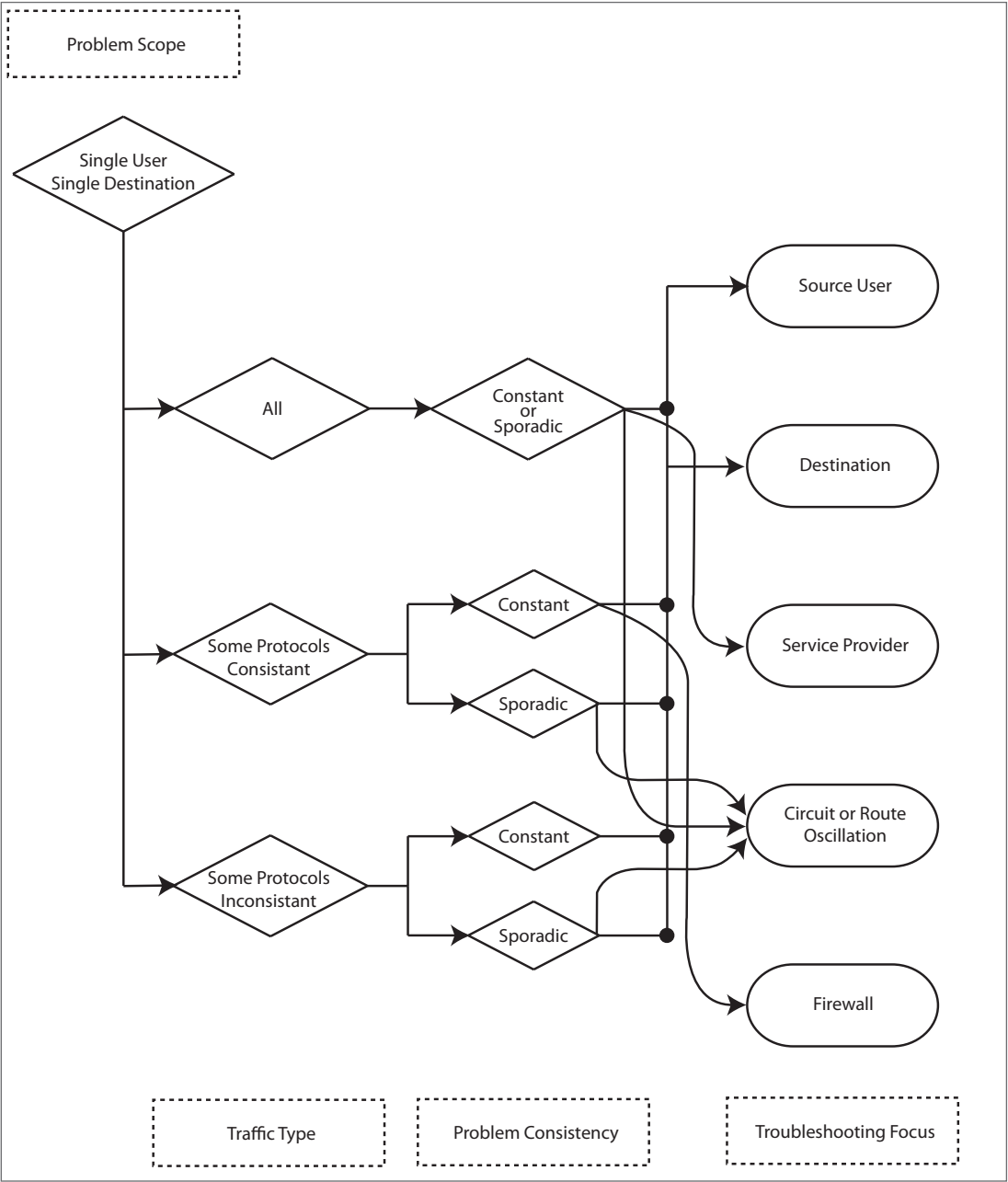


Figure 1-1a Single User Having Difficulty Connecting to a Single Destination

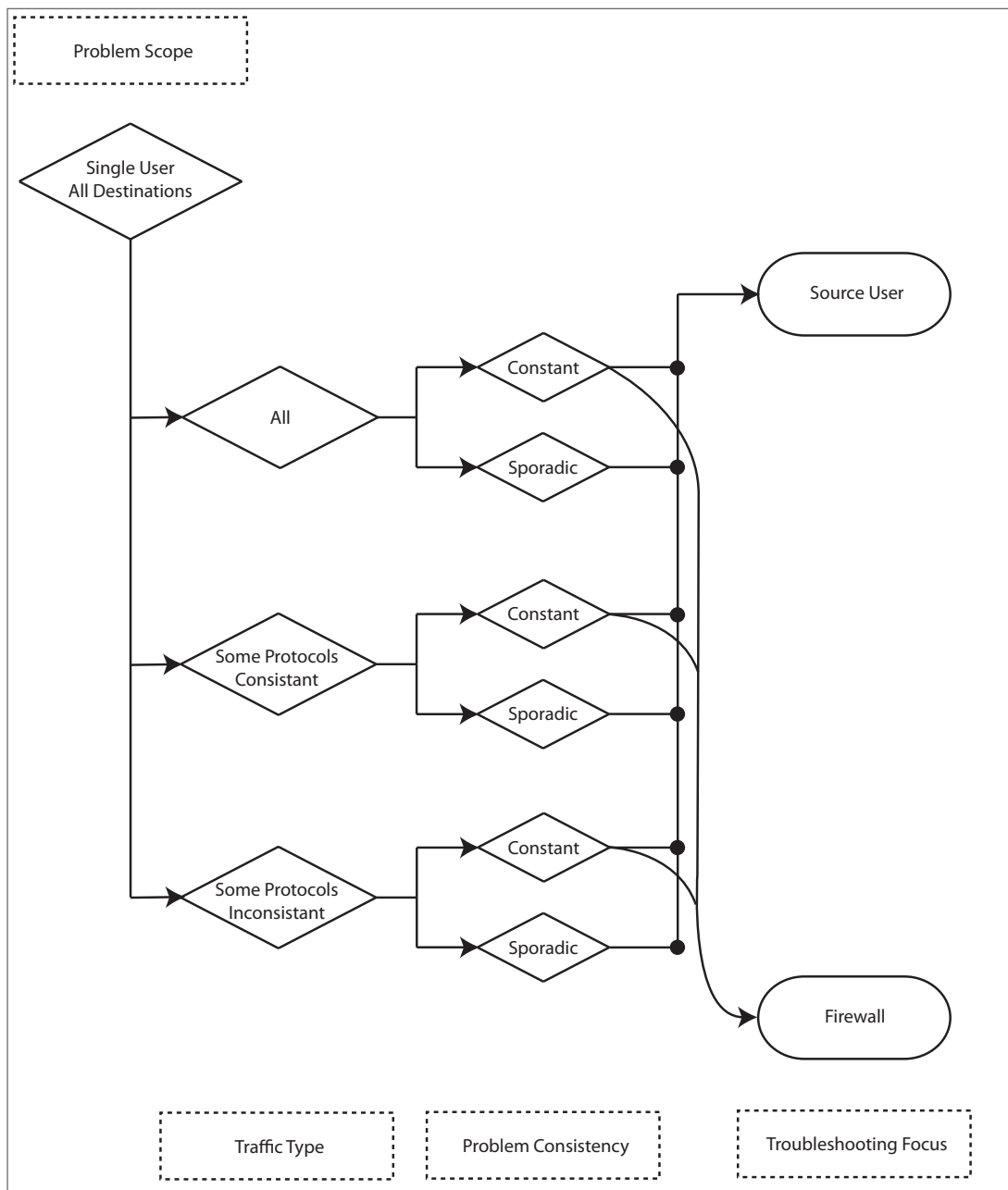


Figure 1-1b Single User Having Difficulty Connecting to all Destinations

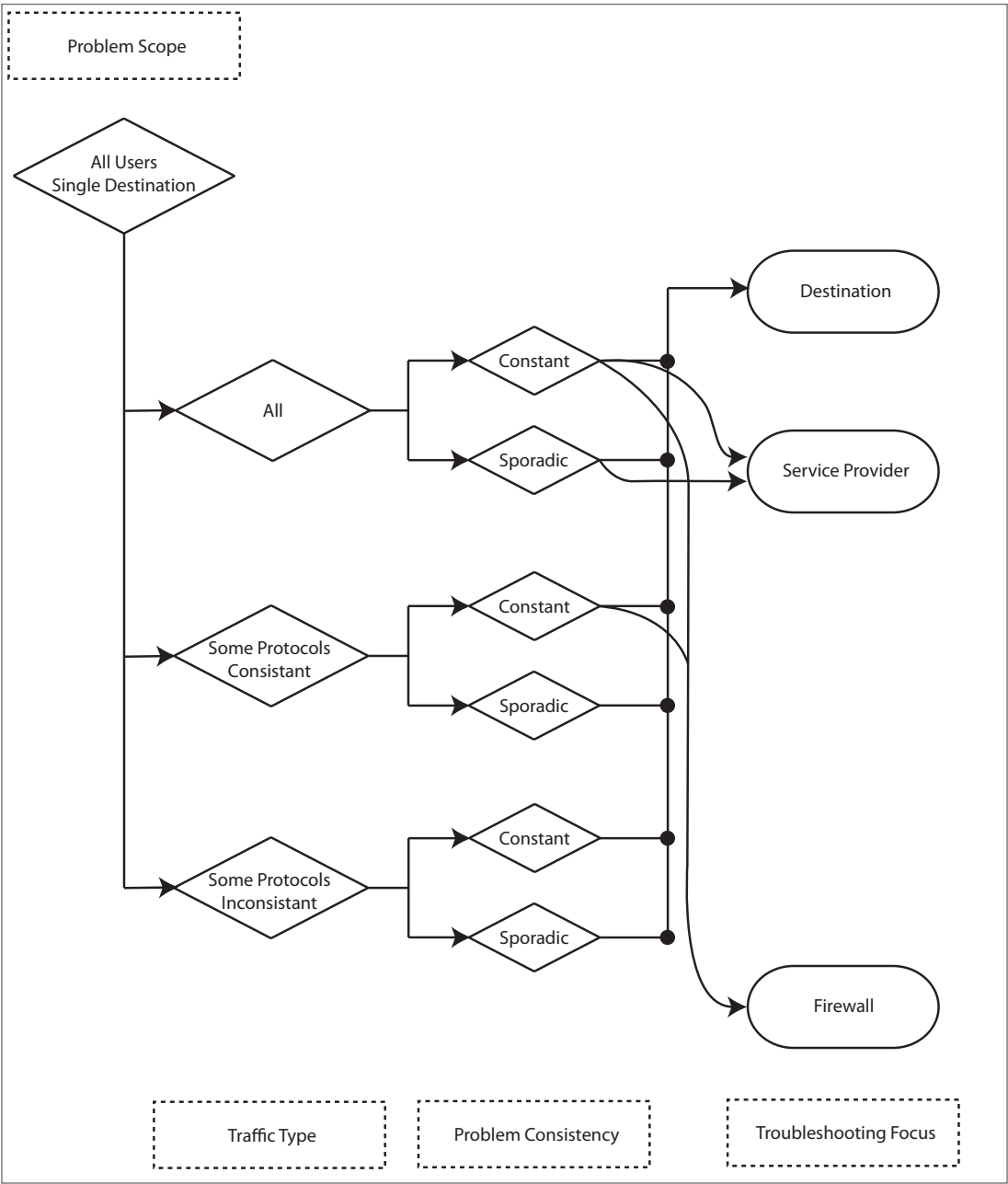


Figure 1-1c All Users Having Difficulty Connecting to a Single Destination

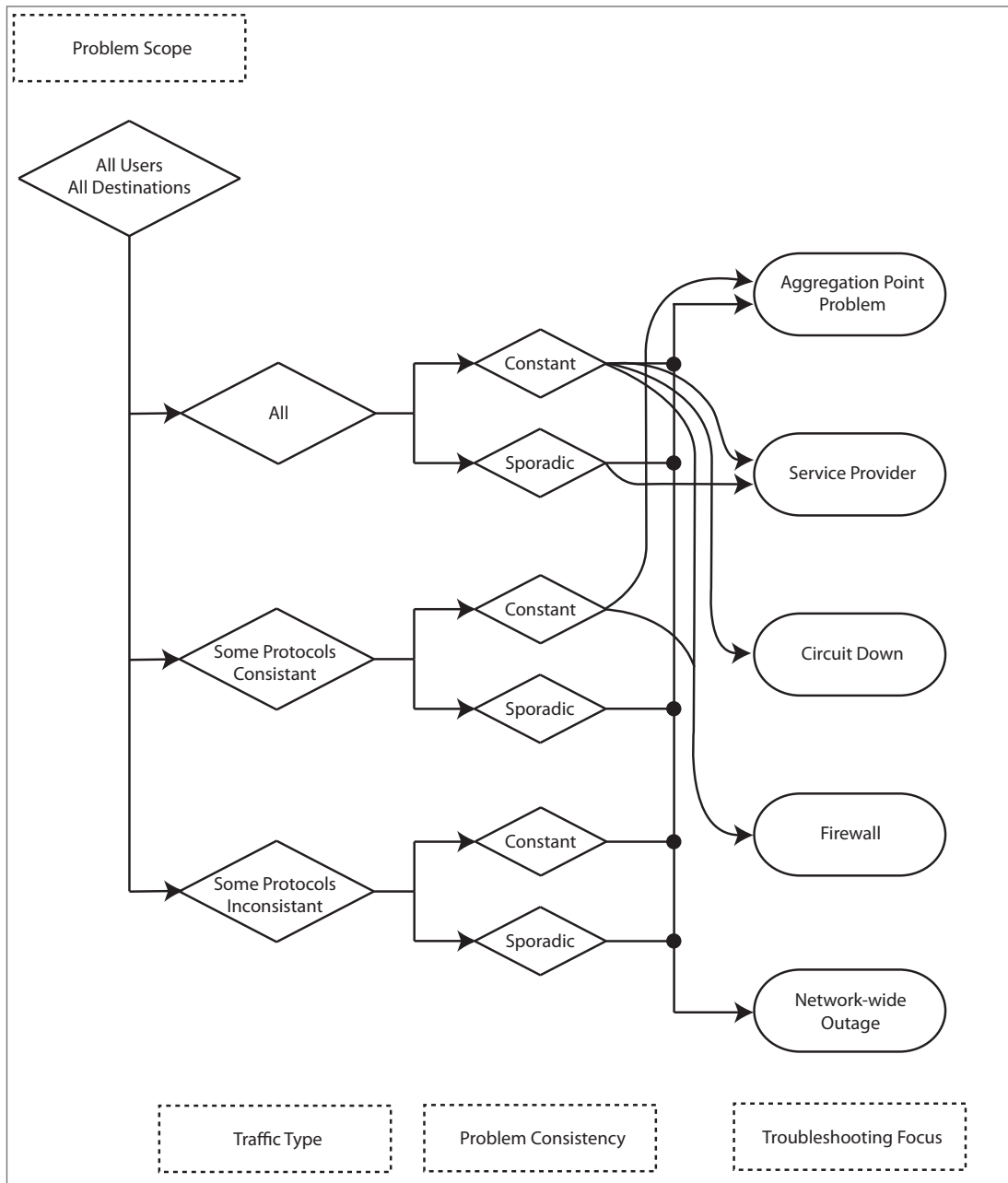


Figure 1-1d All Users Having Difficulty Connecting to all Destinations

TIP These figures help you to identify the scope and possible cause of a network outage. By using this information in conjunction with the Fix Test that follows, you should be able to more quickly isolate problems and restore service to your customers.

The Fix Test

Throughout this booklet we're going to apply the same set of questions, called the Fix Test, to ourselves and our work. The set of questions runs something like this (and we encourage you to create your own Fix Test using these as an example).

What is the Scope of the Problem?

The scope of an outage can mean many things to many people. Some people may declare it's an apocalypse if their primary account is occasionally slow to download a single website, while others might raise an alarm only when their entire network is down. What you should look for with this question is an objective view of the problem, absent emotion. This is the most important initial aspect of an outage to understand.

How Many Distinct Source Networks are Affected?

What Destinations are Involved?

You should then be able to determine if the problem is at the source, the destination, or is something larger. If it is a single user (or network) reporting problems to "everything," you should focus your efforts on the network elements close to the source. If many people are reporting problems to a single destination (or network), the problem is likely close to the destination, or is potentially the result of a problem at a network interconnect such as a peering point. If you can't seem to isolate the problem to either the sources or destination, the event is probably network-wide and it's probably time to hit the emergency button.

Who Reported the Problem First?

This question can help identify where (geographically as it relates to the network) the problem started. Whether the problem is source related, destination related, or a network-wide outage, this question can help narrow down where you should begin looking.

What Type(s) of Traffic is Affected?

Answering this question can help you understand at which OSI model network layer the problem is happening. Total loss of connectivity usually indicates the problems are at Layer 3 or perhaps a circuit is down. Layer 2 problems are generally protocol agnostic, but rarely cause a complete outage. Upper layer (Layers 4, 5, 6, and 7) problems are often caused by firewall issues.

The answer to this question should allow you to identify not only the area where you should focus your effort, but also the device type. Layer 2 problems typically mean you should focus on Ethernet switches or Layer 2 errors on the routers and end-host ports. If it's a Layer 3 problem, you need to check the routers and the IP stack on the end-hosts.

Is the Problem Constant or Sporadic?

Constant problems are usually caused by either errant configurations or persistent problems such as hardware failures. Sporadic problems generally indicate an oscillating issue, such as route or circuit-flapping or perhaps a traffic spike. Again, the answer to these questions helps you to figure out where you should start looking.

For constant problems, you should check logs to see if there were any changes made to the network recently or consult with the operations center to find out if there was any maintenance planned when the outage began. If not, the next step would be to try to identify if the hardware is causing the problem. Answers to the previous questions should assist in identifying where to begin the search.

Sporadic problems are a bit harder to nail down. Spiking traffic should be easy to identify, especially if you have a network management suite to poll the network for interface transmit and receive rates. The scope of the problem should help identify potential interfaces for further investigation.

If the problem is still not apparent, it is likely an oscillating problem. Syslog and other NMS utilities should help you identify a flapping link and the answers to the previous questions should give you a pretty good idea of where to look. If it's a route-oscillation problem, you will have to walk the path to the destination and back to the source (remember, its bidirectional!) as thoroughly as possible, monitoring each hop for changes in the routing for each network in question.

This Book's Network

This book uses the topology shown in Figure 1-2 as its core network. It was conceived to show off both enterprise networks (albeit at the larger scale) and a sense of the large networks at carriers and service providers. The same monitoring and troubleshooting skills should apply equally no matter the size of your network, just on a larger (and busier) scale.

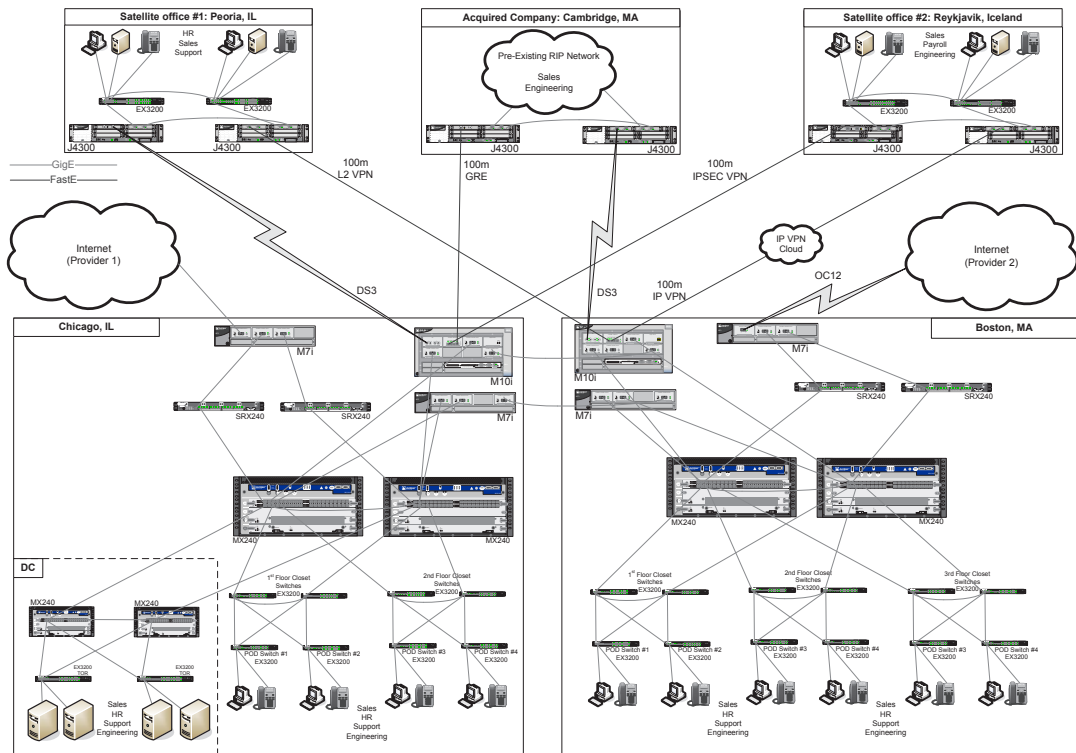


Figure 1-2 The Topology Used for this Book's Network

Physical Design

The physical design depicted in this topology represents a fairly simple enterprise network supporting two main sites (Chicago and Boston), two remote sites (Peoria, IL and Reykjavik, Iceland), and a small datacenter in the Chicago site. The main goals of this physical design were to provide all possible redundancy, while allowing for scaling.

The core of the network features two M10i routers, four M7i routers as well as four MX240 Ethernet aggregation routers. The M10i routers provide primary connectivity between the two main sites and terminate WAN connections to the remote offices over a variety of technologies. A single M7i in each site provides connectivity to the Internet, which will later be protected by a pair of SRX firewalls (right now the SRX are not doing any filtering) and the other provides redundant connectivity to the opposite main site. Finally, the MX240 routers aggregate the closet EX-Series Ethernet switches in each main site and serve as a Layer 3 boundary between the datacenter and the rest of the network. In the remote sites, J Series routers serve as the gateway routers and an aggregation point for the EX Series switches. Additionally, an acquired company has been connected to the core M7i routers in a method similar to that used for the remote offices. This site has a slightly different architecture. This design provides both redundancy (at the chassis level) and allows for scaling as the modular design and chassis selections allow for increased bandwidth and additional edge aggregation devices without the need for expensive hardware replacement within the core.

Logical Design

Like the physical design, the logical design was built to provide redundancy while allowing for fast convergence and an easy path for future deployments such as MPLS, multicast, and IPv6.

Satellite Sites

You may have also noticed that the remote offices are connected not only by traditional WAN circuit technologies, but also by logical connections providing pseudo-wires using IPSec and Layer 2 VPN technologies. From the perspective of the rest of the network, these connections are the same as any other physical media, but since these are logical connections, there is an impact on monitoring and troubleshooting.

IGP

The main IGP in this design will be OSPF. OSPF runs in a single area (area 0) on the MX and M series routers. The J Series routers in the remote offices also run OSPF, but the two OSPF domains are separate and the acquired site has historically run RIP. OSPF is used because of its relative ease of configuration, its convergence characteristics, its support of MPLS and its familiarity for our operations teams. IS-IS would work equally well. The decision to choose OSPF or IS-IS often comes down to comfort level and experience with the protocol. When all other factors are equal, familiarity is a perfectly valid basis for choosing a protocol.

BGP

BGP supports various functions in this network. As the network is multi-homed to the Internet, external BGP (eBGP) is run with the service providers. For this case, AS-path prepending and local-preference are used to influence routing decisions such that the Chicago Internet connection is preferred over the Boston connection. Internally, all M, MX, and J Series routers run internal BGP (iBGP) in a full-mesh. The remote offices and the acquired company redistribute their local IGP routes into BGP and BGP into their local IGPs. eBGP is also used with a third service provider which provides an MPLS IP VPN service for redundant connectivity to the Iceland site.

Summary

The goal of creating a monitoring and troubleshooting process is to give you an idea of what to look for before ever typing a `show` command. It should give you a head start not only in where to look, but in what to look *for*. It also allows you to preemptively contact additional personnel and, if necessary, event management groups to begin any triage and notification protocols. The additional personnel should speed resolution of the problem.

TIP Front-line support can better handle issues when they are aware of them before the phone rings, and their feedback to operations and engineering can assist in isolating and resolving the problem.

Throughout this book, the approach to identify the root causes of a variety of problems remains the same. This book describes an approach to isolating a problem as it relates to these questions and includes the instrumentation used to assist in root cause identification.

Finally, management and monitoring techniques are reviewed to pro-actively look for issues before they impact your customers, and how to use these techniques during an outage.

All monitoring and troubleshooting techniques described in this book are based on Junos OS Release 10.1.

Chapter 2

Putting the Fix Test to Work

Traffic Engineering and Overutilization (abbr.) 18

The Fix Test 27

Summary 28

Once you have identified the root cause of a problem (Chapter 1), the next step is to resolve it. Two types of fixes are discussed in this chapter: short-term fixes and long-term fixes.

These are intentionally generic terms, but it will be demonstrated that short-term fixes, and yes, at times, even hacks are acceptable resolutions as long as they meet our book's key Fix Test criteria:

- The fix does not cause other problems.
- The fix survives a reboot.
- The fix is well communicated.
- The fix is operationally understandable.
- And, the fix is replaced with a long-term fix in a reasonable amount of time.

Assuming these requirements are met, a short-term fix is completely acceptable. The main goal of any fix (short-term or long-term) should always be the *quick restoration of services*.

Traffic Engineering and Overutilization Troubleshooting

A great example of our Fix Test approach is traffic-engineering in the short-term and capacity upgrades in the long-term. Take our network, for example, as shown in Figure 1-2. The full Internet routing table is received through BGP from both of the service-providers and we use local-preference to select Chicago as our primary exit point, with the Boston peering point serving as a backup. AS path pre-pending is used to ensure that return traffic also takes the Chicago connection, but during peak hours, users complain that their Internet access is slow.

As the problem exists for all users and all external destinations, one can only guess that the problem is not specific to any one user and that the problem is likely at some aggregation point (like an exit point!). One can also guess that the problem is traffic level related, because it is sporadic.

Using either the network management (Figure 2-1) suite or by checking traffic levels on the CLI (following Figure 2-1) you can see that the network is over-utilizing the gigabit Ethernet link outbound to the primary service-provider in Chicago. A network operator would be presented with a graph similar to the one shown in Figure 2-1.

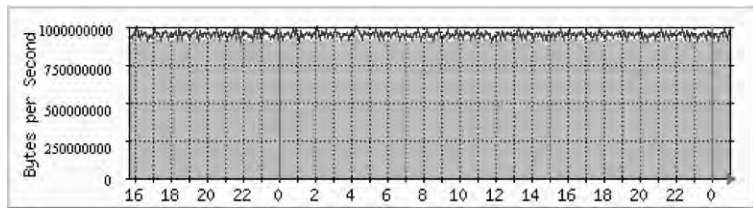


Figure 2-1 Chicago-edge-1-as-44 Utilization Graph

If the network operator were to check the interface statistics using the Junos CLI, it would also show the problem, specifically the input and output rates.

```
ps@chicago-edge-1> show interfaces ge-0/0/9
Physical interface: ge-0/0/9, Enabled, Physical link is Up
  Interface index: 137, SNMP ifIndex: 118
  Description: Connection to isp-1
  Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps, MAC-REWRITE Error: None,
  Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled,
  Auto-negotiation: Enabled, Remote fault: Online
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Current address: 00:19:e2:25:b0:09, Hardware address: 00:19:e2:25:b0:09
  Last flapped   : 2009-10-13 13:51:19 PDT (2d 20:44 ago)
  Input rate    : 3217455289 bps (13928377 pps)
  Output rate   : 9843638222 bps (37520944 pps)
  Active alarms  : None
  Active defects : None
```

Looking at the bolded output above, you can see that the outbound connection to the provider is near line rate. The CLI output is a snapshot, so running this command several times is recommended to get a better understanding of the true traffic situation. And if you use your network management system to look at Boston, you would see something akin to Figure 2-2.

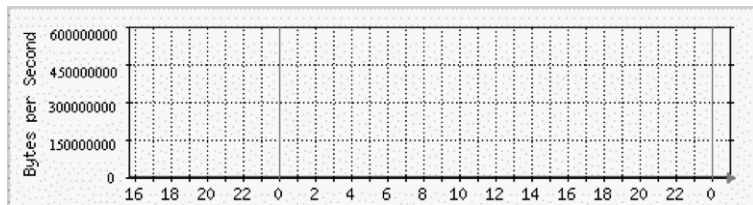


Figure 2-2 Boston-edge-1-as-107 Utilization Graph

Using the Junos CLI, the `show interfaces` command will display the input and output rates for the Boston connection. While a network management system is the right tool to actively monitor the network and alert on errors, nothing can replace the CLI for immediate, specific information gathering. The combination of these two tool-sets provides for the quickest isolation and remediation of network issues.

```
ps@boston-edge-1> show interfaces so-3/3/2
Physical interface: so-3/3/2, Enabled, Physical link is Up
  Interface index: 167, SNMP ifIndex: 151
  Description: Connection to isp-2
  Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode, Speed: OC12,
  Loopback: None, FCS: 16, Payload scrambler: Enabled
  Device flags   : Present Running
  Interface flags: Point-To-Point SNMP-Traps Internal: 0x4000
  Link flags     : Keepalives
  Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
  Keepalive: Input: 0 (never), Output: 0 (never)
  LCP state: Down
  NCP state: inet: Not-configured, inet6: Not-configured, iso: Not-configured, mpls:
  Not-configured
  CHAP state: Closed
  PAP state: Closed
  CoS queues    : 8 supported, 4 maximum usable queues
  Last flapped  : 2009-10-14 07:03:58 PDT (2d 03:37 ago)
Input rate    : 1207 bps (6 pps)
Output rate   : 2943 bps (17 pps)
  SONET alarms  : None
  SONET defects : None
```

Looking at the same bit rates inbound and outbound (bolded above) on the Boston provider circuit, you see that it is nearly empty.

You could (and should) request an upgrade of your peering capacity with your primary provider, but that can take weeks. You need a short-term solution to this overutilization problem. Since there is another egress point in Boston to our secondary service-provider, you could change your routing policy, forcing some outbound traffic to use the backup link, alleviating the over-utilization of the Chicago circuit.

Use the `show bgp summary` command on your Boston edge router to see that you are not currently selecting any routes from your Boston service-provider.

```
ps@boston-edge-1> show bgp summary
Groups: 2 Peers: 3 Down peers: 0
Table      Tot Paths  Act Paths Suppressed  History Damp State  Pending
inet.0     13986      6993      0           0         0         0
Peer       AS         InPkt    OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Damped...
```

```

10.25.30.1      10      7013      7112      0      0      13:07 6993/6993/0
0/0/0
10.25.30.3      10       28      7111      0      0      12:41 0/0/0
0/0/0
18.32.16.102    107     7006     6277     0      0      8:10 0/6993/0
0/0/0

```

The output of `show bgp summary` shows that AS 107 is sending 6993 routes, but none of these routes are active on boston-edge-1. This is shown in the field that is currently displaying “0/6993/0”. The first value is active routes, the second is received routes and the last shows the number of dampened routes. To review the policy applied to this BGP session, use the `show configuration protocols bgp group [group-name]` command:

```

ps@boston-edge-1> show configuration protocols bgp group ebgp-as-107
type external;
import as-107-in-secondary;
export aggregate-out;
peer-as 107;
neighbor 18.32.16.102;
ps@boston-edge-1> show configuration policy-options policy-statement as-107-in-
secondary
term localpref-50 {
    then {
        local-preference 50;
    }
}

```

When you review your peer configuration and applied import policy, you can see why you’re having problems. The local-preference on all routes from our Boston service provider is set to 50, causing the Chicago service provider to act as the preferred egress point for all traffic destined for the Internet. Local-preference is the most significant criteria in the BGP route selection process. The higher the value, the more preferred the route is. The default value is 100, so a setting of 50 would make any route learned from this BGP peer less preferred than the same route learned from the primary provider, whose routes are getting the default local-preference.

Let’s confirm. Using the `show route` command for an Internet destination for Boston:

```

ps@boston-edge-1> show route 8.32.80.0/23
inet.0: 7013 destinations, 14007 routes (7013 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
8.32.80.0/23    *[BGP/170] 00:10:13, localpref 100, from 10.25.30.1
                AS path: 44 107 46355 8081 52274 22469 5890 9532 8078 I
                > to 192.168.14.1 via ge-1/0/0.0
                [BGP/170] 00:05:03, localpref 50

```

```
AS path: 107 44 46355 8081 52274 22469 5890 9532 8078 I
> to 18.32.16.102 via so-3/3/2.0
```

This output confirms our hypothesis. The asterisk in the output indicates the selected route, and you can see that the selected route is the route learned through our Chicago peering point, which can be quickly identified because the first hop in the AS-Path is 44, the service-provider in Chicago. It's selected because of the value configured for the local-preference. To allow some traffic to prefer the Boston egress point, you need to update your policy to match on some routes and set them to a higher local-preference than Chicago. There are no per-prefix traffic statistics, so you should modify your policy, check your interface statistics, and then repeat the cycle and tweak it until you are happy with the traffic levels.

Aim for reducing the egress utilization in Chicago to 70%, which should mean a drop of 300 to 400 megabits/second. An easy way to begin is to configure a local-preference for the Boston service provider for routes that originate from their AS or from their customers.

First, you know your provider sets a BGP community of 107:100 on all customer routes as they document this in their peering policies, which are posted on their website, a common method for network operators to distribute this information; so use this information to develop the policy. Next, let's add a new term which matches on this community, and set the local-preference to 120. That should force the Boston peering point to become preferred for those routes. Let's use 8.32.80.0/23 as an example to see if your change had the desired effect:

```
ps@boston-edge-1> show route 8.32.80.0/23 detail
inet.0: 7013 destinations, 14004 routes (7013 active, 0 holddown, 0 hidden)
8.32.80.0/23 (2 entries, 1 announced)
  *BGP   Preference: 170/-101
    Next hop type: Indirect
    Next-hop reference count: 20970
    Source: 10.25.30.1
    Next hop type: Router, Next hop index: 488
    Next hop: 192.168.14.1 via ge-0/01.0, selected
    Protocol next hop: 10.25.30.1
    Indirect next hop: 8e04000 131070
    State: <Active Int Ext>
    Local AS: 10 Peer AS: 10
    Age: 1 Metric2: 1
    Task: BGP_10.10.25.30.1+179
    Announcement bits (2): 0-KRT 4-Resolve tree 1
    AS path: 44 107 I
    Communities: 107:100
    Localpref: 100
```



```

Router ID: 10.25.30.1
BGP   Preference: 170/-51
      Next hop type: Router, Next hop index: 486
      Next-hop reference count: 6999
      Source: 18.32.16.102
      Next hop: 18.32.16.102 via so-3/3/2.0, selected
      State: <Ext>
      Inactive reason: Local Preference
      Local AS: 10 Peer AS: 107
      Age: 3:57
      Task: BGP_107.18.32.16.102+59002
      AS path: 107 I
      Communities: 107:100
Localpref: 50
      Router ID: 172.17.0.3

```

Before the change, we prefer the Chicago exit point and we see that this is due to local-preference.

To make the change, create a community named `as-107-customers`, which includes `107:100`, and use that community in a newly inserted term. Our final configuration appears as follows:

```

ps@boston-edge-1> show configuration policy-options
policy-statement as-107-in-secondary {
  term localpref-50 {
    then {
      local-preference 50;
    }
  }
  term localpref-120 {
    from community as-107-customers;
    then {
      local-preference 120;
    }
  }
}
community as-107-customers members 107:100;

```

To confirm the change has had the desired effect, once again issue a `show route` command on the example prefix, `8.32.80.0/23`:

```

ps@boston-edge-1> show route 8.32.80.0/23 detail
inet.0: 7013 destinations, 13008 routes (7013 active, 0 holddown, 0 hidden)
8.32.80.0/23 (1 entry, 1 announced)
  *BGP   Preference: 170/-121
        Next hop type: Router, Next hop index: 486
        Next-hop reference count: 8991
        Source: 18.32.16.102
Next hop: 18.32.16.102 via so-3/3/2.0, selected
        State: <Active Ext>

```

```

Local AS: 10 Peer AS: 107
Age: 35:27
Task: BGP_107.18.32.16.102+59002
Announcement bits (3): 0-KRT 3-BGP RT Background 4-Resolve tree 1
AS path: 107 I
Communities: 107:100
Localpref: 120
Router ID: 172.17.0.3

```

You can also issue the `show bgp summary` command to verify that the Boston router is now selecting AS 107 for some prefixes. Before, this command showed 0/6993/0, but now you see that 1256 routes are active from this peer (and 1256 less routes are active from the Chicago peer).

```

ps@boston-edge-1> show bgp summary
Groups: 2 Peers: 3 Down peers: 0
Table      Tot Paths  Act Paths Suppressed  History Damp State  Pending
inet.0      13986      6993      0          0        0        0
Peer       AS      InPkt    OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Damped...
10.25.30.1   10      7013     7112     0       0      13:07 5737/6993/0
0/0/0
10.25.30.3   10       28     7111     0       0      12:41 0/0/0      0/0/0
18.32.16.102 107    7006     6277     0       0      8:10 1256/6993/0    0/0/0

```

Perfect! Now routes with the community 107:100 are transiting through the Boston SONET link (see the bolded line above). You can also confirm that routes not matching this community continue to prefer the Chicago link, and use ISP-1's aggregate route 178.0.0.0/8 to validate this preference, by looking at the route on the Boston router:

```

ps@boston-edge-1> show route 178.63.18.22 detail
inet.0: 7013 destinations, 13008 routes (7013 active, 0 holddown, 0 hidden)
178.0.0.0/8 (2 entries, 1 announced)
  *BGP Preference: 170/-101
    Next hop type: Indirect
    Next-hop reference count: 17982
    Source: 10.25.30.1
    Next hop type: Router, Next hop index: 488
    Next hop: 192.168.14.1 via ge-0/0/0.0, selected
    Protocol next hop: 10.25.30.1
    Indirect next hop: 8e04000 131070
    State: <Active Int Ext>
    Local AS: 10 Peer AS: 10
    Age: 1:23:37 Metric2: 1
    Task: BGP_10.10.25.30.1+179
    Announcement bits (2): 0-KRT 4-Resolve tree 1
    AS path: 44 I

```

```

Localpref: 100
Router ID: 10.25.30.1
BGP   Preference: 170/-51
      Next hop type: Router, Next hop index: 486
      Next-hop reference count: 8991
      Source: 18.32.16.102
      Next hop: 18.32.16.102 via so-3/3/2.0, selected
      State: <Ext>
      Inactive reason: Local Preference
      Local AS: 10 Peer AS: 107
      Age: 1:20:22
      Task: BGP_107.18.32.16.102+59002
      AS path: 107 44 I
      Localpref: 50
      Router ID: 172.17.0.3

```

The configuration is working as expected. Boston prefers the Gigabit Ethernet connection to the Chicago site for this destination. For the real test, you have to now check traffic levels to see how much traffic has been moved to the alternate route by checking the bit rates on the interfaces:

```

ps@boston-edge-1> show interfaces so-3/3/2
Physical interface: so-3/3/2, Enabled, Physical link is Up
  Interface index: 167, SNMP ifIndex: 151
  Description: Connection to isp-2
  Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode, Speed: OC12,
  Loopback: None, FCS: 16, Payload scrambler: Enabled
  Device flags   : Present Running
  Interface flags: Point-To-Point SNMP-Traps Internal: 0x4000
  Link flags     : Keepalives
  Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
  Keepalive: Input: 0 (never), Output: 0 (never)
  LCP state: Down
  NCP state: inet: Not-configured, inet6: Not-configured, iso: Not-configured, mpls:
  Not-configured
  CHAP state: Closed
  PAP state: Closed
  CoS queues   : 8 supported, 4 maximum usable queues
  Last flapped : 2009-10-14 07:03:58 PDT (2d 03:37 ago)
  Input rate   : 1724 bps (9 pps)
  Output rate  : 3063137642 bps (10599092 pps)
  SONET alarms : None
  SONET defects: None

```

If you open a second CLI session to the router, you could use the monitor interface so-3/2/2 command to monitor real-time traffic on boston-edge-1:

```

ps@boston-edge-1> monitor interface so-3/3/2
boston-edge-1                               Seconds: 18                               Time: 16:11:23
                                                Delay: 0/0/17

Interface: so-3/2/2, Enabled, Link is Up
Encapsulation: PPP, Keepalives, Speed: OC12
Traffic statistics:
  Input bytes:                               35514 (1973 bps)                               [184]
  Output bytes:                               55136476428 (3063137579 bps)           [30631491127]
  Input packets:                             162 (9 pps)                               [10]
  Output packets:                            190782746 (10599092 pps)                   [10435138]
Error statistics:
  Input errors:                              0                                           [0]
  Input drops:                               0                                           [0]
  Input framing errors:                      0                                           [0]
  Input runs:                               0                                           [0]
  Input giants:                              0                                           [0]
  Policed discards:                          0                                           [0]
  L3 incompletes:                            0                                           [0]
  L2 channel errors:                         0                                           [0]
  L2 mismatch timeouts:                     0                                           [0]
  Carrier transitions:                       1                                           [0]
  Output errors:                             0                                           [0]
  Output drops:                              0                                           [0]
  Aged packets:                              0                                           [0]
Next='n', Quit='q' or ESC, Freeze='f', Thaw='t', Clear='c', Interface='i'

```

These commands show that the Boston edge router is now sending approximately 300 megabits over its SONET link to the service provider. Running a similar command on the Chicago edge router, you can see the drop in traffic. Let's check on the Gigabit Ethernet again one last time.

```

ps@chicago-edge-1> show interfaces ge-0/0/9
Physical interface: ge-0/0/9, Enabled, Physical link is Up
  Interface index: 137, SNMP ifIndex: 118
  Description: Connection to isp-1
  Link-level type: Ethernet, MTU: 1514, Speed: 1000mbps, MAC-REWRITE Error: None,
  Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled,
  Auto-negotiation: Enabled, Remote fault: Online
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Link flags     : None
  CoS queues     : 8 supported, 8 maximum usable queues
  Current address: 00:19:e2:25:b0:09, Hardware address: 00:19:e2:25:b0:09
  Last flapped   : 2009-10-13 13:51:19 PDT (2d 20:44 ago)
  Input rate     : 3172844311 bps (10682977 pps)
  Output rate    : 6739379368 bps (23159379 pps)  Active alarms : None
  Active defects : None

```

The bolded line shows that the output traffic levels on the Chicago provider connection have dropped to ~670 megabits, which meets the goal of reducing outbound traffic to 70% utilization.

The Fix Test

You successfully engineered your network traffic to resolve the loss users were experiencing. Since this is a short-term fix, you need to give it the short-term fix test to make sure it is acceptable:

Does the Fix Cause Other Problems?

None that are apparent. Using your NMS system, you should monitor the Boston circuit daily to ensure it does not get over-utilized at peak traffic times.

Does the Fix Survive a Reboot?

Yes.

Is the Fix Well Communicated?

That is up to you. Under normal circumstances, you would update tickets tracking this issue and send an email to operational personnel providing details on the fix, what to monitor, and when to escalate any related issues.

Is the Fix Operationally Understandable?

The fix is understandable. You are using a service-provider supplied community to engineer some traffic to prefer your Boston peering point.

Will the Fix be Replaced with a Long-Term Fix in a Reasonable Amount of Time?

Unfortunately, this may not really be up to you. Your service provider must coordinate the upgrade and the local loop provider may need to install a new circuit. While this may or may not be completed in a reasonable amount of time, you were in an outage scenario and traffic engineering was the most appropriate short-term fix.

Summary

While your specific traffic engineering problems and issues will always be different than this chapter's example, the purpose was to illustrate a network outage and show how to apply The Fix Test to it. Traffic engineering is always a good example to showcase because when it goes sour everyone knows. A simple set of rules will help in your approach to and effectiveness with troubleshooting.

Listen to your network users, but factor in their emotions. Minor or localized network problems may appear worse to some users, depending on the impact.

Apply a set formula for examining a problem. A consistent approach yields consistent results. An example of such a formula is:

What is the scope of the problem?

How many distinct source networks are affected?

What destinations are involved?

Who reported the problem first?

What type(s) of traffic is affected?

Is the problem constant or sporadic?

Test your theories and always confirm from another source. A combination of instrumentation and practical tests should prove that your fix worked.

Short-term fixes lead to long-term resolutions. Your primary objective is to restore service in an operationally supportable way and often this involves short-term fixes.

Allow yourself to improve your formula as you go. If you consistently use an evolving formula, your results should always improve.

Chapter 3

CLI Instrumentation

<i>Environmental Commands</i>	<i>30</i>
<i>Chassis Commands</i>	<i>32</i>
<i>Request Support Information</i>	<i>37</i>
<i>Summary</i>	<i>38</i>

To keep your network stable and maintain a consistently high uptime, you are going to need to be able to troubleshoot problems as they arise. The most essential skill when it comes to troubleshooting Junos devices is your ability to understand and operate the Junos CLI.

Let's begin by examining some of the helpful system instrumentation commands that allow you to verify that the system is functioning properly and to help you to begin troubleshooting when things are not working.

Environmental Commands

Most of the environmental instrumentation commands can be found at the chassis hierarchy level:

```
ps@dunkel-re0> show chassis ?
```

Possible completions:

alarms	Show alarm status
craft-interface	Show craft interface status
environment	Show component status and temperature, cooling system speeds
ethernet-switch	Show Ethernet switch information
fabric	Show internal fabric management state
firmware	Show firmware and operating system version for components
fpc	Show Flexible PIC Concentrator status
hardware	Show installed hardware components
location	Show physical location of chassis
mac-addresses	Show media access control addresses
pic	Show Physical Interface Card state, type, and uptime
routing-engine	Show Routing Engine status
sibs	Show Switch Interface Board status
synchronization	Show clock synchronization information
temperature-thresholds	Show chassis temperature threshold settings

A quick way to assess the status of the chassis is to issue the show chassis alarms command:

```
ps@dunkel-re0> show chassis alarms
```

1 alarms currently active

Alarm time	Class	Description
2010-01-19 11:47:35 PST	Major	PEM 3 Not OK

Here our chassis seems to be having a problem with power entry module (PEM) number 3. This usually indicates a power source problem. Either the cable is unplugged or there is a problem with the circuit breaker. To better understand the problem, issue the show chassis environment pem command:


```
ps@dunkel-re0> show chassis environment pem
```

```

PEM 0 status:
  State           Online
  Temperature      OK
  DC Output:       OK
PEM 1 status:
  State           Online
  Temperature      OK
  DC Output:       OK
PEM 2 status:
  State           Online
  Temperature      OK
  DC Output:       OK
PEM 3 status:
  State           Present

```

PEM 3 is displayed as Present, leading us to believe that the power input is the source of the problem. And here you should implement the decision tree in Figure 1-1, maybe starting with having an onsite technician confirm that the cable and circuit breaker are functioning properly.

Let's check with the `show chassis environment` command to display all the current environmental data for the device:

```
ps@dunkel-re0> show chassis environment
```

Class	Item	Status	Measurement
Temp	PEM 0	OK	
	PEM 1	OK	
	PEM 2	OK	
	PEM 3	Check	
	Routing Engine 0OK	45 degrees C / 113 degrees F	
	Routing Engine 1	OK	43 degrees C / 109 degrees F
	CB 0	OK	41 degrees C / 105 degrees F
	CB 1	OK	39 degrees C / 102 degrees F
	SIB 0	OK	41 degrees C / 105 degrees F
	SIB 1	OK	41 degrees C / 105 degrees F
	SIB 2	OK	41 degrees C / 105 degrees F
	SIB 3	OK	44 degrees C / 111 degrees F
	FPC 0 Intake	OK	31 degrees C / 87 degrees F
	FPC 0 ExhaustOK	42 degrees C / 107 degrees F	
	FPC 1 Intake	OK	31 degrees C / 87 degrees F
	FPC 1 Exhaust	OK	41 degrees C / 105 degrees F
	FPC 2 Intake	OK	30 degrees C / 86 degrees F
	FPC 2 Exhaust	OK	41 degrees C / 105 degrees F
	FPC 3 Intake	OK	32 degrees C / 89 degrees F
	FPC 3 Exhaust	OK	43 degrees C / 109 degrees F
	FPC 4 Intake	OK	31 degrees C / 87 degrees F
	FPC 4 Exhaust	OK	42 degrees C / 107 degrees F
	FPM GBUS	OK	33 degrees C / 91 degrees F

```

Fans  Top Left Front fan    OK      Spinning at normal speed
      Top Right Rear fan    OK      Spinning at normal speed
      Top Right Front fan   OK      Spinning at normal speed
      Top Left Rear fan     OK      Spinning at normal speed
      Bottom Left Front fan  OK      Spinning at normal speed
      Bottom Right Rear fan  OK      Spinning at normal speed
      Bottom Right Front fan OK      Spinning at normal speed
      Bottom Left Rear fan   OK      Spinning at normal speed
      Rear Fan 1 (TOP)       OK      Spinning at normal speed
      Rear Fan 2             OK      Spinning at normal speed
      Rear Fan 3             OK      Spinning at normal speed
      Rear Fan 4             OK      Spinning at normal speed
      Rear Fan 5             OK      Spinning at normal speed
      Rear Fan 6             OK      Spinning at normal speed
      Rear Fan 7 (Bottom)    OK      Spinning at normal speed
Misc  CIP                   OK

```

You can see how the `show chassis environment` command provides information on the status of the power entry modules, temperatures, and fan operation. Temperature alarms are also displayed here (and can also be displayed with the `show chassis alarms` command) and are often launched by site problems such as cooling system failures, incorrect rack and system layouts, or fan failures (which would also be shown in the output of the `show chassis environment` and `show chassis alarms` commands).

TIP Fan failures and PEM failures that are not caused by bad cables or circuit breaker problems generally require a Juniper Networks RMA (Return Material Authorization) and should have a ticket opened with the Juniper Technical Assistance Center (JTAC).

Chassis Commands

The other main chassis level concerns include the status of the routing-engine(s), FPCs (Flexible PIC Concentrator), and PICs (Physical Inter-face Card). The `show chassis routing-engine`, `show chassis fpc`, and `show chassis fpc pic-status` commands can display this information:

Here's a sample of the output from the `show chassis routing-engine` command when issued on a router with a single routing engine:

```

ps@doppelbock> show chassis routing-engine
Routing Engine status:
  Temperature      32 degrees C / 89 degrees F
  CPU temperature   32 degrees C / 89 degrees F

```

```

DRAM                    512 MB
Memory utilization      36 percent
CPU utilization:
  User                  2 percent
  Background            0 percent
  Kernel                4 percent
  Interrupt             0 percent
  Idle                  94 percent
Model                   RE-2.0
Serial ID               c40000078cf97701
Start time              2010-01-12 05:56:58 EST
Uptime                  7 days, 21 hours, 4 seconds
Load averages:         1 minute  5 minute  15 minute
                       0.08      0.02     0.01

```

You can see valuable information in this output. The operating temperature, CPU utilization, and uptime are all important. Network management suites and the routers themselves alarm for threshold breaches of some of these values (temperature, for example) and events related to others, but understanding this output on the CLI is also valuable when troubleshooting and diagnosing the overall health of a device.

And here's the `show chassis routing-engine` command when issued on a router with a dual routing engines. Note that the current state of slot 0 is master and slot 1 is backup, these are the default mastership priorities.

```

ps@dunkel-re0> show chassis routing-engine
Routing Engine status:
Slot 0:
  Current state          Master
  Election priority      Master
  Temperature            45 degrees C / 113 degrees F
  CPU temperature        51 degrees C / 123 degrees F
  DRAM                   3584 MB
  Memory utilization     9 percent
  CPU utilization:
    User                 0 percent
    Background           0 percent
    Kernel               3 percent
    Interrupt            0 percent
    Idle                 97 percent
  Model                  RE-A-2000
  Serial ID              1000702757
  Start time             2010-01-19 11:42:50 PST
  Uptime                 23 hours, 26 minutes, 46 seconds
  Load averages:        1 minute  5 minute  15 minute
                       0.00      0.04     0.05

```

Routing Engine status:

Slot 1:

Current state	Backup
Election priority	Backup
Temperature	43 degrees C / 109 degrees F
CPU temperature	47 degrees C / 116 degrees F
DRAM	3584 MB
Memory utilization	8 percent
CPU utilization:	
User	0 percent
Background	0 percent
Kernel	0 percent
Interrupt	0 percent
Idle	100 percent
Model	RE-A-2000
Serial ID	1000699981
Start time	2010-01-19 11:28:08 PST
Uptime	23 hours, 41 minutes, 28 seconds

The output shown here for a dual routing engine equipped router is what you would expect when it is operating normally. If the Current state for the routing engine is displayed as **present** (as shown below), you might need to investigate it further:

ps@dunkel-re0> **show chassis routing-engine**

Routing Engine status:

Slot 0:

Current state	Master
Election priority	Master
Temperature	45 degrees C / 113 degrees F
CPU temperature	52 degrees C / 125 degrees F
DRAM	3584 MB
Memory utilization	9 percent
CPU utilization:	
User	0 percent
Background	0 percent
Kernel	3 percent
Interrupt	1 percent
Idle	97 percent
Model	RE-A-2000
Serial ID	1000702757
Start time	2010-01-19 11:42:50 PST
Uptime	1 day, 39 minutes, 22 seconds
Load averages:	1 minute 5 minute 15 minute
	0.01 0.01 0.02

Routing Engine status:

Slot 1:

Current state	Present
---------------	---------

A routing engine in the Present state indicates that the routing-engine is installed in the router, but is not functioning properly. It might be because the other routing engine is performing a normal boot-up. However, if this state persists, or the routing engine does not show up at all, troubleshooting is likely necessary.

If possible, attempt to connect to the device on the console port. If the console port is not responding, have the onsite technician remove and reseal the routing engine while you monitor the console port. If there is still no output on the screen, or if there is output but the routing engine fails to boot, open a case with JTAC for further troubleshooting or to set up an RMA.

The `show chassis fpc` and `show chassis fpc pic-status` commands provide information on the current states of the installed FPCs and PICs:

```
ps@dunkel-re0> show chassis fpc
```

Slot	State	Temp (C)	CPU Utilization (%)	Interrupt	Memory DRAM (MB)	Utilization (%)	Heap	Buffer
0	Online	42	2	0	1024	2	49	
1	Online	40	1	0	1024	2	49	
2	Online	41	1	0	1024	2	49	
3	Online	43	1	0	1024	2	50	
4	Online	42	1	0	1024	2	49	
5	Empty							
6	Empty							
7	Empty							

```
ps@dunkel-re0> show chassis fpc pic-status
```

Slot 0	Online	M320 E2-FPC Type 3
PIC 0	Online	10x 1GE(LAN), 1000 BASE
PIC 1	Online	10x 1GE(LAN), 1000 BASE
Slot 1	Online	M320 E2-FPC Type 3
PIC 0	Online	4x OC-48 SONET
PIC 1	Online	8x 1GE(TYPE3), IQ2
Slot 2	Online	M320 E2-FPC Type 2
PIC 0	Online	4x OC-12 SONET, SMIR
PIC 1	Online	2x OC-12 ATM-II IQ, MM
Slot 3	Online	M320 E2-FPC Type 1
PIC 0	Online	1x OC-12 SONET, SMIR
PIC 1	Online	1x OC-12 ATM-II IQ, MM
PIC 2	Online	4x OC-3 SONET, SMIR
PIC 3	Online	4x OC-3 SONET, MM
Slot 4	Online	M320 E2-FPC Type 1
PIC 0	Online	4x CHDS3 IQ
PIC 2	Online	1x CHOC12 IQ SONET, SMIR
PIC 3	Online	4x OC-3 SONET, SMIR

For normal FPCs and PICs, all components should be displayed as **Online**. Any other status might require further investigation. Like routing-engines, FPCs and PICs do need to boot up, so states other than **Online** are acceptable shortly after a reboot of the router, FPC, or PIC.

However, if the non-normal state persists, the first troubleshooting step is to attempt to reboot the PIC or FPC. To reboot an FPC, issue the `request chassis fpc slot [slot-number] restart` command:

```
ps@dunkel-re0> request chassis fpc slot 4 restart
Restart initiated, use "show chassis fpc" to verify
```

Repeatedly running the `show chassis fpc` command allows you to follow the progress of the restarting FPC:

```
ps@dunkel-re0> show chassis fpc
```

Slot	State	Temp (C)	CPU Total	Utilization (%) Interrupt	Memory DRAM (MB)	Utilization (%) Heap	Utilization (%) Buffer
0	Online	42	4	0	1024	2	49
1	Online	40	1	0	1024	2	49
2	Online	41	1	0	1024	2	49
3	Online	43	1	0	1024	2	50
4	Offline	---Restarted by cli command---					
5	Empty						
6	Empty						
7	Empty						

```
ps@dunkel-re0> show chassis fpc
```

Slot	State	Temp (C)	CPU Total	Utilization (%) Interrupt	Memory DRAM (MB)	Utilization (%) Heap	Utilization (%) Buffer
0	Online	42	4	0	1024	2	49
1	Online	40	1	0	1024	2	49
2	Online	41	1	0	1024	2	49
3	Online	43	1	0	1024	2	50
4	Present	42					
5	Empty						
6	Empty						
7	Empty						

```
ps@dunkel-re0> show chassis fpc
```

Slot	State	Temp (C)	CPU Total	Utilization (%) Interrupt	Memory DRAM (MB)	Utilization (%) Heap	Utilization (%) Buffer
0	Online	42	4	0	1024	2	49
1	Online	40	1	0	1024	2	49
2	Online	41	1	0	1024	2	49
3	Online	43	1	0	1024	2	50
4	Online	40	1	0	1024	2	49
5	Empty						
6	Empty						
7	Empty						

To reboot a PIC, first issue the `request chassis pic fpc-slot [slot-number] pic-slot [pic-slot number] offline` command. Check that the PIC is offline by issuing the `show chassis pic fpc-slot [slot-number] pic-slot [pic-slot number]` command. And then bring the PIC back online by issuing the `request chassis pic fpc-slot [slot-number] pic-slot [pic-slot number] online` command:

```
ps@dunkel-re0> request chassis pic fpc-slot 4 pic-slot 0 offline
fpc 4 pic 0 offline initiated, use "show chassis fpc pic-status 4" to verify
ps@dunkel-re0> show chassis pic fpc-slot 4 pic-slot 0
FPC slot 4, PIC slot 0 information:
  State                Offline
ps@dunkel-re0> request chassis pic fpc-slot 4 pic-slot 0 online
fpc 4 pic 0 online initiated, use "show chassis fpc pic-status 4" to verify
ps@dunkel-re0> show chassis pic fpc-slot 4 pic-slot 0
FPC slot 4, PIC slot 0 information:
  Type                4x CHDS3 IQ
  State                Online
  PIC version          2.7
  Uptime                1 second
```

If the problem persists, you might try reseating the FPC or PIC as discussed in the routing engine troubleshooting section. If that does not resolve the problem, open a case with JTAC.

If you need to open a case for a routing engine, FPC, or PIC problem, be sure to include the output from the `request support information` command and a copy of the messages and chassisd log files (stored in the `/var/log` directory) in the new JTAC case.

Request Support Information

The `request support information` command is a batch command that automatically runs a number of different CLI commands. These commands are extremely useful for Juniper's support organizations when troubleshooting any issue. The output of this command can either be saved in a buffer within the telnet or SSH client, or the output can be saved locally on the router and transferred using FTP, SCP, or SFTP.

The following is an example of how to save and gather the output of the `request support information` command using SFTP.

Issue the `request support information` command and redirect the output to a file:

```
ps@dunkel-re0> request support information | save rsi-dunkel-01202010.log
Wrote 7679 lines of output to 'rsi-dunkel-01202010.log'
```

Use SFTP to download the saved file:

```
server-1>sftp ps@dunkel
Connecting to dunkel...
ps@dunkel's password:
sftp> get rsi-dunkel-01202010.log
Fetching /var/home/ps/rsi-dunkel-01202010.log to rsi-dunkel-01202010.log
/var/home/ps/rsi-dunkel-01202010.log          100% 381KB 381.2KB/s 00:00
```

Log Files

It is also quite useful for JTAC to have a copy of the messages and chassisd log files, which exist in the /var/log directory on the router. While an SFTP session is open with the router, you can copy those files as well:

```
sftp> cd /var/log
sftp> get messages
Fetching /var/log/messages to messages
/var/log/messages                          100% 32KB 31.7KB/s 00:00
sftp> get chassisd
Fetching /var/log/chassisd to chassisd
/var/log/chassisd                         100% 1967KB 1.9MB/s 00:01
```

You now have all of the information you need to open a JTAC case. Juniper's support teams can help you with the remaining troubleshooting and, if necessary, create an order for replacement hardware.

Summary

Network management suites provide an excellent method for actively monitoring a network, polling for specific values, and to some degree isolating problems, but nothing can replace the ability to efficiently navigate the CLI. Most real time troubleshooting, diagnosis, and resolution steps involve using the CLI, which makes a solid understanding of the CLI all the more important.

There are entire books about the Junos CLI and its potential to monitor and troubleshoot specific issues. This chapter introduced a few key commands and the rest of this booklet will explore the CLI's ability to examine a device's inner workings.

MORE? Need more on Junos? See the other booklets in this Day One series, *Junos Fundamentals*, at <http://www.juniper.net/dayone>.

Chapter 4

System Monitoring and Troubleshooting

<i>Syslog</i>	40
<i>SNMP Polling</i>	43
<i>SNMP Traps</i>	51
<i>Summary</i>	54

Before this booklet gets you looking at connectivity and protocols, you must first ensure your system is functioning properly. This chapter walks you through some of the basic syslog, SNMP polling, SNMP traps, and CLI instrumentation so you can assess the operation of your system. This chapter also acts as the basis for later monitoring and troubleshooting discussions.

Syslog

Administrators should collect system logs, change-logs, and interactive-commands because it allows them to use such syslog files to troubleshoot and monitor. At the same time, they help to corroborate network events with configuration changes. A syslog file can also provide the syslog server with enough information to appropriately alarm, notify, and identify the root cause of problems where and when possible.

Juniper offers both *local logging*, which writes messages to files in the `/var/log` directory located either on the hard disk drive or the compact flash disk, depending on the platform, and *remote logging*, where syslog messages are sent to a remote syslog server.

BEST PRACTICE To limit the amount of writing done to the local hard drive or compact flash disk, it's recommended that local system logging be configured conservatively – the logging levels can be increased and additional tracing can be enabled whenever troubleshooting is necessary.

Remote system logging can be configured more liberally, but it should be measured by the usefulness of the information being gathered. Too many routine log messages can make it more difficult to find the syslog messages relevant to the actual outage, extending an outage rather than allowing you to quickly resolve the issue.

Also, which syslog messages are gathered should be dependent on whether the system or personnel are responsible for monitoring the logs. If people are responsible for monitoring the logs, it is best to keep the messages file short and log only important messages. The technicians should be alerted when warranted by the situation so they can investigate further.

If the syslog messages are tracked by a network monitoring system, it can be helpful to track more in-depth messages to allow for granular impact identification and accurate root-cause isolation.

Syslog messages are tagged with a severity syslog . The severity hierarchy, from least important to most important, is as follows:

Debug
Info
Notice
Warning
Error
Critical
Alert
Emergency

NOTE Junos supports all syslog severities but debug.

Let's show a default configuration that performs the following three functions:

1. Enables the logging of any messages of level *emergency* or higher to the user (messages are displayed to the console or terminal);
2. Any messages of type authorization with a level of info or higher, *or* any message of type notice or higher, are sent to a file called *messages*, which is stored in */var/log*;
3. And, any interactive-commands issued by users are sent to a file called *interactive-commands*, which is also kept in */var/log*:

```
ps@dunkel-re0> show configuration system syslog
user * {
    any emergency;
}
file messages {
    any notice;
    authorization info;
}
file interactive-commands {
    interactive-commands any;
}
```

The default configuration is a good start, but you'll need to tailor it to meet your needs. Since both technicians and an NMS system use your syslog servers, let's attempt to provide a useful volume of information without flooding the logs.

Let's configure a facility for messages sent to this server – a *facility* is another tag attached to the system logs that allows the syslog server to filter certain messages. The syslog server administrator can use this

facility to filter messages to a file. Below, the router is configured for additional local logging and to send system logs to a syslog server at 172.19.110.10:

```
ps@dunkel-re0> configure
Entering configuration mode
[edit]
ps@dunkel-re0# set system syslog host 172.19.110.10 any notice
[edit]
ps@dunkel-re0# set system syslog host 172.19.110.10 interactive-commands any
[edit]
ps@dunkel-re0# set system syslog host 172.19.110.10 change-log any
[edit]
ps@dunkel-re0# set system syslog host 172.19.110.10 facility-override local3
[edit]
ps@dunkel-re0# show system syslog
user * {
    any emergency;
}
host 172.19.110.10 {
    any notice;
    change-log any;
    interactive-commands any;
    facility-override local3;
}
file messages {
    any notice;
    authorization info;
}
file interactive-commands {
    interactive-commands any;
}
[edit]
ps@dunkel-re0# show | compare
[edit system syslog]
+   host 172.19.110.10 {
+       any notice;
+       change-log any;
+       interactive-commands any;
+       facility-override local3;
+   }
[edit]
ps@dunkel-re0# commit
commit complete
```

The syslog server now receives notice level logs, change-logs, and interactive commands, and they are sent with a facility override of local3.

The syslog server should be receiving all the syslog information you could need to troubleshoot most issues, including Layer 1 problems. But remember our commitment to *confirm* all troubleshooting steps with a second network opinion? Let's run a Fix Test by triggering a link transition and then monitor our syslog servers to see if they receive notice.

Disabling the SONET link on so-3/3/2:

```
ps@dunkel-re0> configure
Entering configuration mode
[edit]
ps@dunkel-re0# set interfaces so-3/3/2 disable
[edit]
ps@dunkel-re0# commit
commit complete
Resulting syslog message:
Jan 19 15:48:41 172.19.110.171 mib2d[4549]: SNMP_TRAP_LINK_DOWN: ifIndex 151,
ifAdminStatus down(2), ifOperStatus down(2), ifName so-3/3/2
```

The syslog message is bolded. And it is indeed working.

Now you have a working, default syslog model. Go ahead and modify this in the future for short-term troubleshooting, but for now, let's save this as our syslog template to enlist other aids.

SNMP Polling

Now that syslog is configured, let's move on to the simple network management protocol (SNMP). SNMP can be used either to passively respond to queries or to actively send SNMP messages, called *traps*.

NOTE SNMP is disabled by default on Juniper Networks routers.

First let's configure the router to allow SNMP polling from a certain address, and an SNMP community of tr4pp15t (an SNMP community is a simple password to authenticate the entity making the query):

```
ps@dunkel-re0> configure
Entering configuration mode
[edit]
ps@dunkel-re0# set snmp community tr4pp15t authorization read-only
[edit]
ps@dunkel-re0# set snmp community tr4pp15t clients 172.19.110.10/32
[edit]
ps@dunkel-re0# show snmp
```

```

community tr4pp15t {
    authorization read-only;
    clients {
        172.19.110.10/32;
    }
}
[edit]
ps@dunkel-re0# show | compare
[edit snmp]
+ community tr4pp15t {
+     authorization read-only;
+     clients {
+         172.19.110.10/32;
+     }
+ }
[edit]
ps@dunkel-re0# commit
commit complete

```

This configuration allows read-only SNMP access from an SNMP server located at 172.19.110.10.

The SNMP server needs to use the `tr4pp15t` community to conduct the SNMP poll of our router and you can test this by sending an SNMP query of `system.sysDescr.0` to the router from 172.19.110.10 (nms-1).

NOTE The `system.sysDescr.0` string is an *object identifier* (OID) that instructs the router on what value to respond with.

In this case, query the `system.sysDescr.0` OID, which provides system information from the Junos device. Junos provides the vendor, platform, kernel, and build information when queried with this string:

```

nms-1> snmpget -v2c -c tr4pp15t dunkel system.sysDescr.0
SNMPv2-MIB::sysDescr.0 = STRING: Juniper Networks, Inc. m320 internet router, kernel
Junos 10.1R1.8 #0: 2010-02-12 17:15:05 UTC    builder@queth.juniper.net:/volume/build/
Junos/10.1/release/10.1R1.8/obj-i386/bsd/sys/compile/JUNIPER Build date: 2010-02-12
16:37:34 UTC Copyright (c) 1996

```

To confirm our configuration, attempt another SNMP poll of the router from a different server:

```

server-1> snmpget -v2c -c tr4pp15t dunkel system.sysDescr.0
Timeout: No Response from dunkel.

```

The query timed-out, which means our router is only listening to SNMP queries from 172.19.110.10.

There are numerous SNMP objects that can be queried for information. Many of these objects are dependent on what services, protocols, and configurations are enabled.

BEST PRACTICE Juniper recommends that you configure the following base objects to monitor system operation:

- Hardware inventory
- Routing engine CPU utilization
- Routing engine memory utilization
- Routing engine temperature

The following sections describe these SNMP objects in greater detail.

Hardware Inventory

To poll the device for its hardware inventory, you can use the `snmpwalk` utility, which walks the SNMP tree starting at the value provided:

The hardware inventory OID is `.1.3.6.1.4.1.2636.3.1.13.1.5`

```
nms-1> snmpwalk -v2c -c tr4pp15t dunkel .1.3.6.1.4.1.2636.3.1.13.1.5
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.1.1.0.0 = STRING: "midplane"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.2.1.0.0 = STRING: "PEM 0"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.2.2.0.0 = STRING: "PEM 1"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.2.3.0.0 = STRING: "PEM 2"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.2.4.0.0 = STRING: "PEM 3"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.1.1.0 = STRING: "Top Left Front fan"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.1.2.0 = STRING: "Top Right Rear fan"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.1.3.0 = STRING: "Top Right Front fan"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.1.4.0 = STRING: "Top Left Rear fan"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.2.1.0 = STRING: "Bottom Left Front fan"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.2.2.0 = STRING: "Bottom Right Rear fan"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.2.3.0 = STRING: "Bottom Right Front fan"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.2.4.0 = STRING: "Bottom Left Rear fan"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.3.1.0 = STRING: "Rear Fan 1 (TOP)"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.3.2.0 = STRING: "Rear Fan 2"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.3.3.0 = STRING: "Rear Fan 3"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.3.4.0 = STRING: "Rear Fan 4"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.3.5.0 = STRING: "Rear Fan 5"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.3.6.0 = STRING: "Rear Fan 6"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.4.3.7.0 = STRING: "Rear Fan 7 (Bottom)"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.7.1.0.0 = STRING: "FPC: M320 E2-FPC Type 3 @
0/*/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.7.2.0.0 = STRING: "FPC: M320 E2-FPC Type 3 @
1/*/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.7.3.0.0 = STRING: "FPC: M320 E2-FPC Type 2 @
```

```

2/*/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.7.4.0.0 = STRING: "FPC: M320 E2-FPC Type 1 @
3/*/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.7.5.0.0 = STRING: "FPC: M320 E2-FPC Type 1 @
4/*/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.1.1.0 = STRING: "PIC: 10x 1GE(LAN), 1000
BASE @ 0/0/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.1.2.0 = STRING: "PIC: 10x 1GE(LAN), 1000
BASE @ 0/1/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.2.1.0 = STRING: "PIC: 4x OC-48 SONET @
1/0/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.2.2.0 = STRING: "PIC: 8x 1GE(TYPE3), IQ2 @
1/1/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.3.1.0 = STRING: "PIC: 4x OC-12 SONET, SMIR @
2/0/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.3.2.0 = STRING: "PIC: 2x OC-12 ATM-II IQ, MM
@ 2/1/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.4.1.0 = STRING: "PIC: 1x OC-12 SONET, SMIR @
3/0/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.4.2.0 = STRING: "PIC: 1x OC-12 ATM-II IQ, MM
@ 3/1/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.4.3.0 = STRING: "PIC: 4x OC-3 SONET, SMIR @
3/2/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.4.4.0 = STRING: "PIC: 4x OC-3 SONET, MM @
3/3/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.5.1.0 = STRING: "PIC: 4x CHDS3 IQ @ 4/0/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.5.3.0 = STRING: "PIC: 1x CHOC12 IQ SONET,
SMIR @ 4/2/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.8.5.4.0 = STRING: "PIC: 4x OC-3 SONET, SMIR @
4/3/*"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.9.1.0.0 = STRING: "Routing Engine 0"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.9.2.0.0 = STRING: "Routing Engine 1"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.10.1.1.0 = STRING: "FPM GBUS"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.10.1.2.0 = STRING: "FPM Display"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.12.1.0.0 = STRING: "CB 0"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.12.2.0.0 = STRING: "CB 1"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.15.1.0.0 = STRING: "SIB 0"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.15.2.0.0 = STRING: "SIB 1"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.15.3.0.0 = STRING: "SIB 2"
SNMPv2-SMI::enterprises.2636.3.1.13.1.5.15.4.0.0 = STRING: "SIB 3"

```

To find environmental-related values, the operator must take the last four octets of the hardware entry for a given component, and concatenate that onto the objects in the next sections. (The final four octets for the routing-engine are boldfaced.)

Routing Engine CPU Utilization

Take the information you've learned from your inventory query to construct a new query to obtain the CPU utilization of the two routing engines installed in the chassis. To accomplish this, add 9.1.0.0 and 9.2.0.0 to your CPU utilization OID:

CPU Utilization OID: .1.3.6.1.4.1.2636.3.1.13.1.8.x.x.x.x

Examples:

```
nms-1> snmpwalk -v2c -c tr4pp15t dunkel .1.3.6.1.4.1.2636.3.1.13.1.8.9.1.0.0
SNMPv2-SMI::enterprises.2636.3.1.13.1.8.9.1.0.0 = Gauge32: 3
nms-1> snmpwalk -v2c -c tr4pp15t dunkel .1.3.6.1.4.1.2636.3.1.13.1.8.9.2.0.0
SNMPv2-SMI::enterprises.2636.3.1.13.1.8.9.2.0.0 = Gauge32: 0
```

This shows that the CPU utilization is 3% on routing engine 0 and 0% on routing-engine 1, which makes sense since routing-engine 0 is the primary RE and routing-engine 1 is the backup.

Since we always try to confirm these values from another source, let's issue the `show chassis routing-engine` command on the router:

```
ps@dunkel-re0> show chassis routing-engine
```

Routing Engine status:

Slot 0:

Current state	Master
Election priority	Master
Temperature	45 degrees C / 113 degrees F
CPU temperature	52 degrees C / 125 degrees F
DRAM	3584 MB
Memory utilization	9 percent
CPU utilization:	
User	0 percent
Background	0 percent
Kernel	3 percent
Interrupt	0 percent
Idle	97 percent
Model	RE-A-2000
Serial ID	1000702757
Start time	2010-01-19 11:42:50 PST
Uptime	21 hours, 50 minutes, 15 seconds
Load averages:	1 minute 5 minute 15 minute
	0.00 0.03 0.02

Routing Engine status:

Slot 1:

Current state	Backup
Election priority	Backup
Temperature	43 degrees C / 109 degrees F
CPU temperature	47 degrees C / 116 degrees F

```

DRAM                3584 MB
Memory utilization   8 percent
CPU utilization:
  User               0 percent
  Background         0 percent
  Kernel             0 percent
  Interrupt          0 percent
  Idle               100 percent
Model               RE-A-2000
Serial ID            1000699981
Start time           2010-01-19 11:28:08 PST
Uptime              22 hours, 4 minutes, 52 seconds

```

As shown, the idle values match the utilization values. Note that the utilization value and idle value should correlate, adding up to 100%:

Routing-engine 0 - 3% + 97% = 100%

Routing-engine 1 - 0% + 100% = 100%

Routing Engine Memory Utilization

You can also use SNMP to monitor memory utilization, using the same process of combining the memory utilization OID with the hardware index found in the inventory query executed previously:

Memory Utilization OID: .1.3.6.1.4.1.2636.3.1.13.1.11.x.x.x.x

Warning level: 70%

Examples:

```

nms1> snmpwalk -v2c -c tr4pp15t dunkel .1.3.6.1.4.1.2636.3.1.13.1.11.9.1.0.0
SNMPv2-SMI::enterprises.2636.3.1.13.1.11.9.1.0.0 = Gauge32: 9
nms-1> snmpwalk -v2c -c tr4pp15t dunkel .1.3.6.1.4.1.2636.3.1.13.1.11.9.2.0.0
SNMPv2-SMI::enterprises.2636.3.1.13.1.11.9.2.0.0 = Gauge32: 8

```

This indicates that routing-engine 0 is running at 9% memory utilization while routing-engine 1 is running at 8% memory utilization.

Again, we always confirm, so let's run the `show chassis routing-engine` command on the router:

```
ps@dunkel-re0> show chassis routing-engine
```

Routing Engine status:

Slot 0:

```

Current state       Master
Election priority    Master
Temperature          45 degrees C / 113 degrees F
CPU temperature      52 degrees C / 125 degrees F
DRAM                3584 MB

```

```

Memory utilization          9 percent
CPU utilization:
  User                      0 percent
  Background                0 percent
  Kernel                   3 percent
  Interrupt                 0 percent
  Idle                     97 percent
Model                      RE-A-2000
Serial ID                  1000702757
Start time                 2010-01-19 11:42:50 PST
Uptime                     21 hours, 50 minutes, 15 seconds
Load averages:            1 minute  5 minute  15 minute
                          0.00      0.03     0.02

Routing Engine status:
Slot 1:
  Current state             Backup
  Election priority         Backup
  Temperature               43 degrees C / 109 degrees F
  CPU temperature           47 degrees C / 116 degrees F
  DRAM                     3584 MB
Memory utilization          8 percent
CPU utilization:
  User                      0 percent
  Background                0 percent
  Kernel                   0 percent
  Interrupt                 0 percent
  Idle                     100 percent
Model                      RE-A-2000
Serial ID                  1000699981
Start time                 2010-01-19 11:28:08 PST
Uptime                     22 hours, 4 minutes, 52 seconds

```

And our confirm test shows the correct memory utilization.

Routing Engine Temperature

The next SNMP object that can be queried for information is routing engine temperature. Juniper Networks routers power off automatically to protect themselves if a high-level red temperature alarm has been declared, but it is also valuable to poll the router for this information to identify problems before they cause impact. Different platforms and components have different thresholds. The `show chassis temperature-thresholds` command displays these thresholds for these different components.

Here is sample output from the `show chassis temperature-thresholds` command showing a dual routing-engine chassis with several FPCs and FEBs (Forwarding Engine Boards) installed:

```
ps@dunkel-re0> show chassis temperature-thresholds
```

Item	Fan speed		Yellow alarm		Red alarm	
	Normal	High	Normal	Bad fan	Normal	Bad fan
Chassis default	48	54	65	55	75	65
Routing Engine 0	70	80	95	95	110	110
Routing Engine 1	70	80	95	95	110	110
FPC 0	55	60	75	65	90	80
FPC 1	55	60	75	65	90	80
FPC 2	48	54	70	60	80	70
FPC 3	48	54	70	60	80	70
FPC 4	48	54	70	60	80	70
FPC 5	48	54	70	60	80	70
FEB 0	48	54	70	60	80	72
FEB 1	48	54	70	60	80	72
FEB 2	48	54	70	60	80	72
FEB 3	48	54	70	60	80	72
FEB 4	48	54	70	60	80	72
FEB 5	48	54	70	60	80	80

The SNMP object used to query for component temperatures is: `.1.3.6.1.4.1.2636.3.1.13.1.7.x.x.x.x` and this uses the same concatenation process shown before.

And here are a few examples, starting on routing-engine 0 (9.1.0.0):

```
nms1> snmpwalk -v2c -c tr4pp15t dunkel .1.3.6.1.4.1.2636.3.1.13.1.7.9.1.0.0
SNMPv2-SMI::enterprises.2636.3.1.13.1.7.9.1.0.0 = Gauge32: 45
```

Now on routing-engine 1 (9.2.0.0):

```
nms-1> snmpwalk -v2c -c tr4pp15t dunkel .1.3.6.1.4.1.2636.3.1.13.1.7.9.2.0.0
SNMPv2-SMI::enterprises.2636.3.1.13.1.7.9.2.0.0 = Gauge32: 43
```

And let's confirm our SNMP values using the `show chassis routing-engine` command:

```
ps@dunkel-re0> show chassis routing-engine
Routing Engine status:
Slot 0:
  Current state           Master
  Election priority       Master
  Temperature             45 degrees C / 113 degrees F
  CPU temperature         52 degrees C / 125 degrees F
  DRAM                    3584 MB
  Memory utilization      9 percent
  CPU utilization:
```

```

User                0 percent
Background          0 percent
Kernel             2 percent
Interrupt           0 percent
Idle               97 percent
Model              RE-A-2000
Serial ID          1000702757
Start time         2010-01-19 11:42:50 PST
Uptime             22 hours, 2 minutes, 35 seconds
Load averages:     1 minute  5 minute 15 minute
                   0.00      0.00    0.00

Routing Engine status:
Slot 1:
  Current state      Backup
  Election priority  Backup
  Temperature       43 degrees C / 109 degrees F
  CPU temperature    47 degrees C / 116 degrees F
  DRAM              3584 MB
  Memory utilization 8 percent
  CPU utilization:
    User            0 percent
    Background      0 percent
    Kernel          0 percent
    Interrupt       0 percent
    Idle            100 percent
  Model            RE-A-2000
  Serial ID        1000699981
  Start time       2010-01-19 11:28:08 PST
  Uptime           22 hours, 17 minutes, 14 seconds

```

The temperature values returned from the SNMP query shown here matches our CLI output.

SNMP traps

In addition to polling, SNMP can also actively send traps under certain circumstances. This is extremely important to any NMS system, as it allows the network devices to proactively tell a monitoring system that there is an issue or that an event has occurred. Properly configured SNMP trapping, in conjunction with syslog messages, should provide an NMS system with all that it needs to effectively monitor a network.

An SNMP trap configuration is similar to the SNMP polling configuration implemented in the previous step, except that an additional SNMP community must be configured and an SNMP server must also be

added. Let's begin with an SNMP trap to 172.19.110.10 with a community of tr4pp15t; you may use the same community for polling as well as traps.

You must also specify the types of traps you wish to receive, so here's how to configure SNMP with these additional settings:

```
ps@dunkel-re0> configure
Entering configuration mode
[edit]
ps@dunkel-re0# set snmp trap-group tr4pp15t targets 172.19.110.10
[edit]
ps@dunkel-re0# set snmp trap-group tr4pp15t categories ?
Possible completions:
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
  authentication        Authentication failures
  chassis               Chassis or environment notifications
  configuration         Configuration notifications
  link                  Link up-down transitions
  remote-operations     Remote operations
  rmon-alarm            RMON rising and falling alarms
  routing               Routing protocol notifications
  services              Services notifications
> sonet-alarms          SONET alarm trap subcategories
  startup               System warm and cold starts
  vrrp-events           VRRP notifications
```

Okay, let's configure traps for the authentication, chassis, configuration, link, routing, sonet-alarms, and startup categories:

```
[edit]
ps@dunkel-re0# set snmp trap-group tr4pp15t categories authentication
[edit]
ps@dunkel-re0# set snmp trap-group tr4pp15t categories chassis
[edit]
ps@dunkel-re0# set snmp trap-group tr4pp15t categories configuration
[edit]
ps@dunkel-re0# set snmp trap-group tr4pp15t categories link
[edit]
ps@dunkel-re0# set snmp trap-group tr4pp15t categories routing
[edit]
ps@dunkel-re0# set snmp trap-group tr4pp15t categories sonet-alarms
[edit]
ps@dunkel-re0# set snmp trap-group tr4pp15t categories startup
```

Issue the `show snmp` command (in configuration mode) to display the SNMP configuration with the added changes:

```
[edit]
ps@dunkel-re0# show snmp
community tr4pp15t {
    authorization read-only;
    clients {
        172.19.110.10/32;
    }
}
trap-group tr4pp15t {
    categories {
        authentication;
        chassis;
        link;
        routing;
        startup;
        configuration;
        sonet-alarms;
    }
    targets {
        172.19.110.10;
    }
}
```

As shown, the SNMP configuration contains the allowed poller, 172.19.110.10, which is granted read-only access and must use the tr4pp15t community to poll and a trap-group for SNMP traps, sending SNMP traps of several different categories to 172.10.110.10, also using the tr4pp15t community.

Issue the `show | compare` command to display the changes you have made to the configuration (in this case to the SNMP configuration). The “-” indicates deletions and the “+” indicates additions:

```
[edit]
ps@dunkel-re0# show | compare
[edit snmp]
+ trap-group tr4pp15t {
+     categories {
+         authentication;
+         chassis;
+         link;
+         routing;
+         startup;
+         configuration;
+         sonet-alarms;
+     }
+     targets {
+         172.19.110.10;
+     }
+ }
```

And commit the configuration:

```
[edit]
ps@dunkel-re0# commit
commit complete
```

It's time to validate and confirm our configuration by causing a link transition. This time, disable the SONET link on so-3/3/2 to see what happens:

```
ps@dunkel-re0> configure
Entering configuration mode
[edit]
ps@dunkel-re0# set interfaces so-3/3/2 disable
[edit]
ps@dunkel-re0# commit
commit complete
```

Resulting SNMP trap:

```
172.19.110.171: Link Down Trap (0) Uptime: 22:24:50.09, IF-MIB::ifIndex.154 = INTEGER:
154, IF-MIB::ifAdminStatus.154 = INTEGER: down(2), IF-MIB::ifOperStatus.154 = INTEGER:
down(2), IF-MIB::ifName.154 = STRING: so-3/3/2
```

This triggers the SNMP trap as expected.

Summary

Half of running a well operated network is implementing a network management system and providing that NMS system with the logs and traps it needs to effectively identify and report problems. This is the proactive portion of monitoring and troubleshooting and is an invaluable asset for any well-run network. The complement to an NMS, as you will see in the following chapters, is active troubleshooting (and monitoring) through the Junos CLI.

Chapter 5

Layer 1 and Layer 2 Monitoring and Troubleshooting

<i>Important Interface Commands</i>	<i>56</i>
<i>Layer 1 Monitoring and Troubleshooting</i>	<i>56</i>
<i>SONET Alarms and Defects.....</i>	<i>61</i>
<i>Layer 2 Monitoring and Troubleshooting.....</i>	<i>65</i>
<i>Summary</i>	<i>72</i>

Using the paradigm of the OSI model is instrumental in monitoring and troubleshooting a network because the root-causes of so many network problems are found at Layer 1 and Layer 2, and frankly, too often after spending unnecessary time troubleshooting the higher layers. So put into practice a system that does not ignore first troubleshooting network problems at Layer 1 and Layer 2. As you investigate Layer 1 and Layer 2, monitoring and troubleshooting here and elsewhere, you might notice a couple of repeating tenets – work with what the technology gives you and understand the role your routers and switches play at these layers.

ALERT! For the remainder of the book, any monitoring and troubleshooting refers to syslogs and SNMP traps (as discussed in Chapters 3 and 4) and it is assumed that your configuration approximates the syslog and SNMP configurations.

Juniper Networks routers support many Layer 1 and Layer 2 technologies including SONET, ISDN, Ethernet, and PPP. Junos offers many system logging, SNMP, and instrumentation features to monitor the operation of these protocols. Syslog and SNMP provide useful, high-level information on Layer 1 and Layer 2 issues such as link-down and link-up. Additional monitoring requires tracing configuration and the use of some of the command line instrumentation.

Important Interface Commands

The most important CLI commands you can use when monitoring an interface are `show interface extensive` and `monitor interface`.

The `show interface extensive` command contains all the information for both the physical (Layers 1 and 2) interface and any configured logical (Layers 2 and 3) interfaces.

Layer 1 Monitoring and Troubleshooting

Layer 1 is a foundational layer in the OSI model that defines the methods in which network elements send and receive raw data over a medium. SONET/SDH and Ethernet are examples of well-known Layer 1 protocols. SONET is used in the following examples, but the methods shown are applicable to any Layer 1 technology.

The following is sample output from the `show interface extensive` command issued for a SONET interface. The sections of interest are in **bold** and explained following the output:

```
ps@dunkel-re0> show interfaces so-2/0/0 extensive
Physical interface: so-2/0/0, Enabled, Physical link is Up
Interface index: 162, SNMP ifIndex: 154, Generation: 163
Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode, Speed: OC12,
Loopback: None, FCS: 16, Payload scrambler: Enabled
Device flags   : Present Running
Interface flags: Point-To-Point SNMP-Traps Internal: 0x4000
Link flags     : Keepalives
Hold-times     : Up 0 ms, Down 0 ms
Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
Keepalive statistics:
  Input : 0 (last seen: never)
  Output: 0 (last sent: never)
LCP state: Opened
NCP state: inet: Opened, inet6: Not-configured, iso: Not-configured, mpls:
Not-configured
CHAP state: Closed
PAP state: Closed
CoS queues   : 8 supported, 8 maximum usable queues
Last flapped : 2010-01-19 11:49:06 PST (1d 02:43 ago)
Statistics last cleared: 2010-01-20 14:32:52 PST (00:00:02 ago)
Traffic statistics:
  Input bytes :           192           760 bps
  Output bytes :           89           808 bps
  Input packets:           4           1 pps
  Output packets:          1           1 pps
IPv6 transit statistics:
  Input bytes :           0
  Output bytes :           0
  Input packets:           0
  Output packets:          0
Input errors:
Errors: 0, Drops: 0, Framing errors: 0, Runts: 0, Giants: 0, Bucket drops: 0,
Policed discards: 0, L3 incompletes: 0, L2 channel errors: 0, L2 mismatch timeouts: 0,
HS link CRC errors: 0, HS link FIFO overflows: 0
Output errors:
  Carrier transitions: 0, Errors: 0, Drops: 0, Aged packets: 0,
  HS link FIFO underflows: 0, MTU errors: 0
Egress queues: 8 supported, 4 in use
Queue counters:
  Queued packets  Transmitted packets  Dropped packets
0 best-effort      2              2              0
1 expedited-fo     0              0              0
2 assured-forw     0              0              0
3 network-cont     2              2              0
SONET alarms : None
SONET defects : None
```

```

SONET PHY:           Seconds      Count  State
  PLL Lock           0            0 OK
  PHY Light          0            0 OK
SONET section:
  BIP-B1             0            0
  SEF                 0            0 OK
  LOS                 0            0 OK
  LOF                 0            0 OK
  ES-S                0
  SES-S              0
  SEFS-S              0
SONET line:
  BIP-B2             0            0
  REI-L              0            0
  RDI-L              0            0 OK
  AIS-L              0            0 OK
  BERR-SF            0            0 OK
  BERR-SD            0            0 OK
  ES-L               0
  SES-L              0
  UAS-L              0
  ES-LFE             0
  SES-LFE            0
  UAS-LFE            0
SONET path:
  BIP-B3             0            0
  REI-P              0            0
  LOP-P              0            0 OK
  AIS-P              0            0 OK
  RDI-P              0            0 OK
  UNEQ-P             0            0 OK
  PLM-P              0            0 OK
  ES-P               0
  SES-P              0
  UAS-P              0
  ES-PFE             0
  SES-PFE            0
  UAS-PFE            0

```

Received SONET overhead:

```

F1      : 0x00, J0      : 0x00, K1      : 0x00, K2      : 0x00
S1      : 0x00, C2      : 0xcf, C2(cmp) : 0xcf, F2      : 0x00
Z3      : 0x00, Z4      : 0x00, S1(cmp) : 0x00

```

Transmitted SONET overhead:

```

F1      : 0x00, J0      : 0x01, K1      : 0x00, K2      : 0x00
S1      : 0x00, C2      : 0xcf, F2      : 0x00, Z3      : 0x00
Z4      : 0x00

```

Received path trace: pilsener-re0 so-2/2/0

```

70 69 6c 73 65 6e 65 72 2d 72 65 30 20 73 6f 2d  pilsener-re0 so-
32 2f 32 2f 30 00 00 00 00 00 00 00 00 00 00 00 2/2/0.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 0d 00 .....

```

```

Transmitted path trace: dunkel-re0 so-2/0/0
 64 75 6e 6b 65 6c 2d 72 65 30 20 73 6f 2d 32 2f  dunkel-re0 so-2/
30 2f 30 00 00 00 00 00 00 00 00 00 00 00 00 00  0/0.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
HDLCD configuration:
  Policing bucket: Disabled
  Shaping bucket : Disabled
  Giant threshold: 4484, Runt threshold: 3
Packet Forwarding Engine configuration:
  Destination slot: 2, PLP byte: 1 (0x00)
  Direction : Output
  CoS transmit queue

```

	%	Bandwidth bps	%	Buffer usec	Priority	Limit
0 best-effort	95	590976000	95	0	low	none
3 network-control	5	31104000	5	0	low	none

```

Logical interface so-2/0/0.0 (Index 66) (SNMP ifIndex 179) (Generation 134)
Flags: Point-To-Point SNMP-Traps 0x4000 Encapsulation: PPP
Protocol inet, MTU: 4470, Generation: 141, Route table: 0
Addresses, Flags: Is-Preferred Is-Primary
  Destination: 18.32.74.0/30, Local: 18.32.74.1, Broadcast: 18.32.74.3,
  Generation: 144

```

Let's examine this output in depth, layer by layer, so you'll know what to look for in your own `show interface extensive` or `monitor interface` command.

Major elements of monitoring and troubleshooting Layer 1 and Layer 2 technologies include an understanding of how the protocol monitors, how errors can impact the operation of the connection, and how those errors are displayed to the operator.

SONET Mode

Juniper Networks SONET PICs can operate in either SONET or SDH mode.

SONET (Synchronous Optical Networking) and SDH (Synchronous Digital Hierarchy) were developed as part of Telcordia Technologies Generic Requirements document GR-253-CORE. They detail two different methods for multiplexing bit streams over a fiber optic cable. SONET and SDH are very similar protocols. SDH was developed later and can be viewed as a super-set of SONET, but because of the global ubiquity of SDH and limited use of SONET within North America, SDH is considered the standard.

The actual functioning of the protocol is very complex and is outside the scope of this book. All you need to know is that the encapsulation and interleaving performed by the SONET/SDH layer allow for extremely low latency. The SONET/SDH configuration is done at the chassis hierarchy level and applied to all ports on the PIC. SONET and SDH mismatches can cause Loss of Frame SONET errors (described later).

FCS

The FCS, or frame checksum, is used for packet validation. Juniper's default behavior is to use a 16-bit frame checksum, but can also be configured with a 32-bit checksum that improves reliability, but might not be supported on all network elements.

The quickest way to identify a checksum error is to monitor for framing errors by repeatedly running the `show interface [interface name] extensive` command or using the `monitor interface [interface name]` command. Rapidly increasing framing errors are generally indicative of a checksum error.

Payload Scrambling

Payload scrambling, bolded in the show command output, is a common culprit in a malfunctioning a SONET connection. Like many SONET parameters, payload-scrambling must agree between the two ends of a circuit, and a conflict between the two sides causes SONET errors.

Certain transport standards, such as the ITU-T GR-253 standard, require a certain density of ones (as opposed to zeroes) in the digital stream. The main purpose of this requirement is for timing recovery. The percent of ones needed for a T1, for example, is 12.5 percent, or one bit in every byte.

To meet this requirement, payload-scrambling is used as an encoding algorithm to ensure that this requirement is met and is enabled by default on the SONET interfaces of Juniper Networks routers. Troubleshooting a payload-scrambling mismatch can be tricky. A payload-scrambling mismatch does not cause SONET layer issues (such as defects) on either side of the connection, but the side with payload-scrambling enabled logs input errors which a Juniper Networks router presents as input giants. The side of the connection without payload-scrambling configured will display framing errors.

BEST PRACTICE As with FCS, it is advisable to simply check that both sides are configured with the same payload-scrambling setting, and best practice recommends that you enable payload-scrambling on both sides.

Input Errors

Many different SONET errors can trigger input errors and while their causes vary, they are indicative of a problem and their cause should be examined and corrected. Framing errors, runts, and giants are typically due to misconfiguration. Framing errors can be caused by an FCS mismatch (16 on one side and 32 on the other, for example) or payload-scrambling mismatches.

Runts occur when a received packet is smaller than the minimum frame size (64 bytes on SONET). These are the exception as they are often caused by cabling or connection problems.

Giants are the result of a received packet being larger than the maximum frame size (16KB). These can be caused by payload-scrambling mismatches as described above, but can also be caused by other conditions. If input giants continue and configuration mismatches have been ruled out, the provider of the circuit should be contacted to assist in resolving the problem.

SONET Alarms and Defects

SONET is the most complex Layer 1 technology commonly used on Juniper Networks devices, so let's use it as an example of troubleshooting at Layer 1.

To troubleshoot SONET links, it is important to understand the hierarchical nature of SONET. The SONET/SDH hierarchy is made up of three layers: section, line, and path layers as illustrated in Figure 5-1.

Section is the lowest level of the hierarchy and is defined as a single fiber connection that can be terminated by any network element such as a router, signal regenerator, or add-drop multiplexor (ADM). Devices in this role are often called *Section Terminating Equipment* (STE).

The line layer is responsible for the synchronization and multiplexing of information with SONET/SDH frames. Examples of Line Terminating Elements (LTEs) include routers and ADMs.

Finally, at the path layer, Path Terminating Elements (PTEs, such as a router) connect non-SONET/SDH networks to the SONET/SDH network. This hierarchy allows you to quickly isolate problems at a specific layer.

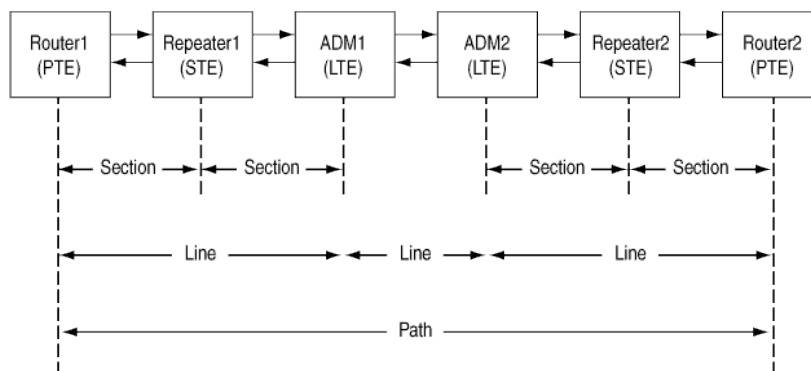


Figure 5-1 Sample SONET Network

Using Figure 5-1 as a reference, the following sections list the most common SONET errors, their possible causes, and some recommendations on where to start troubleshooting.

Loss of Signal (LOS)

A loss of signal (LOS) alarm indicates that there is a physical link problem with the connection to the router receive port from the neighboring SONET equipment transmit port. To locate the LOS alarm, check the connection between the router port and the first SONET network element. Show command extensive output in a LOS situation may include the following:

- Active alarms: LOL, PLL, LOS
- Active defects: LOL, PLL, LOF, LOS, SEF, AIS-L, AIS-P, PLM-P

Loss of Frame (LOF)

SONET uses the A1 and A2 bytes in the section overhead to align frames using specific bit patterns. If an element detects errors in this pattern for three consecutive milliseconds, an LOF error is issued. If you receive an LOF error, check the connection between the router and the first SONET network element and ensure that there is no framing mismatch (for example, SONET or SDH) between network elements.

Alarm Indication Signal (AIS)

An AIS signal is sent downstream to signal an error condition. These are the two types of AIS alarms:

- Alarm indication signal line (AIS-L): Sent by an STE to a downstream LTE when an LOS or LOF is detected on an upstream SONET section.
- Alarm indication signal path (AIS-P): Sent by an LTE to a downstream PTE when an LOS or LOF is detected on an upstream SONET section.

Remote Defect Indication (RDI)

The RDI is the complement to the AIS and is sent upstream when an error is detected. Like AIS, there are path and line versions of this signal.

- Remote defect indication line (RDI-L): Sent upstream to a peer LTE when an AIS-L or low-level defects are detected.
- Remote defect indication path (RDI-P): Sent upstream to a peer PTE when a defect in the signal, typically an AIS-P, is detected.

Remote Error Indication (REI)

A remote error indication signal is sent upstream to signal an error condition. There are two types of REI alarms:

- Remote error indication line (REI-L): Sent to the upstream LTE when errors are detected in the B2 byte.
- Remote error indication path (REI-P): Sent to the upstream PTE when errors are detected in the B3 byte.

Bit Error Rate (BER)

Bit error rate alarms are declared when the number of BIP-B2 errors hits a certain threshold. These error counters are shown in the output of the `show interface extensive` command as previously shown. Depending on the threshold, there are two types of BER alarms. In both cases, the interface is taken down.

Bit error rate-signal degrade (BERR-SD) is declared when a bit error rate of 10^{-6} is reached.

Bit error rate-signal failure (BERR-SF) is declared when a bit error rate of 10^{-3} is reached.

Bit errors can be caused by any of the following:

- Degrading optical fiber
- Optical transmitter or receiver problems
- Dirty fiber-optic connector
- Clocking issues
- Too much attenuation in the optical signal
- BIP-B1 and BIP-B3 are not used in the BER alarm calculations

Payload Label Mismatch (PLM)

The C2 byte is used as a way to identify the type of payload the SONET header is carrying, like the Ethertype field in an Ethernet header. The default for this value is CF, which indicates “IP within PPP *without* payload scrambling.”

The most common alternative setting for this byte is 16, which indicates “IP within PPP *with* payload scrambling.” A difference in the C2 values between network elements results in the circuit being down and defects of RDI-P (Remote Defect Indicator [Path]) and PLM-P (Path Label Mismatch [Path]).

The extensive output of the `show interfaces` command displays these types of errors as well as the current C2 byte setting. A C2 setting of 0x00 triggers an “unequipped payload” (UNEQ-P) alarm and indicates a provisioning problem.

Loss of Pointer Path (LOP-P)

An LOP-P alarm indicates a possible provisioning problem and occurs when the Juniper Networks router cannot determine whether or not a payload pointer is valid. The router monitors the H1 and H2 bytes, located in the line overhead. This alarm is usually discovered upon initial provisioning of SONET circuits, and is not generally seen after the circuit has been installed in the network.

Phase Lock Loop (PLL)

The PLL alarm occurs when the PLL cannot lock on to a timing device, and indicates a possible hardware or network timing problem. The

cause of the problem differs depending on the type of system board on the router. For example:

On the M20 and M40 Internet routers with the OC48-SM-IR PIC and the M160 Internet router with the OC192 board, the problem might be caused by the following:

- An out-of-tolerance clock on the transmitting device, if clocking external is configured.
- An out-of-tolerance clock on the transmitting device or a problem with the board being unable to lock on to its internal clock to derive the transmit clock, if clocking internal is configured.

To further diagnose the problem, try the following:

- Configure clocking to external. If the alarm disappears, the board might not have locked to the internal clock used to derive the outgoing clock.
- Configure clocking to internal and make sure that a loopback fiber is plugged in. If the PLL alarm persists, it is most likely a hardware problem. However, you may not be able to determine if the direction is on the inbound or outbound side of the board.

Layer 2 Monitoring and Troubleshooting

The next layer up in the OSI model, Layer 2, manages the transmission of logical data frames between adjacent network nodes on the same network segment. Examples of Layer 2 protocols are Ethernet (yes, it operates at both Layer 1 and Layer 2), frame-relay, and HDLC.

This section discusses the monitoring and troubleshooting of PPP (point-to-point protocol), a popular Layer 2 protocol used in connecting routers over WAN links and also the default protocol for serial connections on Juniper Networks routers. Like Layer 1 troubleshooting, most of what you need to examine regarding Layer 2 can be found in the output of the `show interface` command.

The following example output originated from the same SONET interface shown previously, but this time focuses on Layer 2 (PPP) information:

```
[edit]
ps@dunkel-re0> show interfaces so-1/2/3
Physical interface: so-1/2/3, Enabled, Physical link is Up
Interface index: 148, SNMP ifIndex: 133
```

```

Description: Connection to kenny so-1/2/3
Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode, Speed: OC3,
Loopback: None, FCS: 16, Payload scrambler: Enabled
Device flags   : Present Running
Interface flags: Point-To-Point SNMP-Traps Internal: 0x4000
Link flags     : Keepalives
Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
Keepalive: Input: 275076 (02:44:12 ago), Output: 275032 (02:44:09 ago)
LCP state: Opened
NCP state: inet: Opened, inet6: Not-configured, iso: Not-configured, mp1s:
Not-configured
CHAP state: Closed
PAP state: Closed
CoS queues   : 4 supported, 4 maximum usable queues
Last flapped : 2010-04-09 14:38:52 PDT (2d 23:50 ago)
Input rate    : 656 bps (0 pps)
Output rate   : 696 bps (0 pps)
SONET alarms  : None
SONET defects : None

Logical interface so-1/2/3.0 (Index 69) (SNMP ifIndex 142)
  Flags: Point-To-Point SNMP-Traps 0x4000 Encapsulation: PPP
  Protocol inet, MTU: 4470
    Flags: None
    Addresses, Flags: Is-Preferred Is-Primary
      Destination: 10.33.18.4/30, Local: 10.33.18.6, Broadcast: 10.33.18.7

```

The PPP related information is **bolded** above and shows the important aspects of its operation. LCP (Link Control Protocol) and NCP (Network Control Protocol) are protocols within the PPP suite that manage the parameters and operation of the link and network functions of PPP. LCP is responsible for negotiating link parameters (packet transmission size) and validates that the link is acceptable (based on a set of criteria).

PPP cannot continue unless LCP has moved to an open state. After LCP negotiates, NCP utilizes Layer 3 protocol modules to negotiate parameters for a multitude of Layer 3 protocols. In this way, PPP does not need to be modified to support new Layer 3 protocols. The NCP module that concerns us in this case is IPCP, the Internet Protocol Control Protocol. IPCP negotiates IP layer parameters such as the compression mechanism, IP addressing (for example, in dynamic subscriber environments) and other values. IPCP must also be open for a given Layer 3 protocol before any packet of that type can be transmitted.

NOTE Other examples of NCP modules are IPv6CP for IP version 6, IPxCP for Internet Packet Exchange (IPX) and ATCP (AppleTalk Control Protocol).

The default values for all of these parameters should work well in most cases, but as with Layer 1, some may require that both sides agree and these are a good place to start looking in the event that LCP or NCP do not proceed to an open state.

The CHAP and PAP protocols cause many PPP problems (in non-dynamic address environments). CHAP and PAP are used to authenticate the identity of the peer and typically operate after NCP completes its negotiation.

CHAP or PAP failures close NCP and LCP and the process restarts. CHAP and PAP both use passwords to authenticate the neighbor, but use different mechanisms to do so. PAP is insecure in that it sends the password in plain text, while CHAP relies on an MD5 hash to avoid this vulnerability. Neither protocol is commonly used on large serial links. They are typically limited to validating the identity of a client in a subscriber environment. If CHAP or PAP is used, it is recommended that the type of authentication and password be confirmed on both sides. Below is a sample configuration showing PAP authentication with a misconfigured password. As an example, traceoptions are used to isolate the problem. In this case, you have found you cannot pass traffic over a PPP link and that LCP is not opened.

```
ps@dunkel-re0> ping 10.33.18.5 count 5
PING 10.33.18.5 (10.33.18.5): 56 data bytes
```

```
--- 10.33.18.5 ping statistics ---
5 packets transmitted, 0 packets received, 100% packet loss
```

```
ps@dunkel-re0> show interfaces so-1/2/3
Physical interface: so-1/2/3, Enabled, Physical link is Up
  Interface index: 148, SNMP ifIndex: 133
  Description: Connection to maibock
  Link-level type: PPP, MTU: 4474, Clocking: Internal, SONET mode, Speed: OC3,
  Loopback: None, FCS: 16, Payload scrambler: Enabled
  Device flags   : Present Running
  Interface flags: Point-To-Point SNMP-Traps Internal: 0x4000
  Keepalive settings: Interval 10 seconds, Up-count 1, Down-count 3
  Keepalive: Input: 0 (never), Output: 0 (never)
LCP state: Down
  NCP state: inet: Not-configured, inet6: Not-configured, iso: Not-configured, mpls:
  Not-configured
```

```

CHAP state: Closed
PAP state: Closed
CoS queues      : 4 supported, 4 maximum usable queues
Last flapped    : 2010-04-09 14:38:52 PDT (3d 00:34 ago)
Input rate      : 80 bps (0 pps)
Output rate     : 88 bps (0 pps)
SONET alarms    : None
SONET defects   : None

Logical interface so-1/2/3.0 (Index 73) (SNMP ifIndex 521)
  Flags: Hardware-Down Point-To-Point SNMP-Traps 0x4000 Encapsulation: PPP
  Protocol inet, MTU: 4470
  Flags: Protocol-Down
  Addresses, Flags: Dest-route-down Is-Preferred Is-Primary
    Destination: 10.33.18.4/30, Local: 10.33.18.6, Broadcast: 10.33.18.7

```

The ping attempt and interface output show that LCP is down, which explains why traffic is not passing over this link. The configuration shows that CHAP authentication is used, so this is a good place to start troubleshooting.

```

ps@dunkel-re0> show configuration interfaces so-1/2/3
description "Connection to maibock";
unit 0 {
  ppp-options {
    chap {
      default-chap-secret "$9$PFF/9A00IcF3hreKx7ik.539BIcM87cyvL"; ## SECRET-DATA
    }
  }
  family inet {
    address 10.33.18.6/30;
  }
}

```

The fastest way to confirm that the problem is not CHAP related is to reenter the password ("ppp-password" in our case) on both sides of the connection. However, to show the debugging capabilities of Junos, traceoptions are used to diagnose the problem. Junos traceoptions serve to provide debugging information for a given protocol or function. Generally, protocol traceoptions are configured at the hierarchy level corresponding to that protocol. In the case of PPP, traceoptions are configured at the [edit protocols ppp] hierarchy. There are two main configuration parameters for traceoptions, the file, to which the system logs the debugging messages, and flags, which specify the types of information to be logged. For this example, the file is called "ppp-log.txt". The following shows the configuration options for PPP traceoptions:

```
[edit]
ps@dunkel-re0# set protocols ppp traceoptions flag ?
Possible completions:
access          Trace access code
address-pool    Trace address pool code
all             Trace all areas of code
auth            Trace authentication code
chap            Trace CHAP code
ci              Trace ci code
config          Trace configuration code
ifdb            Trace interface database code
lcp             Trace LCP state machine code
memory          Trace memory management code
message         Trace message processing code
mlppp           Trace MLPPP code
ncp             Trace NCP state machine code
pap             Trace PAP code
ppp             Trace PPP protocol processing code
radius          Trace RADIUS processing code
redundancy      Trace redundancy code
rtsock          Trace routing socket code
session         Trace session management code
signal          Trace signal handling code
timer           Trace timer code
ui              Trace user interface code
```

There are many traceoptions flags for PPP, but if you suspect CHAP to be the problem, that's the best place to start. PPP traceoptions also offer a level parameter, which allows you to limit the output to a certain severity. For this example, you use a level of "all" to start with, leaving the traceoptions configuration as follows:

```
ps@dunkel-re0> show configuration protocols ppp
traceoptions {
    file ppp-log.txt;
    level all;
    flag chap;
}
```

After committing this configuration, the next step in diagnosing the problem is monitoring the traceoptions file. To do so, you issue the monitor start [filename] command:

```
ps@dunkel-re0> monitor start ppp-log.txt
```

This command performs the equivalent of the “tail” command in UNIX environments and allows the user to monitor additions to a file. After a short period of time, you see the following displayed to the screen:

```
ps@dunkel-re0>
*** ppp-log.txt ***
Apr 12 16:12:20 so-1/2/3.0: CHAP - Stopping protocol timer
Apr 12 16:12:20 so-1/2/3.0: CHAP - Starting authentication
Apr 12 16:12:20 so-1/2/3.0: CHAP - End authen(0x8231004): FAILURE
```

The important line in the log is the failure error. This shows that authentication is failing and that the password should be reset on both sides.

Junos also has several CLI commands to monitor the status of PPP. The most useful of commands are `show ppp interface [interface name]` extensive and `show ppp summary`. Below are example outputs of these commands in the current down state:

```
ps@dunkel-re0> show ppp interface so-1/2/3 extensive
Sessions for interface so-1/2/3
  Session so-1/2/3.0, Type: PPP, Phase: Establish
    LCP
      State: Creq-sent
      Last started: 2010-04-12 16:16:16 PDT
      Last completed: 2010-04-12 16:16:14 PDT
      Negotiated options:
        Authentication protocol: CHAP, Authentication algorithm: MD5,
        Magic number: 2543706641, MRU: 4470
    Authentication: CHAP
      State: Closed
      Last started: 2010-04-12 16:16:14 PDT
      Last completed: 2010-04-12 16:13:26 PDT
    IPCP
      State: Closed
      Last started: 2010-04-12 16:13:26 PDT
      Last completed: 2010-04-12 16:13:26 PDT
      Negotiated options:
        Local address: 10.33.18.6, Remote address: 10.33.18.4, Primary DNS: 0.0.0.0,
        Secondary DNS: 0.0.0.0
```

```
ps@dunkel-re0> show ppp summary
Interface      Session type  Session phase  Session flags
so-1/2/3.0     PPP          Authenticate
```


In an operational setting, LCP and IPCP (and any other configured Layer 3 protocols) should be in an “Opened” state in the output of `show ppp interface extensive` and `show ppp summary` should list the session phase as “Network.”

When the passwords are reset on both sides, the following logs are displayed (you’re still monitoring, right?):

```
Apr 12 16:13:26 so-1/2/3.0: CHAP - Starting protocol timer (2 sec, 0 nsec)
Apr 12 16:13:26 so-1/2/3.0: CHAP - Stopping protocol timer
Apr 12 16:13:26 so-1/2/3.0: CHAP - Starting authentication
Apr 12 16:13:26 so-1/2/3.0: CHAP - End authen(0x8231004): SUCCESS
```

Note the SUCCESS message. Following this, you can once again ping over the circuit, and the CLI instrumentation validates that PPP is now functioning properly.

```
ps@dunkel-re0> show ppp interface so-1/2/3 extensive
```

```
Sessions for interface so-1/2/3
```

```
Session so-1/2/3.0, Type: PPP, Phase: Network
```

```
LCP
```

```
State: Opened
```

```
Last started: 2010-04-12 16:19:41 PDT
```

```
Last completed: 2010-04-12 16:19:41 PDT
```

```
Negotiated options:
```

```
Authentication protocol: CHAP, Authentication algorithm: MD5,
```

```
Magic number: 2544040945, MRU: 4470
```

```
Authentication: CHAP
```

```
State: Success
```

```
Last started: 2010-04-12 16:19:41 PDT
```

```
Last completed: 2010-04-12 16:19:41 PDT
```

```
IPCP
```

```
State: Opened
```

```
Last started: 2010-04-12 16:19:41 PDT
```

```
Last completed: 2010-04-12 16:19:41 PDT
```

```
Negotiated options:
```

```
Local address: 10.33.18.6, Remote address: 10.33.18.5, Primary DNS: 0.0.0.0,
```

```
Secondary DNS: 0.0.0.0
```

```
ps@dunkel-re0> show ppp summary
```

Interface	Session type	Session phase	Session flags
so-1/2/3.0	PPP	Network	

BEST PRACTICE Note that the states for LCP and IPCP are now “Opened” and show ppp summary shows the session phase as “Network.” Before you end your monitoring session, you should remove the traceoptions configuration used for PPP debugging. While traceoptions do not impact the router, it does create unnecessary traceoptions configuration and files.

To do so, issue the `delete protocols ppp traceoptions` command in configuration mode:

```
ps@dunkel-re0> configure
Entering configuration mode

[edit]
ps@dunkel-re0# delete protocols ppp traceoptions

[edit]
ps@dunkel-re0# commit and-quit
re0:
configuration check succeeds
re1:
commit complete
re0:
commit complete
Exiting configuration mode
```

You can stop monitoring of the `ppp-log.txt` file using the `monitor stop` command. Note that monitoring is bound to your terminal and stops automatically when you log out:

```
ps@dunkel-re0> monitor stop
```

Summary

This chapter used SONET and PPP to summarize how to monitor and troubleshoot Layer 1 and 2 alarms and errors. SONET was used because it is the most complex Layer 1 technology commonly used on Juniper Networks devices, and PPP was used because of its popularity and interoperability. However, the same approach can be used for any other Layer 1 or 2 technology, including Ethernet.

That approach, be it for SONET or Ethernet or any other Layer 1 or 2 technology, is:

- Use the instrumentation and information afforded by the protocol.
- This information is contained within the output of the `show interface extensive` command.
- While output from `show interface extensive` provides a snapshot; the `monitor interface` command provides information in real time, updating its display of an interface's most important statistics, warnings, errors, and attributes.

The following is an example of what the `monitor` command can display on a UNIX terminal:

```
dunkel-re0                      Seconds: 36                      Time: 10:32:10
                                Delay: 0/0/19

Interface: so-2/0/0, Enabled, Link is Up
Encapsulation: PPP, Keepalives, Speed: 0C12
Traffic statistics:
    Input bytes:                408874 (0 bps)                [0]
    Output bytes:               444937 (0 bps)                [0]
    Input packets:              31104 (0 pps)                 [0]
    Output packets:             31361 (0 pps)                 [0]
Encapsulation statistics:
    Input keepalives:           15527                        [0]
    Output keepalives:          15537                        [0]
    LCP state: Down
Error statistics:
    Input errors:                0                            [0]
    Input drops:                 0                            [0]
    Input framing errors:        255                          [0]
    Input runts:                 0                            [0]
    Input giants:                0                            [0]
    Policed discards:            0                            [0]
    L3 incompletes:              255                          [0]
    L2 channel errors:           0                            [0]
    L2 mismatch timeouts:        0 Carrier transitiz         [0]
Interface warnings:
    o Received keepalive count is zero
    o INET NCP is not Opened
    o LCP state is not Opened
Next='n', Quit='q' or ESC, Freeze='f', Thaw='t', Clear='c', Interface='i'
```

BEST PRACTICE Ensure that Layer 1 settings such as FCS, payload-scrambling (both SONET and SDH), speed, flow-control, and duplex-mode (all Ethernet) and Layer 3 settings such as authentication *agree*. Also, understand the role your device plays in the network (this is more important in SONET networks than Ethernet networks).

Chapter 6

Layer 3 Monitoring

<i>RIP</i>	76
<i>IS-IS</i>	78
<i>OSPF</i>	85
<i>BGP</i>	91
<i>Summary</i>	98

So far, this Day One booklet has discussed the monitoring and troubleshooting of Layer 1 and Layer 2 protocols. Layer 1 and Layer 2 issues are typically isolated to contained systems whose configuration is only significant locally.

For Layer 3 monitoring and troubleshooting, it is important to understand that an IP network is an interconnected, mutually dependent system in which outages and changes can have a global impact.

This chapter discusses how to effectively monitor a Layer 3 IP network. Much like Layer 1 and Layer 2 monitoring, most operators rely on syslog, SNMP, and monitoring systems to identify problems. Layer 3 network monitoring is much more complicated than it is at Layer 1 and Layer 2. This is because there is more data to process and so there are more possible points of failure. The interconnected nature of an IP network also makes it more difficult to quickly identify root-causes.

On a Layer 3 IP network, monitoring includes:

- Logging configurations associated with commonly used protocols to ensure that the NMS systems are receiving the appropriate logs.
- Logs and SNMP traps and some of the important, protocol related SNMP objects.
- Useful operational mode commands for commonly deployed IP protocols, including RIP, IS-IS, OSPF, and BGP.
- The routing protocol process (RPD), which, among other things, is responsible for logging and SNMP traps for route protocol-related events.

NOTE Each protocol has its own set of logs and traps that are created for events specific to the protocol. Some of these logs and traps are automatic and some require configuration. For example, BGP will not log neighbor state changes without configuring the `log-updown` statement.

RIP

Since RIP is a distance-vector protocol, it does not form adjacencies like a link-state protocol such as OSPF or IS-IS. RIP advertisements are simply multicast on the network and any interested router can process the updates. Because of this passive learning, there is no neighbor state, and as such no RIP related logs when RIP routers on a network segment come up or go down.

RIP Operational Mode Commands

To monitor RIP operation, begin by checking the standard instrumentation available on your device.

Somewhat confusingly named, the `show rip neighbor` command displays the interfaces configured for RIP, rather than displaying any information about RIP neighbors (which, as discussed previously, doesn't exist):

```
ps@dunkel-re0> show rip neighbor
```

Neighbor	State	Source Address	Destination Address	Send Mode	Receive Mode	In Met
so-2/0/0.0	Up	18.32.74.1	224.0.0.9	mcast	both	1

For this example, the router is configured to send and receive RIP updates on interface `so-2/0/0`, which can be confirmed in the configuration (don't forget to always get a second opinion):

```
ps@dunkel-re0> show configuration protocols rip
group core-interfaces {
    neighbor so-2/0/0.0;
}
```

The configuration tells us that the router is doing what it was asked to do. It would be useful to see if the router is sending and receiving RIP updates to get a better understanding of its operation, and it just so happens that there is a command for that, `show rip statistics`:

```
ps@dunkel-re0> show rip statistics
RIPv2 info: port 520; holddown 120s.
      rts learned  rts held down  rqsts dropped  resps dropped
              0              0              0              0

so-2/0/0.0: 1 routes learned; 0 routes advertised; timeout 180s; update interval 30s
```

Counter	Total	Last 5 min	Last minute
Updates Sent	0	0	0
Triggered Updates Sent	0	0	0
Responses Sent	0	0	0
Bad Messages	0	0	0
RIPv1 Updates Received	0	0	0
RIPv1 Bad Route Entries	0	0	0
RIPv1 Updates Ignored	0	0	0
RIPv2 Updates Received	7	7	3
RIPv2 Bad Route Entries	0	0	0
RIPv2 Updates Ignored	0	0	0
Authentication Failures	0	0	0
RIP Requests Received	0	0	0
RIP Requests Ignored	0	0	0

As you can see, RIP is receiving RIPv2 update messages, which is a good sign. The lack of any output updates is acceptable since in Junos by default RIP does not advertise any routing information. You would have to configure RIP to export routing information to see any updates in this table. For this network, the router on the other side of this SONET link is advertising routing information (hence the RIPv2 Updates Received and the 1 routes learned counter shown in the output in bold).

The `show route protocol rip` command can confirm that the device is receiving and processing a RIP update:

```
ps@dunkel-re0> show route protocol rip

inet.0: 10 destinations, 12 routes (10 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.7.2/32      *[RIP/100] 00:00:02, metric 2, tag 0
> to 18.32.74.2 via so-2/0/0.0
224.0.0.9/32      *[RIP/100] 00:15:13, metric 1
MultiRecv

__juniper_private1__inet.0: 5 destinations, 6 routes (5 active, 0 holddown, 0 hidden)
__juniper_private2__inet.0: 1 destinations, 1 routes (0 active, 0 holddown, 1 hidden)
__juniper_private1__inet6.0: 5 destinations, 7 routes (5 active, 0 holddown, 0 hidden)
```

IS-IS

Since IS-IS is a link-state protocol, it has a richer set of monitoring tools. This section discusses the logging, SNMP traps, and instrumentation tools that can be used on the router to monitor its IS-IS operation.

Logging

IS-IS logs information based on the finite state machine of the adjacency process. These logs are important to monitoring the network for IP problems because your IGP is either directly related to network reachability (if all of your routes are kept within your IGP), or indirectly related to connectivity by providing next-hop reachability for BGP and as the underlying protocol for MPLS.

Additionally, IGP down events may (and likely do) trigger reconvergence within your network, which may lead to the overutilization of other links or a suboptimal route. In any event, it is important to know

when your topology changes, both to resolve the outage of a particular adjacency and to mitigate for any unexpected consequences. The basic IS-IS logs for adjacency changes are *adjacency up* and *adjacency down*. Our adjacency is configured as a Level-2 circuit and is an otherwise ordinary configuration as shown here:

```
ps@dunkel-re0> show configuration protocols isis
level 1 disable;
interface so-2/0/0.0;
```

When the adjacency first comes up, the router logs an ADJUP syslog message:

```
Jan 22 15:15:50 172.19.110.171 rpd[4550]: RPD_ISIS_ADJUP: IS-IS new L2 adjacency to
pilsener-re0 on so-2/0/0.0
```

A link-down situation triggers the following log:

```
Jan 22 15:28:21 172.19.110.171 rpd[4550]: RPD_ISIS_ADJDOWN: IS-IS lost L2 adjacency to
pilsener-re0 on so-2/0/0.0, reason: Interface Down
```

If the router on the other side of the connection suffers a soft failure in which the routing-engine goes down but the link stays up (causing a loss of keepalives) the following message is logged:

```
Jan 22 15:31:25 172.19.110.171 rpd[4550]: RPD_ISIS_ADJDOWN: IS-IS lost L2 adjacency to
pilsener-re0 on so-2/0/0.0, reason: Aged Out
```

And finally, in the event that there is a level mismatch in the configuration, you see the following message:

```
Jan 22 15:32:33 172.19.110.171 rpd[4550]: RPD_ISIS_ADJDOWN: IS-IS lost L2 adjacency to
pilsener-re0 on so-2/0/0.0, reason: Level Mismatch
```

SNMP Traps

Junos can also be configured to send SNMP traps when an IS-IS state change occurs. By default, Junos does not send a trap when this happens, so you need to configure an event-option to ensure that traps are sent. Event-options allow you to specify an action to take when a particular local log event occurs, as shown in the following event-options configuration:

```
ps@dunkel-re0> show configuration event-options
policy isisNbrStateChange {
    events [ rpd_isis_adjdown rpd_isis_adjup ];
    then {
        raise-trap;
    }
}
```

This event-option configuration causes the router to send an SNMP trap when either `rpd_isis_adjdown` or `rpd_isis_adjup` appears in the logs file. In the following example, examine the same set of adjacency changes and review the resulting SNMP traps.

IS-IS adjacency up:

```
dunkel-re0.juniper.net [UDP: [172.19.110.171]:62313]: Trap , DISMAN-EVENT-
MIB::sysUpTimeInstance = Timeticks: (27422739) 3 days, 4:10:27.39,
SNMPv2-MIB::snmpTrapOID.0 = OID:
SNMPv2-SMI::enterprises.2636.4.12.0.1,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.2.2 = STRING: "RPD_ISIS_ADJUP",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.3.2 = Hex-STRING: 07 DA 01 16 17 39 33
00 2B 00 00 ,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.4.2 = INTEGER: 7,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.5.2 = INTEGER: 4,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.6.2 = Gauge32: 4550,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.7.2 = STRING: "rpd",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.8.2 = STRING: "dunkel-re0",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.9.2 = STRING: "RPD_ISIS_ADJUP: IS-IS new L2
adjacency to pilsener-re0 on so-2/0/0.0",
SNMPv2-MIB::snmpTrapEnterprise.0 = OID: SNMPv2-SMI::enterprises.2636.1.1.1.2.9
```

IS-IS link down:

```
dunkel-re0.juniper.net [UDP: [172.19.110.171]:62313]: Trap , DISMAN-EVENT-
MIB::sysUpTimeInstance = Timeticks: (27426926) 3 days, 4:11:09.26,
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.2636.4.12.0.1,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.2.3 = STRING: "RPD_ISIS_ADJDOWN",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.3.3 = Hex-STRING: 07 DA 01 16 17 3A 20
00 2B 00 00 , SNMPv2-SMI::enterprises.2636.3.35.1.1.1.4.3 = INTEGER: 6,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.5.3 = INTEGER: 4,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.6.3 = Gauge32: 4550,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.7.3 = STRING: "rpd",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.8.3 = STRING: "dunkel-re0",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.9.3 = STRING: "RPD_ISIS_ADJDOWN:
IS-IS lost L2 adjacency to pilsener-re0 on so-2/0/0.0, reason: Interface Down",
SNMPv2-MIB::snmpTrapEnterprise.0 = OID: SNMPv2-SMI::enterprises.2636.1.1.1.2.9
Soft failure on the remote side:
dunkel-fxp0.pslab.juniper.net [UDP: [172.19.110.171]:62313]: Trap , DISMAN-EVENT-
MIB::sysUpTimeInstance = Timeticks: (27434764) 3 days, 4:12:27.64,
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.2636.4.12.0.1,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.2.5 = STRING: "RPD_ISIS_ADJDOWN",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.3.5 = Hex-STRING: 07 DA 01 16 17 3B 33
00 2B 00 00 , SNMPv2-SMI::enterprises.2636.3.35.1.1.1.4.5 = INTEGER: 6,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.5.5 = INTEGER: 4,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.6.5 = Gauge32: 4550,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.7.5 = STRING: "rpd",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.8.5 = STRING: "dunkel-re0",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.9.5 = STRING: "RPD_ISIS_ADJDOWN:
IS-IS lost L2 adjacency to pilsener-re0 on so-2/0/0.0, reason: Aged Out",
SNMPv2-MIB::snmpTrapEnterprise.0 = OID: SNMPv2-SMI::enterprises.2636.1.1.1.2.9
```

Level mismatch:

```
dunkel-fxp0.pslab.juniper.net [UDP: [172.19.110.171]:62313]: Trap , DISMAN-EVENT-
MIB::sysUpTimeInstance = Timeticks: (27438490) 3 days, 4:13:04.90,
SNMPv2-MIB::snmpTrapOID.0 = OID: SNMPv2-SMI::enterprises.2636.4.12.0.1,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.2.7 = STRING: "RPD_ISIS_ADJDOWN",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.3.7 = Hex-STRING: 07 DA 01 17 00 00 1C
00 2B 00 00 , SNMPv2-SMI::enterprises.2636.3.35.1.1.1.4.7 = INTEGER: 6,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.5.7 = INTEGER: 4,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.6.7 = Gauge32: 4550,
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.7.7 = STRING: "rpd",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.8.7 = STRING: "dunkel-re0",
SNMPv2-SMI::enterprises.2636.3.35.1.1.1.9.7 = STRING: "RPD_ISIS_ADJDOWN:
IS-IS lost L2 adjacency to pilsener-re0 on so-2/0/0.0, reason: Level Mismatch",
SNMPv2-MIB::snmpTrapEnterprise.0 = OID: SNMPv2-SMI::enterprises.2636.1.1.1.2.9
```

IS-IS Operational Mode Commands

There are numerous IS-IS protocol operational commands. These commands can display information regarding IS-IS adjacencies, protocol status, and routing information. The first and most commonly used is the `show isis adjacency` command. This command displays the current IS-IS neighbors and their status. The status reflects the neighborship's progress through the finite state machine (FSM).

For this discussion, let's generalize the IS-IS FSM to the simple state-ment that up is good. An IS-IS adjacency should cycle through the new and initializing stages and evolve into an up state:

```
ps@dunkel-re0> show isis adjacency
Interface      System      L State      Hold (secs) SNPA
so-2/0/0.0     pilsener-re0 2 New        23

ps@dunkel-re0> show isis adjacency
Interface      System      L State      Hold (secs) SNPA
so-2/0/0.0     pilsener-re0 2 Initializing 26

ps@dunkel-re0> show isis adjacency
Interface      System      L State      Hold (secs) SNPA
so-2/0/0.0     pilsener-re0 2 Up         26
```

There are many reasons why an adjacency may not form but some of the common causes include:

- Interface not configured for family ISO
- Missing or duplicate NSAP address (in Junos, the NSAP address is configured as the family iso address on the lo0.0 interface)

- Level mismatch
- Area mismatch
- Authentication type or key failure

In each of these cases, confirm that the configuration matches on both sides of the link.

Now that there is an adjacency, there should be some useful information in our IS-IS database. IS-IS maintains one database for Level 1 information and another for Level 2 information. As Level 1 has been disabled, you should only see the Level 2 database. View the IS-IS database by using the detail output option of the `show isis database` command:

```
ps@dunkel-re0> show isis database detail
IS-IS level 1 link-state database:
```

```
IS-IS level 2 link-state database:
```

```
pilsener-re0.00-00 Sequence: 0x7, Checksum: 0xd28e, Lifetime: 1155 secs
  IS neighbor: dunkel-re0.00          Metric:      10
  IP prefix: 10.200.7.2/32            Metric:      0 Internal Up
  IP prefix: 18.32.74.0/30            Metric:      10 Internal Up
```

```
dunkel-re0.00-00 Sequence: 0x1e5, Checksum: 0x75f2, Lifetime: 1163 secs
  IS neighbor: pilsener-re0.00        Metric:      10
  IP prefix: 10.200.7.1/32            Metric:      0 Internal Up
  IP prefix: 18.32.74.0/30            Metric:      10 Internal Up
```

As one would expect, only IS-IS information shows in our Level 2 database. The database shows the router's local loopback address, the /30 network connects to our neighbor (pilsener-re0) and Pilsener's loopback address. Since the information is in our IS-IS database, our router should be able to derive routing information based on IS-IS. Check this by issuing the `show route protocol isis` command:

```
ps@dunkel-re0> show route protocol isis
```

```
inet.0: 8 destinations, 9 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
10.200.7.2/32      *[IS-IS/18] 00:00:41, metric 10
                  > to 18.32.74.2 via so-2/0/0.0
```

Excellent! Based on the information in IS-IS database, the router selected a route to Pilsener's loopback address.

It might also be useful to monitor IS-IS for its overall operation. The `show isis overview` and `show isis statistics` commands are useful for IS-IS protocol monitoring. Much of the `show isis overview` output is based on the configuration:

```
ps@dunkel-re0> show isis overview
Instance: master
Router ID: 10.200.7.1
Adjacency holddown: enabled
Maximum Areas: 3
LSP life time: 1200
Attached bit evaluation: enabled
SPF delay: 200 msec, SPF holddown: 5000 msec, SPF rapid runs: 3
IPv4 is enabled, IPv6 is enabled
Traffic engineering: enabled
Restart: Enabled
  Restart duration: 210 sec
  Helper mode: Enabled
Level 1
  Internal route preference: 15
  External route preference: 160
  Wide metrics are enabled, Narrow metrics are enabled
Level 2
  Internal route preference: 18
  External route preference: 165
  Wide metrics are enabled, Narrow metrics are enabled
```

This output shows the IP router-id, which can be configured under the `[edit routing-options]` hierarchy level and defaults to the IP address of the `lo0.0` interface or, if a `lo0.0` interface does not exist, the lowest numbered IP interface.

Since the `lo0.0` interface has been configured with IP address 10.200.7.1, the router has selected this address as our router ID. You also see some default values for the LSP lifetime, attached bit evaluation, SPF options, MPLS traffic-engineering support, graceful-restart parameters, as well as level specific configurations including route preferences and metric styles. These are all configurable parameters.

ALERT! Understand what the default values should be in your network. Unless you have a specific reason to adjust these configurable parameters, it's best to leave them alone.

The `show isis statistics` command displays the statistical information regarding IS-IS operation, both in its communication with neighbors and protocol processing:

```
ps@dunkel-re0> show isis statistics
```

```
IS-IS statistics for dunkel-re0:
```

PDU type	Received	Processed	Drops	Sent	Rexmit
LSP	2	2	0	13	0
IIH	7	7	0	7	0
CSNP	45	45	0	90	0
PSNP	12	12	0	1	0
Unknown0	0	0	0	0	0
Totals	66	66	0	111	0

```
Total packets received: 66 Sent: 104
```

```
SNP queue length: 0 Drops: 0
```

```
LSP queue length: 0 Drops: 0
```

```
SPF runs: 8
```

```
Fragments rebuilt: 13
```

```
LSP regenerations: 0
```

```
Purges initiated: 0
```

The first section summarizes how many packets were received, processed, dropped, sent, and retransmitted based on the IS-IS message type. Rapidly increasing drops and/or retransmits are generally indicative of a problem and you should start with Layer 1 and troubleshoot this connection. The second section, starting with `Total` packets received, displays the processing information. For our case, there are no SNP (Sequence Number PDUs) or LSP (Link-State PDU) packets in the queue. If these values are consistently non-zero or the drops counters are incrementing, or both, there may be a problem.

SPF runs are the number of times the shortest path first algorithm has been run. If this number is increasing quickly, this usually means that a network link is flapping or another problem is causing redistributed routes (into IS-IS) to be added and withdrawn. The `show isis spf` command can provide more information on SPF-specific information, such as how frequently SPF is running and what is triggering the SPF runs:

```
ps@dunkel-re0> show isis spf log
```

```
IS-IS level 1 SPF log:
```

```
IS-IS level 2 SPF log:
```

Start time	Elapsed (secs)	Count	Reason
Fri Jan 29 11:36:31	0.000015	2	Reconfig
Fri Jan 29 11:36:41	0.000032	3	New adjacency pilsener-re0 on so-2/0/0.0

This output shows that there have been 3 SPF runs, with the most recent due to a new adjacency. As SPF is not constantly running, it appears the IS-IS network is stable.

OSPF

From an operation perspective, OSPF and IS-IS are very similar. Both form relationships with neighbors using a series of hello messages and a finite state machine (FSM). Both create and synchronize link-state databases and both derive routing-information from these databases.

Logging

Just as logging is important to IS-IS it is also important when running OSPF. The following shows some of the OSPF logging available.

Neighbor up:

```
Jan 28 09:52:34 172.19.110.171 rpd[4550]: RPD_OSPF_NBRUP: OSPF neighbor 18.32.74.2
(so-2/0/0.0) state changed from Init to ExStart due to 2WayRcvd (event reason: neighbor
detected this router)
Jan 28 09:52:34 172.19.110.171 rpd[4550]: RPD_OSPF_NBRUP: OSPF neighbor 18.32.74.2
(so-2/0/0.0) state changed from Exchange to Full due to ExchangeDone (event reason: DBD
exchange of slave completed)
```

Link-down:

```
Jan 28 09:54:38 172.19.110.171 rpd[4550]: RPD_OSPF_NBRDOWN: OSPF neighbor 18.32.74.2
(so-2/0/0.0) state changed from Full to Down due to KillNbr (event reason: interface
went down)
```

Soft-failure of remote side:

```
Jan 28 09:55:56 172.19.110.171 rpd[4550]: RPD_OSPF_NBRDOWN: OSPF neighbor 18.32.74.2
(so-2/0/0.0) state changed from Full to Down due to InActiveTimer (event reason:
neighbor was inactive and declared dead)
```

SNMP Traps

Unlike IS-IS, Junos by default sends SNMP traps when there is an adjacency change for OSPF. As with IS-IS, let's show these different traps.

Neighbor up:

```
dunkel-re0.juniper.net [UDP: [172.19.110.171]:62313]: Trap , DISMAN-EVENT-
MIB::sysUpTimeInstance = Timeticks: (77178300) 8 days, 22:23:03.00,
SNMPv2-MIB::snmpTrapOID.0 = OID:
SNMPv2-SMI::mib-2.14.16.2.2,
SNMPv2-SMI::mib-2.14.1.1.0 = IpAddress: 10.200.7.1,
```

```
SNMPv2-SMI::mib-2.14.10.1.1.18.32.74.2.0 = IPAddress: 18.32.74.2,
SNMPv2-SMI::mib-2.14.10.1.2.18.32.74.2.0 = INTEGER: 0,
SNMPv2-SMI::mib-2.14.10.1.3.18.32.74.2.0 = IPAddress: 10.200.7.2,
SNMPv2-SMI::mib-2.14.10.1.6.18.32.74.2.0 = INTEGER: 8,
SNMPv2-MIB::snmpTrapEnterprise.0 = OID:
SNMPv2-SMI::enterprises.2636.1.1.1.2.9
```

SNMP traps are admittedly cryptic (and there are open source tools such as `snmptt` to help decrypt them). Nonetheless, the information you need is in there. In the above trap, the important portion of the trap is in bold. The object `2.14.10.1.6.[neighbor address].0` is in the `ospfNbrState` MIB. Each potential value means:

- 1 - down*
- 2 - attempt*
- 3 - init*
- 4 - twoWay*
- 5 - exchangeStart*
- 6 - exchange*
- 7 - loading*
- 8 - full*

Based on the information from this SNMP trap, the OSPF neighbor is in state 8 (full).

Link-down:

```
dunkel-re0.juniper.net [UDP: [172.19.110.171]:62313]: Trap , DISMAN-EVENT-
MIB::sysUpTimeInstance = Timeticks: (77303649) 8 days, 22:43:56.49,
SNMPv2-MIB::snmpTrapOID.0 = OID:
SNMPv2-SMI::mib-2.14.16.2.2,
SNMPv2-SMI::mib-2.14.1.1.0 = IPAddress: 10.200.7.1,
SNMPv2-SMI::mib-2.14.10.1.1.18.32.74.2.0 = IPAddress: 18.32.74.2,
SNMPv2-SMI::mib-2.14.10.1.2.18.32.74.2.0 = INTEGER: 0,
SNMPv2-SMI::mib-2.14.10.1.3.18.32.74.2.0 = IPAddress: 10.200.7.2,
SNMPv2-SMI::mib-2.14.10.1.6.18.32.74.2.0 = INTEGER: 1,
SNMPv2-MIB::snmpTrapEnterprise.0 = OID:
SNMPv2-SMI::enterprises.2636.1.1.1.2.9
```

For this example, the trap indicates that neighbor 18.32.74.2 transitioned to state 1 (down). Received at the same time was a link-down trap that helped us to understand the reason for the OSPF down trap:

```
10.200.7.1: Link Down Trap (0) Uptime: 8 days, 22:43:56.49, IF-MIB::ifIndex.179 =
INTEGER: 179, IF-MIB::ifAdminStatus.179 = INTEGER: up(1), IF-MIB::ifOperStatus.179 =
INTEGER: down(2), IF-MIB::ifName.179 = STRING: so-2/0/0.0
```


SNMP Polling

All OSPF object identifiers (OIDs) fall under the OSPF management information base (MIB), which has an OID of 1.3.6.1.2.1.14. While there are many OSPF objects, the most commonly monitored MIB is the OSPF neighbor state (ospfNbrState). This OID uses the format 1.3.6.1.2.1.14.10.1.6.[neighbor address]. Note that this is the neighbor address and not the neighbor's router-id, as you could have multiple OSPF adjacencies to the same neighbor.

```
nms-1> snmpwalk -v2c -c tr4pp15t cartman 1.3.6.1.2.1.14.10.1.6
SNMPv2-SMI::mib-2.14.10.1.6.18.32.74.2.0 = INTEGER: 8
```

The value of this object follows the same mapping as discussed previously with 8 being mapped as “full.” So the adjacency with the neighbor at 18.32.74.2 is in a full state.

OSPF Operational Commands

Due to the similarities between the two protocols, OSPF's operational mode commands are very similar to those of IS-IS. There are OSPF commands for checking neighbor and database state, process overview, and OSPF learned routes.

Let's start by monitoring the neighborship process. Like IS-IS, OSPF neighborship follows a finite-state-machine that ultimately progresses to full state. On point-to-point links, the neighbors progress to full very quickly:

```
ps@dunkel-re0> show ospf neighbor
```

Address	Interface	State	ID	Pri	Dead
18.32.74.2	so-2/0/0.0	Full	10.200.7.2	128	39

On a multi-access network such as an Ethernet LAN, this takes longer since the routers must elect a designated router (DR) and a backup designated router (BDR). The neighborship will start in an init state as shown below and progress into the 2-way state, which is the stage in which the DR and BDR are elected. Once that step is completed, the routers will continue on to database synchronization and, finally, graduate to full synchronization. The following outputs show this progression.

```
ps@dunkel-re0> show ospf neighbor
```

Address	Interface	State	ID	Pri	Dead
172.19.110.187	fe-0/0/0.0	Init	10.200.7.2	128	34

```
ps@dunkel-re0> show ospf neighbor
```

Address	Interface	State	ID	Pri	Dead
172.19.110.187	fe-0/0/0.0	2Way	10.200.7.2	128	37

```
ps@dunkel-re0> show ospf neighbor
```

Address	Interface	State	ID	Pri	Dead
172.19.110.187	fe-0/0/0.0	Full	10.200.7.2	128	35

You may see an OSPF neighborship in the following states:

- **Init:** received hello from a router *without* its own router-id in the neighbor list.
- **2-way:** received hello from a router *with* its own router-id in the neighbor list, DR, and BDR election stage.

Note that this is as far as two non-DR routers will evolve on a multi-access network.

- **ExStart:** master/slave relationship and sequence numbers established
- **Exchange:** database descriptor packets are exchanged, missing LSAs requested
- **Loading:** requested LSAs are added to the database

Again, as with IS-IS, there are a multitude of reasons why an adjacency does not form. The common causes include:

- Area mismatch
- Authentication type or key failure
- Timer mismatch
- Subnet mismatch
- MTU mismatch (the adjacency will be stuck in EXSTART, remember this one, it will happen to you!)

For all of these cases, confirm that the configuration matches on both sides on the link.

Now that there is an OSPF neighborship in full state, there is some useful information in the OSPF database. Like the IS-IS protocol's database separation (based on levels), OSPF creates separate databases for each area the router is connected to. For this example, the router is only connected to area 0, so there's only information in the area 0 database as the following shows:

```

ps@dunkel-re0> show ospf database detail
  OSPF link state database, Area 0.0.0.0
  Type      ID          Adv Rtr      Seq      Age  Opt  Cksum  Len
Router *10.200.7.1      10.200.7.1    0x80000006  290  0x22 0x307a  60
  bits 0x0, link count 3
  id 10.200.7.1, data 255.255.255.255, Type Stub (3)
  TOS count 0, TOS 0 metric 0
  id 10.200.7.2, data 18.32.74.1, Type PointToPoint (1)
  TOS count 0, TOS 0 metric 1
  id 18.32.74.0, data 255.255.255.252, Type Stub (3)
  TOS count 0, TOS 0 metric 1
Router 10.200.7.2      10.200.7.2    0x80000005  303  0x22 0x3078  60
  bits 0x0, link count 3
  id 10.200.7.2, data 255.255.255.255, Type Stub (3)
  TOS count 0, TOS 0 metric 0
  id 10.200.7.1, data 18.32.74.2, Type PointToPoint (1)
  TOS count 0, TOS 0 metric 1
  id 18.32.74.0, data 255.255.255.252, Type Stub (3)
  TOS count 0, TOS 0 metric 1

```

Here, routers Dunkel and Pilsener are both advertising type 1 (router) LSAs, which include information on the connections of each router. Since both routers only have 1 OSPF connection (to each other), their router LSAs look similar. Both include their own router-ids as well as the connection between them. With information in the databases, the routers can derive routing information. Dunkel has learned 10.200.7.2 (Pilsener's loopback) through OSPF and has installed a route over the OSPF connection to that destination.

You can identify installed routes by those denoted with a plus sign or an asterisk, as displayed in the output of the `show route protocol ospf` command:

```

ps@dunkel-re0> show route protocol ospf
inet.0: 14 destinations, 17 routes (14 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.200.7.2/32      *[OSPF/10] 00:10:15, metric 1
> via so-2/0/0.0
18.32.74.0/30     [OSPF/10] 00:10:20, metric 1
> via so-2/0/0.0
224.0.0.5/32      *[OSPF/10] 00:10:47, metric 1
MultiRecv

```

You can also see that there is an OSPF route to 18.32.74.0/30, which is the network connecting Dunkel to Pilsener. The reason this route is not installed is that Dunkel is a directly connected (read: *better*) route to this network. Let's show its route:

```
ps@dunkel-re0> show route 18.32.74.0/30
```

```
inet.0: 14 destinations, 17 routes (14 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
18.32.74.0/30      *[Direct/0] 00:15:06
                  > via so-2/0/0.0
                  [OSPF/10] 00:15:05, metric 1
                  > via so-2/0/0.0
```

Finally, as soon as OSPF is enabled, an OSPF route for 224.0.0.5 is created. This multicast group is used by OSPF, and all OSPF routers are required to listen to it. OSPF also uses the multicast group 224.0.0.6, to which all DRs and BDRs are required to listen.

The status of the OSPF process can be examined using the `show ospf overview` and the `show ospf statistics` commands:

```
ps@dunkel-re0> show ospf overview
```

```
Instance: master
Router ID: 10.200.7.1
Route table index: 0
Full SPF runs: 6, SPF delay: 0.200000 sec, SPF holddown: 5 sec, SPF rapid runs: 3
LSA refresh time: 50 minutes
Area: 0.0.0.0
Stub type: Not Stub
Authentication Type: None
Area border routers: 0, AS boundary routers: 0
Neighbors
Up (in full state): 1
```

This command shows the values of some configurable parameters such as the SPF-delay (time to wait after a topology change to run SPF) and the SPF holddown (time to wait between SPF runs). Additionally, this command provides valuable information about the network such as the areas to which this router is connected, the types of areas it is connected to (Not Stub/Stub/Stub NSSA), any authentication used for neighbors in those areas, and the number of ABRs, ASBRs and neighbors in those areas.

The `show ospf statistics` command, like the `show isis statistics` command, displays the user with information on the types and numbers of packets sent and received, as well as additional information regarding the transmission, retransmission, and processing of OSPF LSAs. This command also provides information on the LSA queue and received errors, both of which should remain at or near 0:

```
ps@dunkel-re0> show ospf statistics
```

Packet type	Total		Last 5 seconds	
	Sent	Received	Sent	Received
Hello	7	5	0	0
DbD	3	2	0	0
LSReq	0	0	0	0
LSUpdate	4	3	0	0
LSAck	2	3	0	0

DBDs retransmitted :	0, last 5 seconds :	0
LSAs flooded :	2, last 5 seconds :	0
LSAs flooded high-prio :	0, last 5 seconds :	0
LSAs retransmitted :	2, last 5 seconds :	0
LSAs transmitted to nbr :	0, last 5 seconds :	0
LSAs requested :	0, last 5 seconds :	0
LSAs acknowledged :	2, last 5 seconds :	0

Flood queue depth :	0
Total retransmit entries :	0
db summaries :	0
lsreq entries :	0

Receive errors:
None

BGP

BGP behaves like a distance-vector protocol, but it also forms neighbor relationships with peers. There is a wealth of syslogging, SNMP polling and trapping, and instrumentation options to monitor the state and operation of BGP.

Let's first review some of the syslog messages associated with BGP.

Logging

BGP syslog messages most commonly pertain to the state of a BGP session. On Juniper Networks routers, you must configure BGP to log session state changes by issuing the `set protocols bgp log-updown` command. This command can also be used at the BGP group or neighbor level, but is most useful at the protocol level.

Here, a simple internal BGP session which establishes peering between Dunkel and Pilsener is shown. Also shown are the resulting syslog messages of the session going up and going down:

```

ps@dunkel-re0> show configuration protocols bgp
log-updown;
group ibgp {
    type internal;
    local-address 10.200.7.1;
    peer-as 1;
    neighbor 10.200.7.2;
}

```

Neighbor up:

```

Jan 29 11:47:44 172.19.110.171 rpd[4550]: RPD_BGP_NEIGHBOR_STATE_CHANGED: BGP peer
10.200.7.2 (Internal AS 1) changed state from OpenConfirm to Established (event
RecvKeepAlive)

```

Neighbor down due to peer reset:

```

Jan 29 11:48:25 172.19.110.171 rpd[4550]: bgp_read_v4_message: NOTIFICATION received
from 10.200.7.2 (Internal AS 1): code 6 (Cease) subcode 4 (Administratively Reset)
Jan 29 11:48:25 172.19.110.171 rpd[4550]: RPD_BGP_NEIGHBOR_STATE_CHANGED: BGP peer
10.200.7.2 (Internal AS 1) changed state from Established to Idle (event RecvNotify)

```

Neighbor down due to peer not being configured on the remote side:

```

Jan 29 11:50:38 172.19.110.171 rpd[4550]: bgp_read_v4_message: NOTIFICATION received
from 10.200.7.2 (Internal AS 1): code 6 (Cease) subcode 3 (Peer Unconfigured)
Jan 29 11:50:38 172.19.110.171 rpd[4550]: RPD_BGP_NEIGHBOR_STATE_CHANGED: BGP peer
10.200.7.2 (Internal AS 1) changed state from Established to Idle (event RecvNotify)

```

Neighbor down due peer timeout:

```

Jan 29 11:54:28 172.19.110.171 rpd[4550]: bgp_hold_timeout: NOTIFICATION sent to
10.200.7.2 (Internal AS 1): code 4 (Hold Timer Expired Error), Reason: holdtime expired
for 10.200.7.2 (Internal AS 1), socket buffer sndcc: 57 rcvcc: 0 TCP state: 4, snd_una:
2737938411 snd_nxt: 2737938468 snd_wnd: 16417 rcv_nxt: 3889039076 rcv_adv: 3889055460,
hold timer 0
Jan 29 11:54:28 172.19.110.171 rpd[4550]: RPD_BGP_NEIGHBOR_STATE_CHANGED: BGP peer
10.200.7.2 (Internal AS 1) changed state from Established to Idle (event HoldTime)

```

SNMP Traps

BGP creates SNMP traps for neighbor state changes and these are contained in the `mib-2.15.3.1.2.[neighbor address]` object. The state is communicated using an integer with the following mappings:

- 1 - idle*
- 2 - connect*
- 3 - active*

4 - *opensent*
 5 - *openconfirm*
 6 - *established*

The following shows some examples of BGP traps based on neighbor state change, focusing on our BGP peer at 10.200.7.2.

Neighbor up:

```
cartman-fxp0.pslab.juniper.net [UDP: [172.19.110.171]:62313]: Trap , DISMAN-
EVENT-MIB::sysUpTimeInstance = Timeticks: (87163316) 10 days, 2:07:13.16,
SNMPv2-MIB::snmpTrapOID.0 = OID:
SNMPv2-SMI::mib-2.15.7.1,
SNMPv2-SMI::mib-2.15.3.1.14.10.200.7.2 = Hex-STRING: 04 00 ,
SNMPv2-SMI::mib-2.15.3.1.2.10.200.7.2 = INTEGER: 6,
SNMPv2-MIB::snmpTrapEnterprise.0 = OID:
SNMPv2-SMI::enterprises.2636.1.1.1.2.9
```

Neighbor down:

```
cartman-re0.juniper.net [UDP: [172.19.110.171]:62313]: Trap , DISMAN-EVENT-
MIB::sysUpTimeInstance = Timeticks: (87321153) 10 days, 2:33:31.53,
SNMPv2-MIB::snmpTrapOID.0 = OID:
SNMPv2-SMI::mib-2.15.7.2,
SNMPv2-SMI::mib-2.15.3.1.14.10.200.7.2 = Hex-STRING: 06 07 ,
SNMPv2-SMI::mib-2.15.3.1.2.10.200.7.2 = INTEGER: 1,
SNMPv2-MIB::snmpTrapEnterprise.0 = OID:
SNMPv2-SMI::enterprises.2636.1.1.1.2.9
```

SNMP Polling for BGP

Junos supports a wide array of SNMP objects which can be polled to monitor the operation of BGP, many more than could possibly be covered in this book, so a few commonly used OIDs, including peer-state and prefix counters, are reviewed.

All Juniper BGP information is contained with the `jnxBgpM2` MIB, which has an OID of 1.3.6.1.4.1.2636.5.1.1. Peer-states are contained within the `jnxBgpM2PeerState` MIB (1.3.6.1.4.1.2636.5.1.1.2.1.1.2). The same state mapping described before apply here as well, so a value of 6 indicates that the peer is established:

```
nms-1> snmpwalk -v2c -c tr4pp15t dunkel 1.3.6.1.4.1.2636.5.1.1.2.1.1.2
SNMPv2-SMI::enterprises.2636.5.1.1.2.1.1.2.0.1.10.200.7.1.1.10.200.7.2 = INTEGER: 6
```

To gain useful information on each peer, you must first get the peer's SNMP index. The router creates a BGP peer index table, and this

information is accessible through the jnxBgpM2PeerIndex MIB (1.3.6.1.4.1.2636.5.1.1.2.1.1.14). The format of the returned string is SNMPv2-SMI::enterprises.2636.5.1.1.2.1.1.14.0.1.[local address].[peer address] = index:

```
nms-1> snmpwalk -v2c -c tr4pp15t cartman 1.3.6.1.4.1.2636.5.1.1.2.1.1.14
SNMPv2-SMI::enterprises.2636.5.1.1.2.1.1.14.0.1.10.200.7.1.10.200.7.2 = Gauge32: 0
```

Based on this information, the BGP session from 10.200.7.1 (local) to 10.200.7.2 (peer) has an index of 0 which can be used when doing peer specific queries.

The jnxBgpM2PrefixInPrefixes, jnxBgpM2PrefixInPrefixesAccepted and jnxBgpM2PrefixInPrefixesRejected are commonly used, and they provide information on the total number prefixes received, total number of prefixes accepted, and total number of prefixes rejected. Their OIDs are 1.3.6.1.4.1.2636.5.1.1.2.6.2.1.7, 1.3.6.1.4.1.2636.5.1.1.2.6.2.1.8 and 1.3.6.1.4.1.2636.5.1.1.2.6.2.1.9, and use the following format: 1.3.6.1.4.1.2636.5.1.1.2.6.2.1.7.[index].

Total prefixes received:

```
nms-1> snmpwalk -v2c -c tr4pp15t cartman 1.3.6.1.4.1.2636.5.1.1.2.6.2.1.7.0
SNMPv2-SMI::enterprises.2636.5.1.1.2.6.2.1.7.0.1.1 = Gauge32: 7
```

Total prefixes accepted:

```
nms-1> snmpwalk -v2c -c tr4pp15t cartman 1.3.6.1.4.1.2636.5.1.1.2.6.2.1.8.0
SNMPv2-SMI::enterprises.2636.5.1.1.2.6.2.1.8.0.1.1 = Gauge32: 1
```

Total prefixes rejected:

```
> snmpwalk -v2c -c tr4pp15t cartman 1.3.6.1.4.1.2636.5.1.1.2.6.2.1.9.0
SNMPv2-SMI::enterprises.2636.5.1.1.2.6.2.1.9.0.1.1 = Gauge32: 6
```

Through SNMP, you can see that seven total routes are received, one of which is accepted and six of which have been rejected, from the peer with index 0 (10.200.7.2). Use the show bgp summary command to confirm:

```
ps@dunkel-re0> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table          Tot Paths  Act Paths Suppressed  History Damp State   Pending
inet.0         7          1          0          0          0          0
Peer          AS      InPkt   OutPkt   OutQ   Flaps Last Up/Dwn State|#Active/
Received/Damped...
10.200.7.2     1       130     123      0      11     53:55 1/7/0      0/0/0
```


BGP Operational Mode Commands

Most BGP operational mode commands display information about neighbors and the prefixes received, accepted, and rejected from those peers. The most frequently used is the `show bgp summary` command. It displays information about all BGP neighbors, their states, and the prefixes learned from their peers as shown here:

```
ps@dunkel-re0> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table          Tot Paths  Act Paths Suppressed  History Damp State   Pending
inet.0          4          1          0          0          0          0
Peer           AS      InPkt   OutPkt   OutQ     Flaps  Last Up/Dwn  State|#Active/
Received/Damped...
10.200.7.2      1        220     211      0        11    1:33:38 1/7/0
0/0/0
10.200.7.3      1          0         0         0         0          1:52 Active
```

Here, seven routes are learned from 10.200.7.2 and one has been accepted. If the session was not established, its state would be displayed in place of the number of prefixes active/received/dampened, as shown with our session to 10.200.7.3. Also shown is that both peers are in AS 1 and that the session to 10.200.7.2 has been up for 1:33:38 and the session to 10.200.7.3 has been down for nearly two minutes. Understanding the length of time a session has been up or down can help correlate the state change to network events or configurations.

The `show bgp neighbor` command provides additional information for each BGP peer. A particular peer can be specified with this command (for example, `show bgp neighbor 10.200.7.2`) to display information specific to that peer, omitting this option displays information for all peers:

```
ps@dunkel-re0> show bgp neighbor 10.200.7.2
Peer: 10.200.7.2+53667 AS 1 Local: 10.200.7.1+179 AS 1
Type: Internal State: Established Flags: <Sync>
Last State: OpenConfirm Last Event: RecvKeepAlive
Last Error: Cease
Options: <Preference LocalAddress LogUpDown PeerAS Refresh>
Local Address: 10.200.7.1 Holdtime: 90 Preference: 170
Number of flaps: 11
Last flap event: RecvNotify
Error: 'Hold Timer Expired Error' Sent: 1 Recv: 0
Error: 'Cease' Sent: 1 Recv: 10
Peer ID: 10.200.7.2 Local ID: 10.200.7.1 Active Holdtime: 90
Keepalive Interval: 30 Peer index: 0
BFD: disabled, down
```

```

NLRI advertised by peer: inet-unicast
NLRI for this session: inet-unicast
Peer supports Refresh capability (2)
Table inet.0 Bit: 10000
RIB State: BGP restart is complete
Send state: in sync
Active prefixes:          1
Received prefixes:        7
Suppressed due to damping: 0
Advertised prefixes:      5
Last traffic (seconds): Received 18 Sent 3 Checked 59
Input messages: Total 287 Updates 11 Refreshes 0 Octets 5730
Output messages: Total 278 Updates 0 Refreshes 0 Octets 5308
Output Queue[0]: 0

```

This output obviously contains a lot of information. Some of the information is related to the configuration, while other information is related to the state. The first few lines reflect the current and recent states of the session, and that the current state is `Established`, the last state was `OpenConfirm` (the preceding step to `Established` in the BGP finite state machine), and that the session's last error message was `Cease`.

You can also see a session of type `Internal`, which means it is an internal BGP (iBGP) session. The local-address is configured as `10.200.7.1` (which is the `lo0.0` address, a common convention for iBGP), the hold timer is set to 90, and the keepalive is set to 30 (the default values). Also, the peer index (the same one used when discussing SNMP) and the capabilities supported by this session are `inet-unicast` (other options include `inet-multicast` or `inet-vpn`).

Towards the end, the output from `show bgp neighbor` includes prefix related information and that seven prefixes have been received and that one is being used. There are no suppressed routes and five routes are advertised to our peer. Following that information, are the BGP control traffic statistics.

Other valuable BGP operational commands include `show route protocol bgp` which displays all routes learned through BGP, `show route advertising-protocol bgp [neighbor]` which displays routes advertised to a peer, and `show route receive-protocol bgp [neighbor]` which displays the routes received from a given peer. The following is output from the `show route protocol bgp` command:

```
ps@dunkel-re0> show route protocol bgp
```

```
inet.0: 16 destinations, 22 routes (16 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```

10.200.7.2/32      [BGP/170] 02:04:22, localpref 100, from 10.200.7.2
                  AS path: I
> via so-2/0/0.0
18.32.74.0/30     [BGP/170] 01:44:10, localpref 100, from 10.200.7.2
                  AS path: I
> via so-2/0/0.0
112.74.9.0/30     *[BGP/170] 02:04:22, localpref 100, from 10.200.7.2
                  AS path: I
> via so-2/0/0.0
172.19.110.0/23   [BGP/170] 02:04:22, localpref 100, from 10.200.7.2
                  AS path: I
> via so-2/0/0.0
181.21.0.0/22     [BGP/170] 02:04:22, localpref 100, from 10.200.7.2
                  AS path: I
> via so-2/0/0.0
192.168.0.0/16    [BGP/170] 02:04:22, localpref 100, from 10.200.7.2
                  AS path: I
> via so-2/0/0.0
199.0.65.0/24     [BGP/170] 02:04:22, localpref 100, from 10.200.7.2
                  AS path: I
> via so-2/0/0.0

```

Here again, the “+” or “*” character indicates that the route is being used. This information correlates with the output of `show bgp neighbor`, where seven routes were being received and using one (112.74.9.0/30).

Let’s see what is being advertised to our peer:

```
ps@dunkel-re0> show route advertising-protocol bgp 10.200.7.2
```

```

inet.0: 16 destinations, 22 routes (16 active, 0 holddown, 0 hidden)
  Prefix            Nexthop          MED      Lc1pref  AS path
* 4.18.99.0/24      Self              100       100      I
* 10.200.7.1/32     Self              100       100      I
* 10.200.7.3/32     Self              100       100      I
* 18.32.74.0/30     Self              100       100      I
* 172.19.110.0/23   Self              100       100      I

```

This output is consistent with the `show bgp neighbor` command in that five routes are being advertised to our peer at 10.200.7.2. It also shows that the routes have a local preference of 100 (the default) and that the AS path is I, meaning Internal, which means that the local AS is originating the route.

The `show route receive-protocol bgp [neighbor]` command displays the routes being received from our neighbor. You would expect to see seven routes based on the previous output of `show bgp neighbor`:

```
ps@dunkel-re0> show route receive-protocol bgp 10.200.7.2
```

```
inet.0: 16 destinations, 22 routes (16 active, 0 holddown, 0 hidden)
  Prefix                Nexthop          MED      Lclpref   AS path
  10.200.7.2/32          10.200.7.2          100       100       I
  18.32.74.0/30          10.200.7.2          100       100       I
  * 112.74.9.0/30        10.200.7.2          100       100       I
  172.19.110.0/23        10.200.7.2          100       100       I
  181.21.0.0/22          10.200.7.2          100       100       I
  192.168.0.0/16         10.200.7.2          100       100       I
  199.0.65.0/24          10.200.7.2          100       100       I
```

This makes sense and aligns not only with expectations based on the output from `show bgp neighbor`, but also based on the output from `show route protocol bgp`: received seven routes and using one (the route to 112.74.9.0/30).

Summary

Each protocol has its own set of logs and traps that are created for events specific to the protocol. Some of these logs and traps require configuration.

Whenever you monitor at Layer 3, it is extremely important to remember that an IP network is an interconnected system and that it is dependent on lower layers in the OSI model. Collecting system logs and SNMP traps is instrumental in allowing both technicians and NMS systems to quickly identify the root cause of problems.

Problems experienced at Layer 3 can easily be caused by problems at Layers 1 and 2. This reinforces the message that developing a troubleshooting strategy which begins by narrowing down the likely causes before you even issue a single command, following the OSI model, and effectively utilizing protocol instrumentation is key to quickly isolating and resolving network issues.

Chapter 7

Layer 3 Troubleshooting

<i>Outage Types</i>	<i>100</i>
<i>Troubleshooting Packet Loss</i>	<i>100</i>
<i>Troubleshooting Routing Loops</i>	<i>104</i>
<i>Troubleshooting Circuit Overutilization</i>	<i>106</i>
<i>Troubleshooting Route Oscillation</i>	<i>106</i>
<i>Troubleshooting Latency</i>	<i>110</i>
<i>Summary</i>	<i>113</i>

This chapter on Layer 3 troubleshooting attempts to convey a methodology and philosophy for troubleshooting an IP network, and thus returns to guidelines presented in Chapter 1, if only to establish a process that you can apply over and over again.

It is nearly impossible to describe every IP problem you may encounter, so instead, this chapter presents effective ways of approaching network problems that can lead to quick problem isolation and resolution.

Outage Types

Let's begin by categorizing the different types of IP outages:

- **Packet Loss:** Packet loss is the failure to deliver a packet to a destination and can be caused by many different problems. The most common culprits are a down circuit, lack of (correct) routing information, routing loops, and over-utilized circuits (with or without class-of-service). Security devices also frequently cause (intentional) dropped packets, but firewalls are outside the scope of this book.
- **Latency:** Latency is a delay in the delivery of packets, which can be caused by suboptimal routing or class-of-service queuing. *Jitter*, a related problem, is a variance in latency and can be problematic in voice and video over IP environments. Jitter is often caused by class-of-service (or lack thereof).

NOTE One important point is that nothing is more important when troubleshooting an IP network than a solid understanding of the protocols being used and how those protocols interact. There are not enough pages in this book to deep dive into RIP, IS-IS, OSPF and BGP, so the assumption here is that you are proficient in the protocols run on your network and how they interact.

Troubleshooting Packet Loss

Packet loss is often one of the easier network problems to address as it is nearly always caused by one of the following:

- Circuit and hardware failures (complete outage)
- Routing loop (complete outage)
- Circuit overutilization (partial outage)
- Route oscillation (partial outage)

To troubleshoot these types of outages let's walk through the steps to quickly identify the router, or routers, involved, identify the root-cause, and, wherever possible, resolve the problem.

Circuit Outages

A complete outage is sometimes the simplest type of outage to resolve. When troubleshooting a complete outage, isolation is the most important aspect, since the resolution likely involves a configuration change, hardware restart or swap, or a call to the Telco vendor.

The most useful tool for troubleshooting this type of outage is *traceroute*. Traceroute sends a series of packets towards a destination with an incrementing time-to-live (TTL) value starting at 1. When a router receives an IP packet, it is required to decrement the TTL. When the TTL reaches zero, the router must send an ICMP time-exceeded message. This ICMP message provides the sending router with the IP address of that particular hop.

Since this process is repeated by every hop in the path, the sending router learns the IP address of every hop in the path. This information can be invaluable to an operator attempting to identify the root cause of an outage.

NOTE A common misconception is that the last responding hop in a traceroute is the cause of the problem. If it responds, it means that your host has reachability to that router and that router has reachability back to your host.

The problem generally lies between the last responsive hop and the first non-responsive hop or *on* the first non-responsive hop. The main point is that using this single command, you can immediately discover where you need to focus your effort.

Consider the network shown in Figure 7-1:

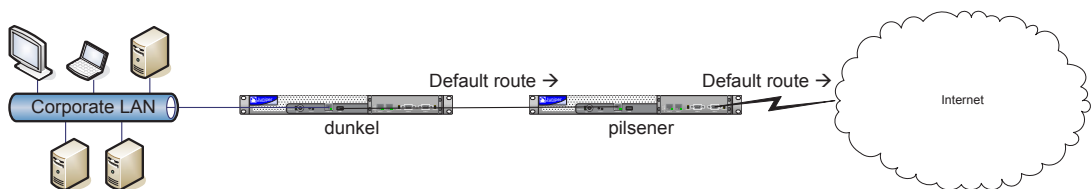


Figure 7-1 Simple Network Topology

This network is comprised of two routers within the site, one (dunkel, our old friend) aggregating our corporate LAN and one (pilsener, a favorite beverage) connecting to the service provider. Pilsener has a static default route with a next-hop to the service provider and is distributing this route into our site's IGP, OSPF. This provides dunkel with a default route with pilsener as the next-hop. The servers and hosts on the LAN have static default routes with dunkel as the next-hop.

Let's begin by viewing a traceroute to a destination on the Internet when all systems are working as expected:

```
server% traceroute 4.2.2.1
traceroute to 4.2.2.1 (4.2.2.1), 30 hops max, 40 byte packets
 1 18.32.75.1 (18.32.75.1)  2.617 ms  1.690 ms  2.851 ms   ← Dunkel
 2 18.32.74.6 (18.32.74.6)  3.386 ms  3.370 ms  5.570 ms   ← Pilsener
 3 4.10.33.2 (4.10.33.2)   13.513 ms 3.905 ms 5.060 ms ← Service provider hop 1
 4 4.1.18.21 (4.1.18.21)  3.778 ms 5.237 ms 5.413 ms   ← Service provider hop 2
 5 4.2.2.1 (4.2.2.1)      10.876 ms 12.568 ms 5.991 ms   ← Destination
```

The most common point of failure in this network is the link to the service provider as shown in Figure 7-2. Let's simulate this outage and repeat our traceroute.

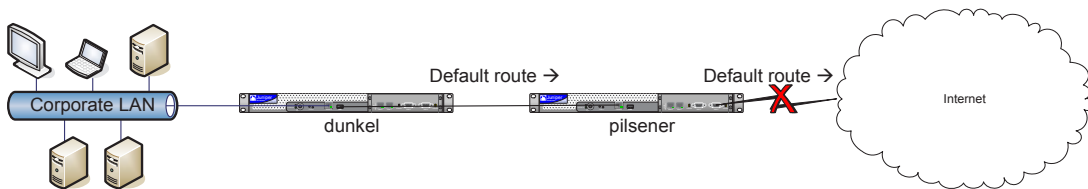


Figure 7-2 Service Provider Link Outage

The scenario shown in Figure 7-2 would yield the traceroute that follows:

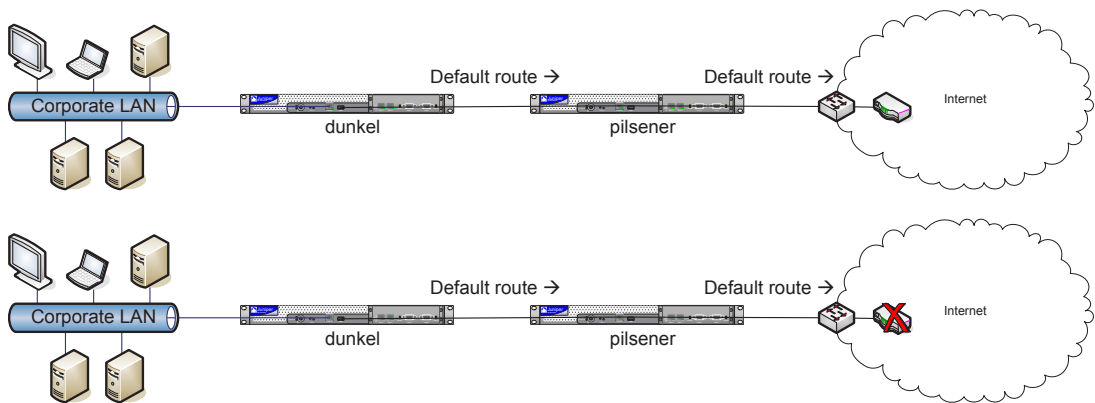
```
ps@dunkel> show route 4.2.2.1
ps@dunkel>
```

Because pilsener's static default route to the service provider disappears when the link goes down, it no longer distributes this route into OSPF, which means dunkel no longer has a route to 4.2.2.1. So dunkel responds with a destination host unreachable error message, which is indicated by the !H characters in the final line of our traceroute:

```
server% traceroute 4.2.2.1
traceroute to 4.2.2.1 (4.2.2.1), 30 hops max, 40 byte packets
 1 18.32.75.1 (18.32.75.1)  1.983 ms  2.440 ms  2.414 ms
 2 18.32.75.1 (18.32.75.1)  2.883 ms !H 4.136 ms !H 3.799 ms !H
```


Since dunkel is 18.32.75.1, it's the logical place to start investigating. In this case, router instrumentation on dunkel, and later on pilsener, would tell us that our link to the service provider has gone down (perhaps syslog or SNMP has already alerted us). At this point, start confirming that the hardware on the local side of the connection is good by reseating, resetting, and/or swapping it out and if the problems continue, contact the service provider for additional assistance.

There may also be situations, such as non-point-to-point Ethernet, in which the failure of the remote side of the connection may not bring the circuit down. Let's change the connection to the service provider to an Ethernet link that happens to go through a Layer 2 switch at our service provider before connecting to a router as shown in Figure 7-3. While Figure 7-4 illustrates a failure of the service provider router (or a failure of the connection from the service provider switch to the service provider router).



Figures 7-3 & 7-4 Ethernet Link and Outage to the Service Provider

The following shows how the traceroute might appear given this type of failure:

```
server % traceroute 4.2.2.1
traceroute to 4.2.2.1 (4.2.2.1), 30 hops max, 40 byte packets
 1 18.32.75.1 (18.32.75.1) 2.891 ms 0.594 ms 1.595 ms
 2 18.32.74.6 (18.32.74.6) 2.425 ms 2.544 ms 2.642 ms
 3 * * *
```

The trace has made it to Pilsener. This is because our service provider Ethernet connection terminates on a switch (which is up) rather than the service provider router (which is not), the connection itself stays up. This means Pilsener's static default route remains valid and it continues to distribute the default route into OSPF:

```
ps@pilsener> show route 4.2.2.1
```

```
inet.0: 11 destinations, 12 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
0.0.0.0/0          *[Static/5] 00:00:33
> to 192.168.14.10 via fe-0/0/1.0
```

Since the traceroute is “starring out” between Pilsener and the next-hop (the service provider), Pilsener is a good place to begin our investigation. The next steps would be to log into Pilsener to issue the `show route` command shown above and attempt to ping the remote side of the connection (192.168.14.10). And, when that fails, it’s time to contact the service provider.

This failure may be harder for an NMS system to catch because there is no link-down event. The only way a monitoring system could catch this type of failure is if it were also performing some type of probing for performance management and connectivity assurance. Many NMS systems have the ability to ping monitor different destinations for this purpose, such as Nagios and WhatsUp Gold.

```
ps@router-3> ping 192.168.14.10
PING 192.168.14.10 (192.168.14.10): 56 data bytes
^C
--- 192.168.14.10 ping statistics ---
5 packets transmitted, 0 packets received, 100% packet loss
```

Troubleshooting Routing Loops

Complete outages can be caused by routing loops. Routing loops occur when two routers (most frequently, directly connected routers) have selected each other as next-hops for a particular destination. This is most often seen in static route environments. A routing loop can also occur when there are route oscillations, but the loop may not be persistent in that situation. Route oscillations are discussed in this chapter.

Consider the following addition to our topology shown in Figure 7-5. The enterprise has expanded and needs an additional LAN aggregation router, Altbier.

Altbier is configured nearly identically to Dunkel. It is receiving a default route from OSPF by way of Pilsener. Pilsener has a static route

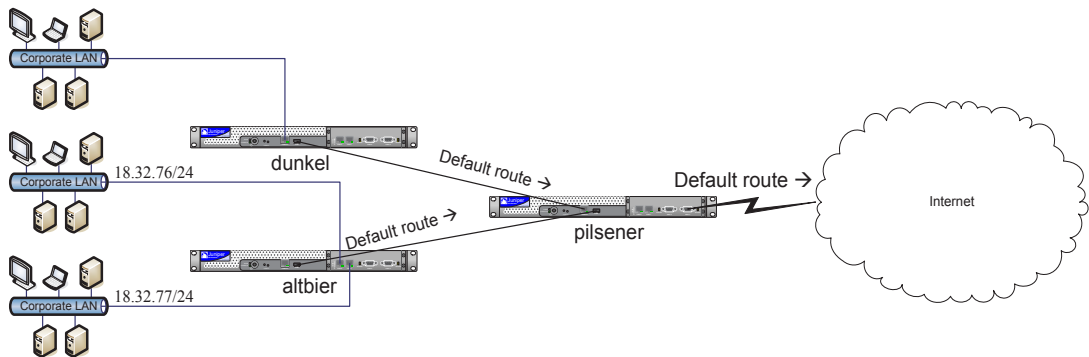


Figure 7-5 Addition of Second LAN Aggregating Router

for the 18.32.76.0/23 network with a next-hop to Altbier. Altbier has a static default route with a next-hop to Pilsener. A traceroute from a Dunkel connected host yields the following:

```

server% traceroute 18.32.76.7
traceroute to 18.32.76.7 (18.32.76.7), 30 hops max, 40 byte packets
 1 18.32.75.1 (18.32.75.1) 4.157 ms 3.735 ms 2.478 ms < dunkel
 2 18.32.74.6 (18.32.74.6) 8.913 ms 5.878 ms 5.518 ms < pilsener
 3 18.32.74.62 (18.32.74.62) 12.153 ms 7.266 ms 5.775 ms < altbier
 4 18.32.76.7 (18.32.76.7) 6.013 ms 4.567 ms 3.612 ms < destination
  
```

One outage that can lead to a routing loop in this scenario is if Altbier's link to the 18.32.76.0/24 network fails. This causes a routing loop because Pilsener has no knowledge of the outage on Altbier and continues to use the static route for the 18.32.76.0/23 network. When the packet reaches Altbier, it does not have a route to the 18.32.76.0/24 network because its interface to that network is down. The next best route it has is the default route towards Pilsener, which then sends the packet right back to Altbier because of its static route.

```

server% traceroute 18.32.76.7
traceroute to 18.32.76.7 (18.32.76.7), 30 hops max, 40 byte packets
 1 18.32.75.1 (18.32.75.1) 6.156 ms 2.181 ms 1.534 ms < dunkel
 2 18.32.74.6 (18.32.74.6) 9.631 ms 10.610 ms 3.273 ms < pilsener
 3 18.32.74.62 (18.32.74.62) 3.315 ms 3.728 ms 6.280 ms < altbier
 4 18.32.74.61 (18.32.74.61) 4.833 ms 8.704 ms 6.481 ms < pilsener
 5 18.32.74.62 (18.32.74.62) 7.148 ms 7.928 ms 3.979 ms < altbier
 6 18.32.74.61 (18.32.74.61) 3.779 ms 4.372 ms 3.427 ms < pilsener
 7 18.32.74.62 (18.32.74.62) 4.701 ms 4.005 ms 9.300 ms < altbier
 8 18.32.74.61 (18.32.74.61) 7.323 ms 7.616 ms 2.357 ms < pilsener
 9 18.32.74.62 (18.32.74.62) 3.373 ms 3.322 ms 10.979 ms < altbier
10 18.32.74.61 (18.32.74.61) 3.315 ms 10.498 ms 4.453 ms < pilsener
  
```

As is shown in this traceroute, the traffic is looping between Pilsener and Altbier. It is immediately clear where the problem could be. In a routing loop, the problem is nearly always on one of the routers on which the traffic is looping. For this example, both routers play a part in the problem. Pilsener is sending traffic to Altbier, even though Altbier's connection to that network is down and Altbier has a down interface to that network.

As a network operator, the first step would be to repair Altbier's connection to the 18.32.76.0/24 network. Later, it would be good to transition the configuration to a more dynamic architecture, such as using OSPF on Altbier to advertise the route as long as the connection to that network is up.

The nice part (if there is one) about complete outages is that they are complete. There is no ambiguity, inconsistency, or indeterminism. A traceroute alone should show you where the problem is and what device to log into to confirm it.

Troubleshooting Circuit Overutilization

Circuit overutilization can often cause packet loss, but with the annoyance that it is often intermittent. Depending on the types of traffic (consistent, bursty), volume of traffic (lightly overutilized or heavily overutilized), and class-of-service configuration, different packet flows may experience different levels of packet loss. The example outage in Chapter 2 provided an application of a consistent troubleshooting approach as well as the Fix Test in a circuit overutilization scenario.

Troubleshooting Route Oscillation

Route oscillation is one of the most difficult problems to identify and isolate quickly. Route oscillation can have different symptoms including complete outage, partial outage, and increased jitter. Route oscillation occurs when a route is repeatedly and quickly added and removed from the routing table. This can be caused by a rapidly transitioning circuit, mutual route redistribution, and a host of other reasons.

The best way to identify a route oscillation problem is to observe it when it happens. This can be accomplished in one of two ways. First, you can run a number of traceroutes, which should show the different paths used as the route oscillates. You can then issue the `show route` command on the router experiencing the oscillation. This should show a combination of different next-hops and/or that the route was learned from different protocols.

The previously described scenario where Altbier lost its connection to the 18.32.76.0/24 network could easily be extended to create a route oscillation problem. When the link is up, traffic flows as expected. However, when the link is down, traffic loops between Pilsener and Altbier. If this link were to rapidly transition, the user and operator would get inconsistent results between working and looping. In this case, there would be both a route oscillation *and* a routing loop.

Mutual route redistribution can also cause route oscillation. Mutual redistribution occurs when two protocols are both redistributed into one another. Unless done strictly and carefully, this type of architecture can cause network outages and make troubleshooting difficult. This technique is most often used during transitional periods (such as moving from EIGRP to OSPF). However, some networks rely on this method to solve other problems and for longer periods of time.

Let's use the transition example to show how mutual redistribution can cause route oscillation. Take the example of our current network, but change the relationship with the service provider to be a BGP peering relationship (rather than using a static route) as shown in Figure 7-6.

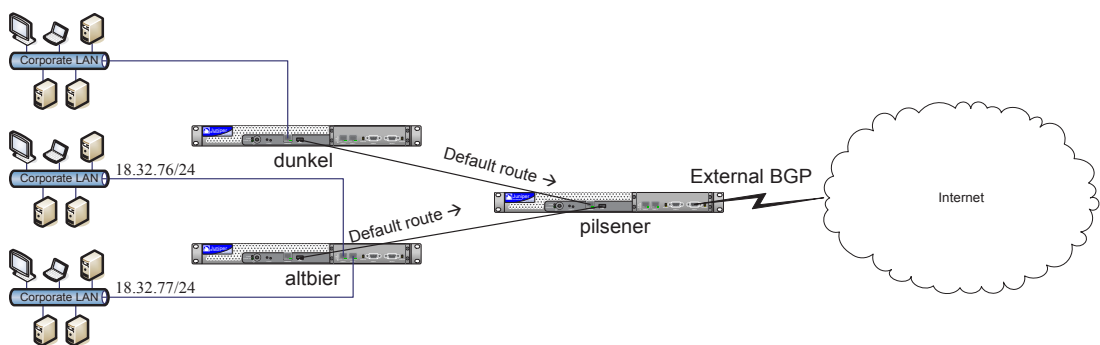


Figure 7-6 eBGP with Internet Service Provider

Over this session between Pilsener and our service provider, let's announce an aggregate route for the site's network (18.32.72.0/21) and receive the full Internet routing table, as such:

```
ps@pilsener-re0> show bgp summary
Groups: 1 Peers: 1 Down peers: 0
Table      Tot Paths  Act Paths Suppressed  History  Damp State  Pending
inet.0      10007      10007      0           0         0      0
Peer      AS      InPkt    OutPkt    OutQ    Flaps  Last Up/Dwn State|#Active/
Received/Accepted/Damped...
4.10.33.2  3356     10019     29        0        0      11:14
10007/10007/10007/0 0/0/0/0
```

You can reveal route-oscillation by testing *to* a destination on the Internet during a period of frequent route flapping for a particular prefix (4.0.0.0/8):

```
ps@pilsener-re0> ping 4.2.2.1
PING 4.2.2.1 (4.2.2.1): 56 data bytes
ping: sendto: No route to host
ping: sendto: No route to host
ping: sendto: No route to host
ping: sendto: No route to host
64 bytes from 4.2.2.1: icmp_seq=4 ttl=64 time=0.524 ms
64 bytes from 4.2.2.1: icmp_seq=5 ttl=64 time=0.697 ms
64 bytes from 4.2.2.1: icmp_seq=6 ttl=64 time=0.558 ms
64 bytes from 4.2.2.1: icmp_seq=7 ttl=64 time=0.579 ms
64 bytes from 4.2.2.1: icmp_seq=8 ttl=64 time=0.625 ms
64 bytes from 4.2.2.1: icmp_seq=9 ttl=64 time=0.582 ms
64 bytes from 4.2.2.1: icmp_seq=10 ttl=64 time=0.563 ms
ping: sendto: No route to host
ping: sendto: No route to host
ping: sendto: No route to host
ping: sendto: No route to host
ping: sendto: No route to host
ping: sendto: No route to host
ping: sendto: No route to host
64 bytes from 4.2.2.1: icmp_seq=18 ttl=64 time=0.613 ms
64 bytes from 4.2.2.1: icmp_seq=19 ttl=64 time=0.585 ms
64 bytes from 4.2.2.1: icmp_seq=20 ttl=64 time=0.642 ms
64 bytes from 4.2.2.1: icmp_seq=21 ttl=64 time=0.605 ms
64 bytes from 4.2.2.1: icmp_seq=22 ttl=64 time=0.562 ms
64 bytes from 4.2.2.1: icmp_seq=23 ttl=64 time=0.589 ms
64 bytes from 4.2.2.1: icmp_seq=24 ttl=64 time=0.607 ms
```

During this time, ping and traceroute instrumentation shows a gaining and losing of the BGP route to 4.0.0.0/8 and, as a result, experiences intermittent packet loss to anything in this network. As per this book's second opinion rule, use the show route command on the destination, 4.2.2.1 to observe the oscillation.

```
ps@pilsener-re0> show route 4.2.2.1
```

```
inet.0: 10017 destinations, 10017 routes (10017 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
4.0.0.0/8          *[BGP/170] 00:00:07, localpref 100
                   AS path: 3356 I
                   > to 4.10.33.2 via so-2/2/0.0
```

```
ps@pilsener-re0> show route 4.2.2.1
```

```
ps@pilsener-re0> show route 4.2.2.1
```

```
inet.0: 10017 destinations, 10017 routes (10017 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
4.0.0.0/8          *[BGP/170] 00:00:02, localpref 100
                   AS path: 3356 I
                   > to 4.10.33.2 via so-2/2/0.0
```

```
ps@pilsener-re0> traceroute 4.2.2.1
```

```
traceroute to 4.2.2.1 (4.2.2.1), 30 hops max, 40 byte packets
```

```
1  4.10.33.2 (4.10.33.2)  0.231 ms  0.411 ms  0.376 ms
```

```
2  4.1.18.21 (4.1.18.21)  0.193 ms  0.469 ms  0.288 ms
```

```
3  4.2.2.1 (4.2.2.1)  0.587 ms  0.612 ms  0.517 ms
```

```
ps@pilsener-re0> traceroute 4.2.2.1
```

```
traceroute to 4.2.2.1 (4.2.2.1), 30 hops max, 40 byte packets
```

```
traceroute: sendto: No route to host
```

```
1 traceroute: wrote 4.2.2.1 40 chars, ret=-1
```

In this case, it would be advisable to contact the service provider and inquire as to why the route is repeatedly being advertised and withdrawn.

Inconsistent packet loss can be difficult to quickly identify and resolve. However, it's usually caused by circuit overutilization, route oscillation, or a rapidly transitioning circuit, or some combination of all of these conditions.

Troubleshooting Latency

Latency is the amount of time it takes for a packet to get from a sender to the receiver. The root cause and isolation of a latency problem can be hard to identify. This is because latency can be inconsistent, can be limited to certain types of traffic, and may not be easily reproducible. An understanding of the topology of your network, the protocols used, the current state of your network, and any features enabled (such as class-of-service) can help you to resolve a latency problem (and loss, as discussed later).

Latency problems tend to cause the most trouble for real-time traffic applications, especially voice and video. Users that are reporting a problem may not even know that latency is the cause. The problem report may simply state that there are gaps in calls or artifacts in video. With normal traffic, such as HTTP, a latent packet isn't a big deal as the receiver simply waits until it receives the next packet and this is usually imperceptible to the user. However, a packet that is too latent in a voice or video stream means that packet is lost as far as the receiver is concerned.

The first step in troubleshooting a latency problem is to identify whether or not the traffic is taking the optimal path. This obviously requires an in-depth knowledge of the network as well as any current outages that would cause sub-optimal routing.

For the example network, traceroute shows that the traffic is taking the optimal path through our network, Dunkel, Pilsener, and then our service provider.

The next step then is to try to identify the network hop that is inducing the latency. The following traceroute shows that the hop between routers 2 and 3 is causing significant latency:

```
server% traceroute 4.2.2.1
traceroute to 4.2.2.1 (4.2.2.1), 30 hops max, 40 byte packets
 1 18.32.75.1 (18.32.75.1) 4.435 ms 3.117 ms 3.413 ms ←Dunkel
 2 18.32.74.6 (18.32.74.6) 4.935 ms 12.434 ms 2.826 ms ← Pilsner
 3 4.10.33.2 (4.10.33.2) 13.513 ms 3.905 ms 5.060 ms ← Service provider hop 1
 4 4.1.18.21 (4.1.18.21) 3.778 ms 5.237 ms 5.413 ms ← Service provider hop 2
 5 4.2.2.1 (4.2.2.1) 128.269 ms 137.346 ms 133.977 ms
```

The latency jumps from about 5 to 10 milliseconds of latency to well over a hundred at the hop from routers 2 to 3.

Now you know where to further investigate. At this point, the problem could either be ingress or egress queuing on router 2 or router 3. So the first thing needed is to check the link from router 2 to router 3, as it appears that there is significant packet queuing.

There should be high utilization on this link, since without high-utilization, packet queuing could not be happening, so check the interface on router 2 handling traffic to router 3 by issuing a `show interfaces` command:

```
ps@pilsener> show interfaces fe-0/0/1
Physical interface: fe-0/0/1, Enabled, Physical link is Up
  Interface index: 128, SNMP ifIndex: 61
  Link-level type: Ethernet, MTU: 1518, Speed: 100mbps, Loopback: Disabled,
  Source filtering: Disabled, Flow control: Enabled
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  CoS queues    : 4 supported, 4 maximum usable queues
  Current address: 00:90:69:6b:14:00, Hardware address: 00:90:69:6b:14:00
  Last flapped  : 2010-01-12 06:00:14 EST (2w5d 22:24 ago)
  Input rate    : 71441794 bps (46877 pps)
  Output rate   : 96542771 bps (63598 pps)
  Active alarms : None
  Active defects: None

Logical interface fe-0/0/1.0 (Index 67) (SNMP ifIndex 56)
  Description: Connection to service provider 1
  Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x8100.5 ] Encapsulation: ENET2
  Input packets : 0
  Output packets: 0
  Protocol inet, MTU: 1500
  Flags: None
  Addresses, Flags: Is-Preferred Is-Primary
    Destination: 4.10.33.0/30, Local: 4.10.33.1/30, Broadcast: 4.10.33.2

Logical interface fe-0/0/1.32767 (Index 68) (SNMP ifIndex 58)
  Flags: SNMP-Traps 0x4000 VLAN-Tag [ 0x0000.0 ] Encapsulation: ENET2
  Input packets : 0
  Output packets: 0
```

It appears that this link is probably overutilized. Verify this by checking the queue statistics. By default, Juniper Networks routers enable two queues. One is a network-control queue, which services the routing protocol and other control plane traffic. It is allocated 5% of the bandwidth and 5% of the buffer space. The other queue is a best-effort queue for all other traffic, which uses the remaining 95% of the bandwidth and 95% of the buffer space.

To determine whether or not there is significant queuing on this link, you need to use the `show interface queue` command. Because you know this traffic is in the best-effort queue, you can limit your command to this forwarding-class:

```
ps@pilsener> show interfaces queue fe-0/0/1 forwarding-class best-effort
Physical interface: fe-0/0/1, Enabled, Physical link is Up
  Interface index: 137, SNMP ifIndex: 32
Forwarding classes: 8 supported, 8 in use
Egress queues: 8 supported, 8 in use
Queue: 0, Forwarding classes: best-effort
  Queued:
    Packets      :          12196904          3179 pps
    Bytes        :      643712025753      9004563 bps
  Transmitted:
    Packets      :          12196904          3179 pps
    Bytes        :      643712025753      9004563 bps
  Tail-dropped packets :          0          0 pps
  RED-dropped packets  :          0          0 pps
    Low               :          0          0 pps
    Medium-low        :          0          0 pps
    Medium-high       :          0          0 pps
    High              :          0          0 pps
  RED-dropped bytes   :          0          0 bps
    Low               :          0          0 bps
    Medium-low        :          0          0 bps
    Medium-high       :          0          0 bps
    High              :          0          0 bps
```

There is queuing going on this interface. About nine megabits of traffic is being queued, which may explain our latency. There are three possible solutions to this type of queuing problem:

- Classify the voice, video, and other real-time traffic into an expedited forwarding (EF) queue with a higher priority than the best-effort queue.
- Investigate whether or not all of the egress traffic is valid.
- Upgrade the speed of the connection to your service provider.

Summary

While Layer 3 monitoring and troubleshooting is fundamentally different from Layer 1 and Layer 2, the same methodology and approach can be used to isolate and resolve Layer 3 problems. Using a consistent, logical approach and applying the Fix Test described in this book should allow you to quickly diagnose and implement supportable short-term fixes.

Whenever troubleshooting at Layer 3, remember that it is an interconnected system and that some routers in your network act based the actions of another router. Route-tagging and class-of-service markings are great examples of this. Nothing can replace experience with your network, the protocols used on it, and the way in which it operates under normal conditions. However, managing your approach to operating your network in conjunction with a sound understanding of the way in which protocols function and how Junos features and instrumentation can assist you make the task significantly easier.

What to Do Next & Where to Go

www.juniper.net/dayone

If you're reading a print version of this booklet, go here to download the PDF version which includes supplemental information in its Appendix. Also, find out what other Day One booklets are currently available.

www.juniper.net/junos

Everything you need for Junos adoption and education.

<http://forums.juniper.net/jnet>

The Juniper-sponsored J-Net Communities forum is dedicated to sharing information, best practices, and questions about Juniper products, technologies, and solutions. Register to participate in this free forum.

www.juniper.net/techpubs

All Juniper-developed product documentation is freely accessible at this site. Find what you need to know about the Junos operating system under each product line.

www.juniper.net/books

Juniper works with multiple book publishers to author and publish technical books on topics essential to network administrators. Check out this ever-expanding list of newly published books. See page iv for a special offer on *Junos High Availability*.

www.juniper.net/training/fasttrack

Take courses online, on location, or at one of the partner training centers around the world. The Juniper Network Technical Certification Program (JNTCP) allows you to earn certifications by demonstrating competence in configuration and troubleshooting of Juniper products. If you want the fast track to earning your certifications in enterprise routing, switching, or security use the available online courses, student guides, and lab guides.