

# **Advanced Juniper Networks Routing in the Enterprise**

---

8.a

***Student Guide***



1194 North Mathilda Avenue  
Sunnyvale, CA 94089  
USA  
408-745-2000  
[www.juniper.net](http://www.juniper.net)

Course Number: EDU-JUN-AJRE

Juniper Networks, the Juniper Networks logo, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. JUNOS and JUNOSe are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

*Advanced Juniper Networks Routing in the Enterprise Student Guide*, Revision 8.a

Copyright © 2006, Juniper Networks, Inc.

All rights reserved. Printed in USA.

Revision 8.a—December 2006

PLACE REVISION HISTORY HERE.

The information in this document is current as of the date listed above.

The information in this document has been carefully verified and is believed to be accurate for software Release 8.0R2. Juniper Networks assumes no responsibilities for any inaccuracies that may appear in this document. In no event will Juniper Networks be liable for direct, indirect, special, exemplary, incidental or consequential damages resulting from any defect or omission in this document, even if advised of the possibility of such damages.

Juniper Networks reserves the right to change, modify, transfer or otherwise revise this publication without notice.

#### YEAR 2000 NOTICE

Juniper Networks hardware and software products do not suffer from Year 2000 problems and hence are Year 2000 compliant. The JUNOS software has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

#### SOFTWARE LICENSE

The terms and conditions for using this software are described in the software license contained in the acknowledgment to your purchase order or, to the extent applicable, to any reseller agreement or end-user purchase agreement executed between you and Juniper Networks. By using this software, you indicate that you understand and agree to be bound by those terms and conditions.

Generally speaking, the software license restricts the manner in which you are permitted to use the software and may contain prohibitions against certain uses. The software license may state conditions under which the license is automatically terminated. You should consult the license for further details.



# Contents

---

<b>Chapter 1:</b>	<b>Course Introduction</b>	<b>1-1</b>
<b>Chapter 2:</b>	<b>JUNOS Policy</b>	<b>2-1</b>
	Policy Language and Policy Evaluation Process Overview	2-3
	Routing Policy	2-11
	Firewall Policy	2-34
	Unicast Reverse-Path Forwarding Checks	2-65
	Lab 1: Policy Configuration	2-71
<b>Chapter 3:</b>	<b>BGP</b>	<b>3-1</b>
	BGP Overview	3-3
	IBGP Implementation	3-14
	EBGP Implementation	3-22
	BGP-IGP Interaction	3-30
<b>Chapter 4:</b>	<b>Enterprise Routing Policies</b>	<b>4-1</b>
	Enterprise BGP Deployment	4-3
	Case Study: Primary/Secondary Routing Policy	4-26
	Lab 2: BGP Configuration	4-75
<b>Chapter 5:</b>	<b>Transitioning Between IGPs</b>	<b>5-1</b>
	Transition Overview	5-3
	Overlay Transition	5-14
	Route Redistribution Transition	5-20
	Integrated Transition	5-26
	Lab 3: IGP Transition	5-34
<b>Chapter 6:</b>	<b>JUNOS Services Overview</b>	<b>6-1</b>
	JUNOS Services Overview	6-3
	Layer 2 Services Configuration	6-7
	Layer 3 Services Overview	6-17
	Lab 4: Layer 2 Services	6-34
<b>Chapter 7:</b>	<b>Stateful Firewall and Network Address Translation</b>	<b>7-1</b>
	Stateful Firewall and NAT Overview	7-3
	Applications	7-7
	Configuring Stateful Firewall Rules	7-16
	Configuring NAT Rules	7-19
	Implementing Stateful Firewall and NAT	7-29
	Monitoring Stateful Firewall and NAT	7-38
	Lab 5: Stateful Firewall and NAT	7-46

<b>Chapter 8:</b>	<b>IPSec VPNs</b>	<b>8-1</b>
	IPSec VPN Overview	8-3
	IPSec VPN Configuration	8-7
	Implementation Considerations	8-18
	Monitoring	8-29
	Lab 6: IPSec VPNs	8-38
<b>Chapter 9:</b>	<b>Class of Service</b>	<b>9-1</b>
	CoS Overview	9-3
	Traffic Classification	9-11
	Traffic Queueing	9-18
	Traffic Scheduling	9-21
	Example	9-31
	Troubleshooting	9-36
	Lab 7: Class-of-Service Configuration	9-41
<b>Chapter 10:</b>	<b>Branch Office Connectivity</b>	<b>10-1</b>
	Overview of Connectivity Options	10-3
	Routing and Security Implications	10-7
	CoS Considerations	10-12
	Lab 8: Branch Office Connectivity (Optional)	10-20
<b>Appendix A:</b>	<b>Life of a Packet</b>	<b>A-1</b>
	Overview	A-3
	Example 1: Interface-Style Service Sets	A-5
	Example 2: Next-Hop-Style Service Sets	A-44
<b>Appendix B:</b>	<b>EIGRP to OSPF Migration Strategies</b>	<b>B-1</b>
<b>Appendix C:</b>	<b>Sample Routing Engine Filter</b>	<b>C-1</b>
<b>Appendix D:</b>	<b>Sample IPSec-over-GRE VPN Configuration</b>	<b>D-1</b>

## Course Overview

---

*Advanced Juniper Networks Routing in the Enterprise* (AJRE) is an instructor-led course designed to provide enterprise network engineers with the knowledge and skills necessary to use Juniper Networks routers to meet their networks' requirements. It covers advanced routing and services configurations of Juniper Networks J-series and M-series platforms, focussing specifically on advanced configurations commonly used in the enterprise environment.

### Objectives

After successfully completing this course, you should be able to:

- Explain how Juniper Networks routers evaluate policies.
- Implement a stateless firewall filter to protect the Routing Engine (RE).
- Implement both an inbound and outbound routing policy for traffic over multiple ISP connections using BGP.
- Monitor and troubleshoot BGP sessions and policy application.
- Successfully transition a network running a distance-vector IGP to use a link-state protocol.
- Configure Layer 2 services, including the Multilink Point-to-Point (MLPPP) protocol, Multilink Frame Relay (MLFR), and the Compressed Real-Time Transport Protocol (CRTP).
- Implement a stateful firewall to enforce a firewall policy.
- Implement Network Address Translation (NAT).
- With a virtual private network (VPN) design, create the VPNs between some combination of J-series and M-series routers.
- Given a class-of-service (CoS) design, configure appropriate CoS policies.
- List two branch-office connectivity solutions and the traffic considerations that apply to each.
- Given a list of technical constraints, implement certain branch-office connectivity solutions.
- Explain the interaction of various configuration elements with CoS configuration.

### Intended Audience

The primary audience for this course is network engineers who work in an enterprise environment.

### Course Level

This is an advanced-level course.

## Prerequisites

The following are the prerequisites for this course:

- The *Operating Juniper Networks Routers in the Enterprise* (OJRE) course or equivalent experience;
- Knowledge, familiarity, and comfort with the JUNOS CLI;
- Experience managing routers (not necessarily Juniper Networks) in an enterprise environment;
- Understanding of destination-based, hop-by-hop IP routing in a Classless Inter-Domain Routing (CIDR) environment; and
- Experience with IGPs in an enterprise environment.

## Course Agenda

---

### Day 1

- Chapter 1: Course Introduction
- Chapter 2: JUNOS Policy
- Chapter 3: BGP

### Day 2

- Chapter 4: Enterprise Routing Policies
- Chapter 5: Transitioning Between IGPs
- Chapter 6: JUNOS Services Overview

### Day 3

- Chapter 7: Stateful Firewall and Network Address Translation
- Chapter 8: IPSec VPNs

### Day 4

- Chapter 9: Class of Service
- Chapter 10: Branch Office Connectivity

## Document Conventions

---

### CLI and GUI Text

Frequently throughout this course, we refer to text that appears in a command-line interface (CLI) or a graphical user interface (GUI). To make the language of these documents easier to read, we distinguish GUI and CLI text from chapter text according to the following table.

Style	Description	Usage Example
Franklin Gothic	Normal text.	Most of what you read in the Lab Guide and Student Guide.
Courier New	Console text: <ul style="list-style-type: none"><li>• Screen captures</li><li>• Noncommand-related syntax</li></ul>	<code>commit complete</code> <code>Exiting configuration mode</code>
Century Gothic	GUI text elements: <ul style="list-style-type: none"><li>• Menu names</li><li>• Text field entry</li></ul>	Select File > Open, and then click Configuration.conf in the Filename text box.

### Input Text Versus Output Text

You will also frequently see cases where you must enter input text yourself. Often this will be shown in the context of where you must enter it. We use bold style to distinguish text that is input versus text that is simply displayed.

Style	Description	Usage Example
Normal CLI	No distinguishing variant.	<code>Physical interface:fxp0, Enabled</code>
Normal GUI		View configuration history by clicking Configuration > History.
<b>CLI Input</b> <b>GUI Input</b>	Text that you must enter.	lab@San_Jose> <b>show route</b> Select File > Save, and enter <b>config.ini</b> in the Filename field.

## Defined and Undefined Syntax Variables

Finally, this course distinguishes between regular text and syntax variables, and it also distinguishes between syntax variables where the value is already assigned (defined variables) and syntax variables where you must assign the value (undefined variables). Note that these styles can be combined with the input style as well.

Style	Description	Usage Example
<i>CLI Variable</i>	Text where variable value is already assigned.	<code>policy my-peers</code>
<i>GUI Variable</i>		Click on <i>my-peers</i> in the dialog.
<u><i>CLI Undefined</i></u>	Text where the variable's value is the user's discretion and text where the variable's value as shown in the lab guide might differ from the value the use must input.	Type <b>set policy</b> <b><u>policy-name</u></b> .
<u><i>GUI Undefined</i></u>		<b>ping 10.0.1.1</b> Select File > Save, and enter <b><u>filename</u></b> in the Filename field.

## Additional Information

---

### Education Services Offerings

You can obtain information on the latest Education Services offerings, course dates, and class locations from the World Wide Web by pointing your Web browser to:

<http://www.juniper.net/training/education/>.

### About This Publication

The *Advanced Juniper Networks Routing in the Enterprise Student Guide* was developed and tested using software version 8.0R2. Previous and later versions of software may behave differently so you should always consult the documentation and release notes for the version of code you are running before reporting errors.

This document is written and maintained by the Juniper Networks Education Services development team. Please send questions and suggestions for improvement to [training@juniper.net](mailto:training@juniper.net).

### Technical Publications

You can print technical manuals and release notes directly from the Internet in a variety of formats:

- Go to <http://www.juniper.net/techpubs/>.
- Locate the specific software or hardware release and title you need, and choose the format in which you want to view or print the document.

Documentation sets and CDs are available through your local Juniper Networks sales office or account representative.

### Juniper Networks Support

For technical support, contact Juniper Networks at <http://www.juniper.net/customers/support/>, or at 1-888-314-JTAC (within the United States) or 408-745-2121 (from outside the United States).





# **Advanced Juniper Networks Routing in the Enterprise**

## **Chapter 1: Course Introduction**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - Get to know one another
  - Identify the objectives, prerequisites, facilities, and materials used during this course
  - Identify additional Juniper Networks courses
  - Describe the Juniper Networks Technical Certification Program (JNTCP)



### This Chapter Discusses:

- Objectives and course content information;
- Additional Juniper Networks courses; and
- Juniper Networks Technical Certification Program.

## Introductions

- What is your name?
- Where do you work?
- What is your primary role in your organization?
- What kind of network experience do you have?
- What is the most important thing for you to learn in this training session?



### Introductions

This slide serves to break the ice by having you introduce yourself and state your reasons for attending the class.

## Course Contents

---

- Chapter 1: Course Introduction
- Chapter 2: JUNOS Policy
- Chapter 3: BGP
- Chapter 4: Enterprise Routing Policies
- Chapter 5: Transitioning Between IGPs
- Chapter 6: JUNOS Services Overview
- Chapter 7: Stateful Firewall and NAT
- Chapter 8: IPSec VPNs
- Chapter 9: Class of Service
- Chapter 10: Branch Office Connectivity



### Course Contents

This slide lists the topics we discuss in this course.

## Prerequisites

- The prerequisites for this course are the following:
  - OJRE or equivalent experience
  - Knowledge, familiarity, and comfort with the JUNOS CLI
  - Experience managing routers (not necessarily Juniper Networks) in an enterprise environment
  - Understanding of destination-based, hop-by-hop IP routing in a CIDR environment
  - Experience with IGPs in an enterprise environment

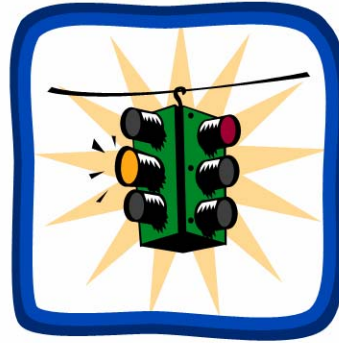


### Prerequisites

This slide lists the prerequisites for this course.

## Course Administration

- Course objectives
- Sign-in sheet
- Schedule
  - Class times
  - Breaks
  - Lunch
- Break and restroom facilities
- Communications
  - Telephones
  - Cellular phones and pagers
  - Internet access



### General Course Administration

This slide documents general aspects of classroom administration.

## Education Materials

- Available in class:

- Lecture material
- Lab guide
- Lab equipment

- Available outside of class:

- Online documentation at [www.juniper.net](http://www.juniper.net)
- Juniper Networks Technical Assistance Center (JTAC)

- Available through your account representative:

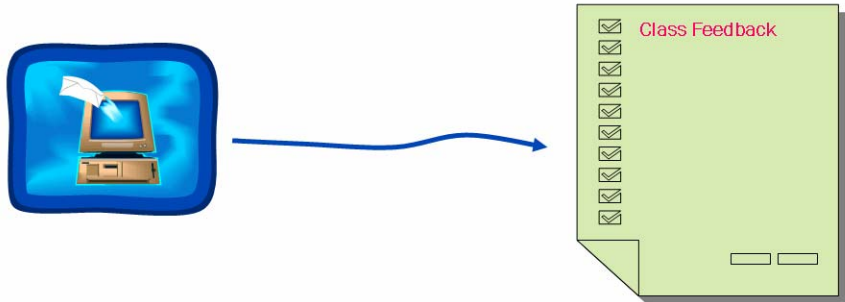
- Documentation CD
- Printed documentation



## Training and Study Materials

This slide describes several options for obtaining study and preparation materials.

## Satisfaction Feedback



- Please be sure to tell us how we did!
  - You will receive a survey to complete either at the end of class, or we will send it to you by e-mail within two weeks
- Completed surveys:
  - Help us serve you better
  - Ensure that you receive a certificate of completion

**Juniper your Net**

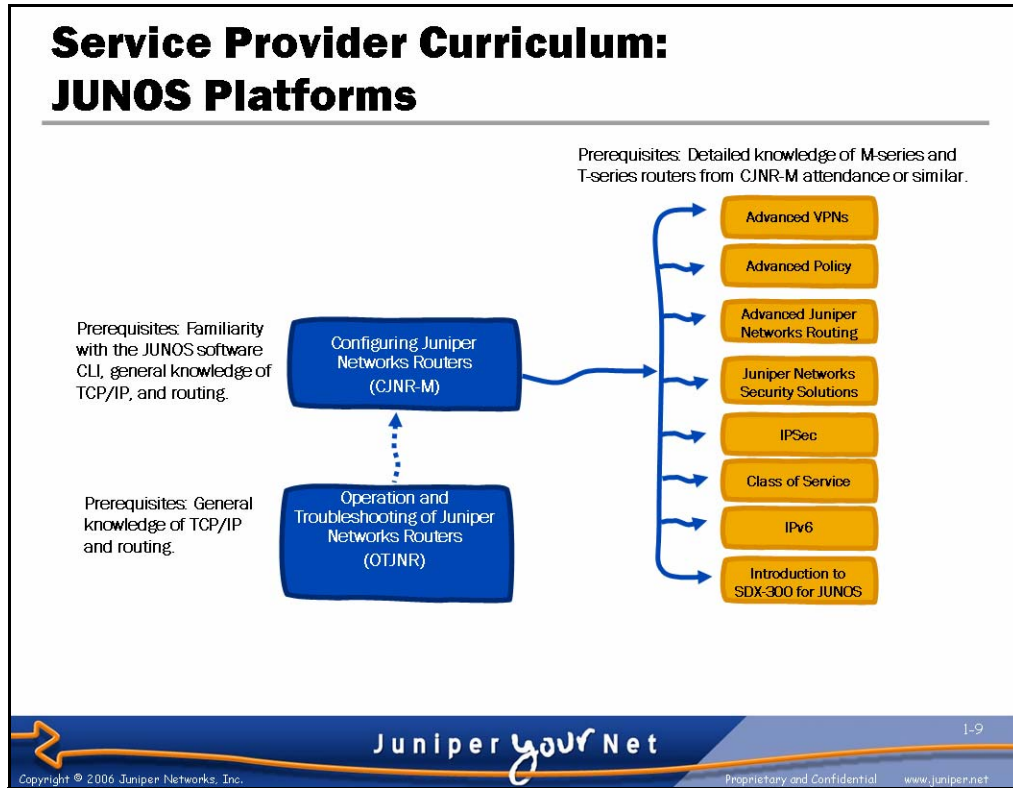
Copyright © 2006 Juniper Networks, Inc. Proprietary and Confidential www.juniper.net 1-8

### Satisfaction Feedback

Juniper Networks uses an electronic survey system to collect and analyze your comments and feedback. Depending on the class you are taking, please complete the survey at the end of the class, or be sure to look for an e-mail about two weeks from class completion that directs you to complete an online survey form (be sure to provide us with your current e-mail address).

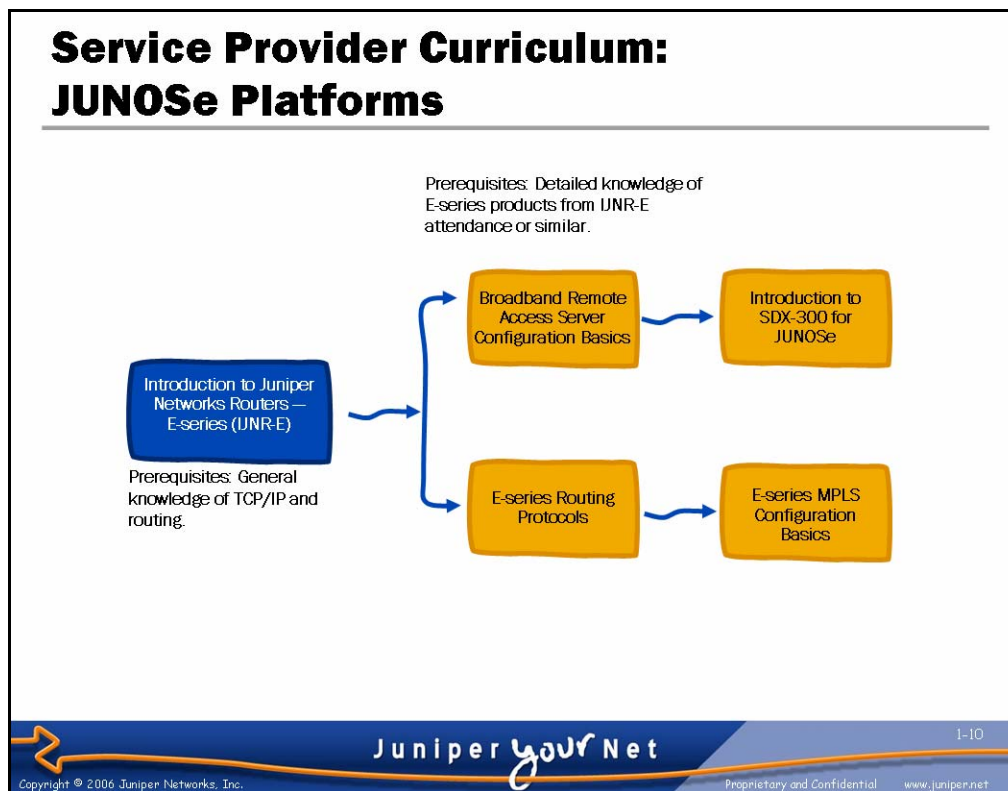
Submitting your feedback entitles you to a certificate of class completion. We thank you in advance for taking the time to help us improve our educational offerings.





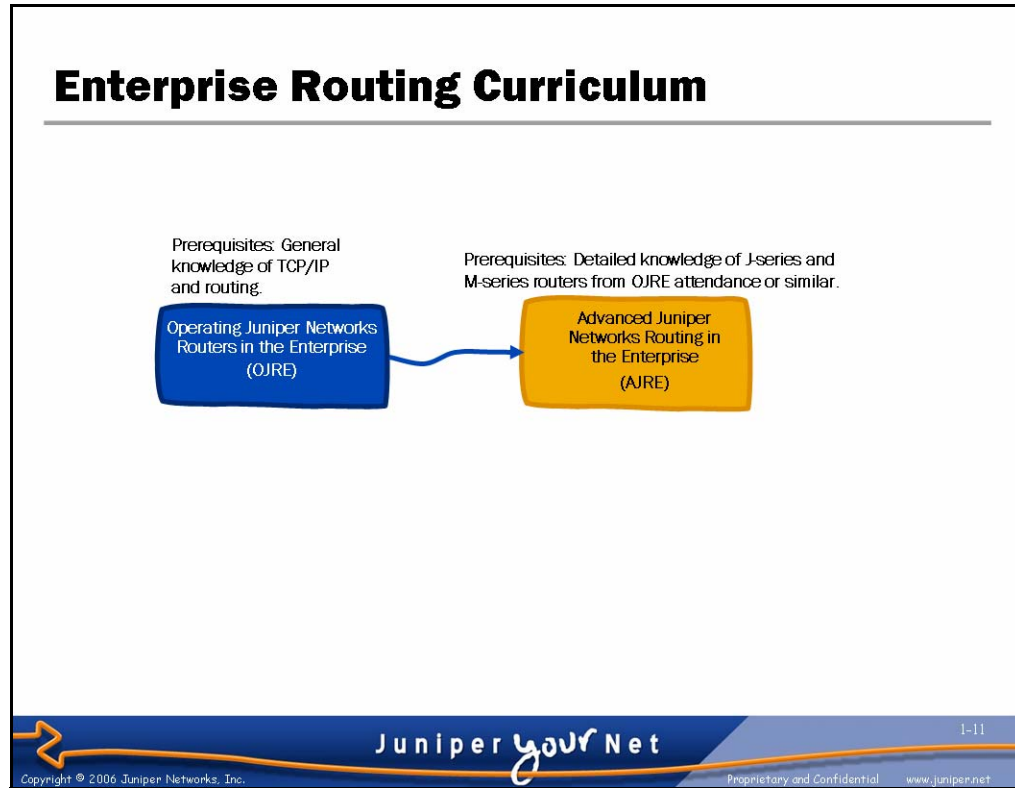
### Service Provider Curriculum: JUNOS Platforms

This slide displays the primary Education Services offerings that support Juniper Networks M-series and T-series technologies in a service provider environment.



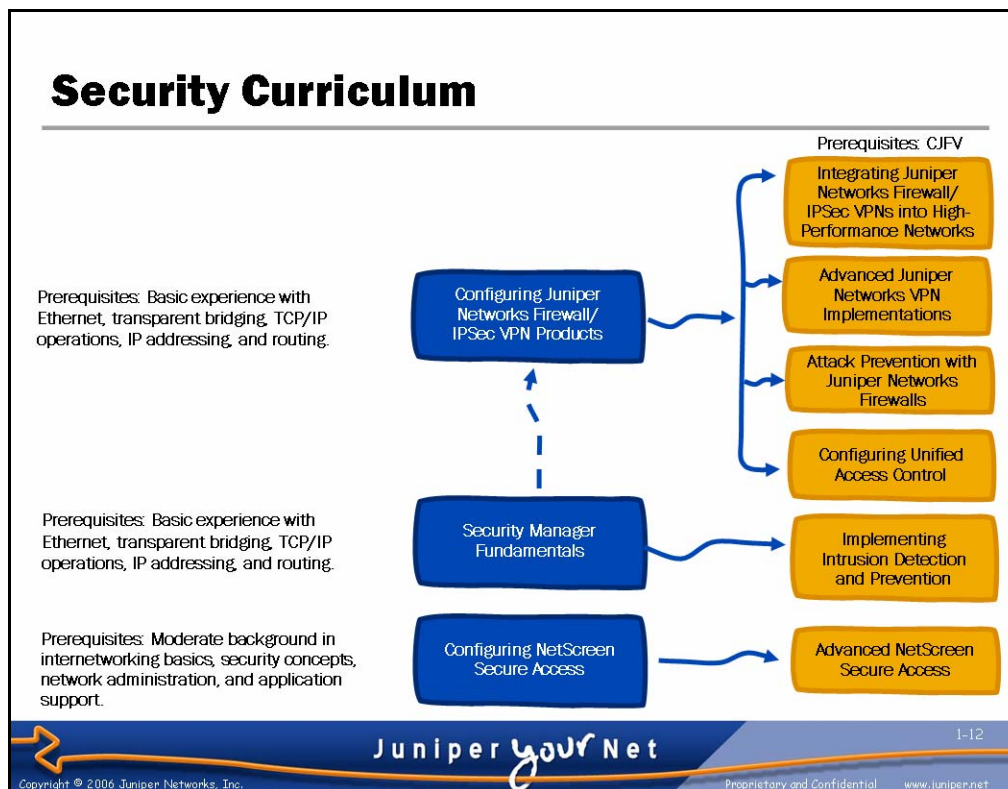
### Service Provider Curriculum: JUNOS<sup>e</sup> Platforms

This slide displays the primary Education Services offerings that support Juniper Networks E-series router technologies.



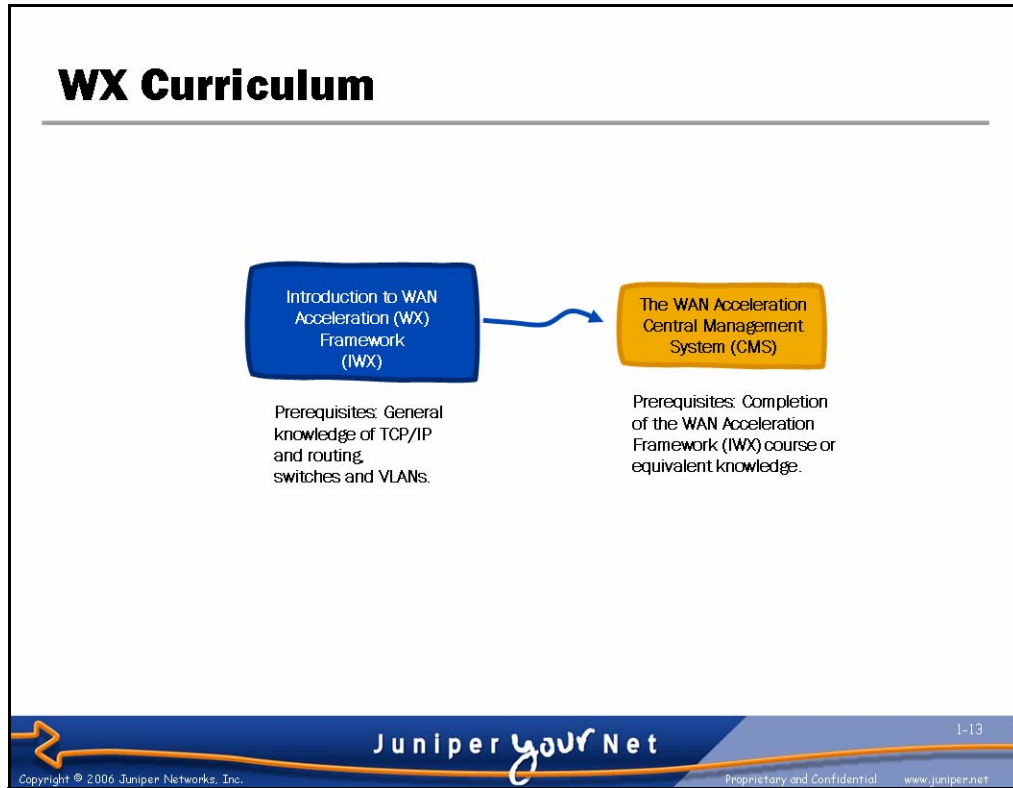
### Enterprise Routing Curriculum

This slide displays the primary Education Services offering that support Juniper Networks M-series and J-series technologies in an enterprise environment.



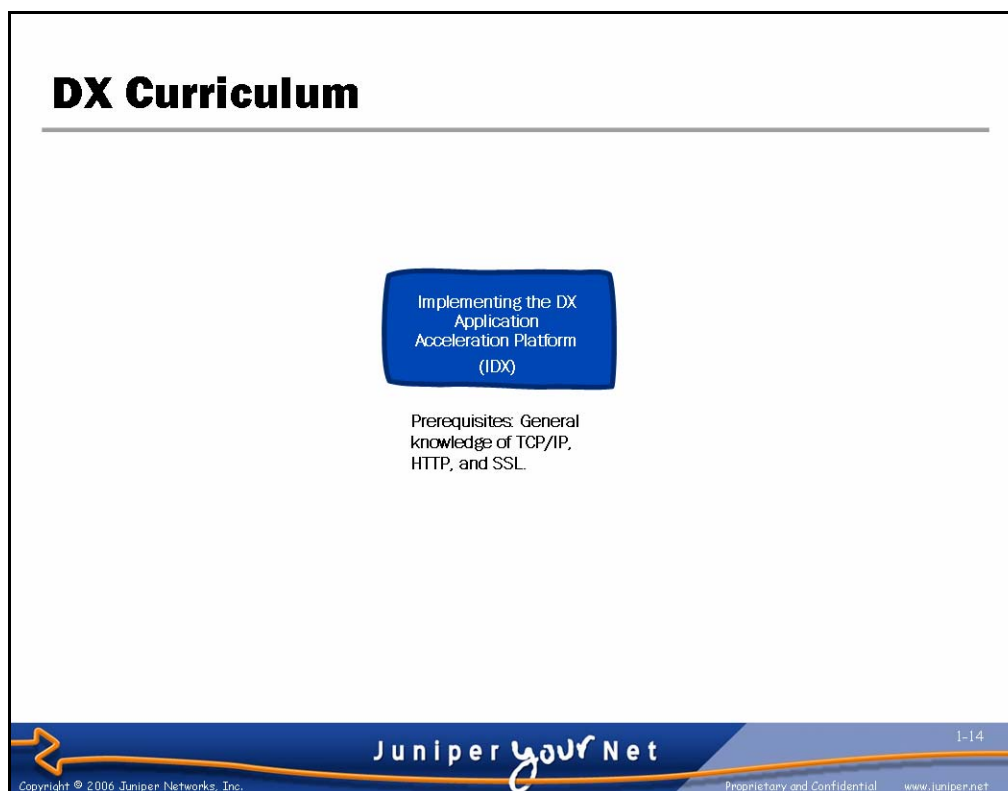
## Security Curriculum

This slide displays the primary Education Services offerings that support Juniper Networks security technologies.



### WX Curriculum

This slide displays the primary Education Services offerings that support Juniper Networks WX Framework technologies.



**DX Curriculum**

---

Implementing the DX  
Application  
Acceleration Platform  
(IDX)

Prerequisites: General  
knowledge of TCP/IP,  
HTTP, and SSL.

Juniper your Net

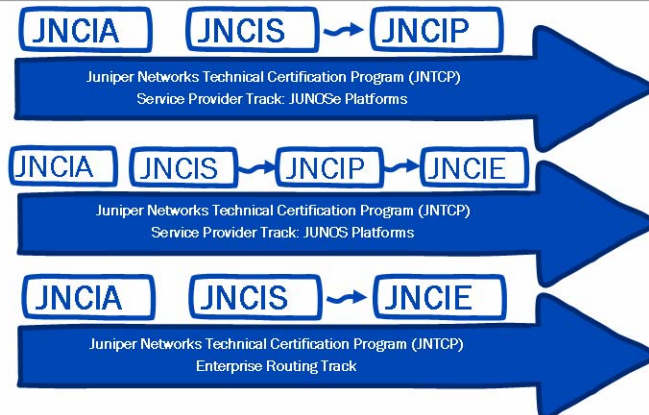
1-14

Copyright © 2006 Juniper Networks, Inc. Proprietary and Confidential www.juniper.net

## DX Curriculum

This slide displays the primary Education Services offerings that support Juniper Networks DX Application Acceleration Platform technologies.

## Technical Certification Programs: Routing Tracks



- Routing technical certification includes the following tracks:
  - Service provider track: JUNOS platforms
  - Service provider track: JUNOS platforms
  - Enterprise routing track
- Routing tracks consist of written and lab-based examination

### Technical Certification Programs: Routing Tracks

This slide outlines the current levels of technical certification offered by Juniper Networks.

## Technical Certification Programs: Security Tracks

Juniper Networks Technical Certification Program (JNTCP)  
Firewall/VPN Track

Juniper Networks Technical Certification Program (JNTCP)  
SSL/IDP Tracks

- Security technical certification includes the following tracks:
  - Firewall/VPN track
  - SSL/IDP tracks
- Security certification programs are written examination only at this time

Juniper your Net

Copyright © 2006 Juniper Networks, Inc. Proprietary and Confidential www.juniper.net 1-16

### Technical Certification Programs: Security Tracks

This slide outlines the current levels of technical certification offered by Juniper Networks.



## Technical Certification Programs: WX Track

---

JNCIA

Juniper Networks Technical Certification Program (JNTCP)  
WX Track

- The WX certification program is written examination only at this time



### Technical Certification Programs: WX Track

This slide outlines the current level of technical certification offered by Juniper Networks.

## Juniper Networks Certified Internet Associate (JNCIA)

- Computer-based, written exam
- Delivered at Prometric testing centers worldwide
- 60 questions, 60 minutes
- Passing Score: 70%
- \$125 USD
- Prerequisite certification: none
- Benefits provided to JNCIAs:
  - Certificate
  - Logo usage
  - Industry recognition
- Validates candidate's general knowledge of IP technologies, platform operating system, and hardware



### The JNCIA Certification

This slide details the JNCIA certification level.

## Juniper Networks Certified Internet Specialist (JNCIS)

- Computer-based, written exam
- Delivered at Prometric testing centers worldwide
- Prerequisite for the JNCIP lab exam
- 75 questions, 90 minutes
- Passing Score: 70%
- \$125 USD
- Prerequisite certification: none
- Benefits provided to JNCISs:
  - Certificate
  - Logo usage
  - Provides ability to take JNCIP exam
  - Industry recognition as an IP and routing platform specialist
- Validates candidate's advanced knowledge of platform operating system, hardware, and IP technologies



### The JNCIS Certification

This slide details the JNCIS certification level.

## Juniper Networks Certified Internet Professional (JNCIP)

- One-day, lab-based exam
- Tests candidate's configuration and design skills for essential technologies
- Testing centers: Sunnyvale, Amsterdam, Herndon, Westford, Remote
- Prerequisite for the JNCIE lab exam
- \$1,250 USD
- Prerequisite certification: JNCIS
- Benefits provided to JNCIPs:
  - Certificate
  - Logo usage
  - Provides ability to take JNCIE exam
  - Industry recognition as an IP and routing platform professional
- Validates candidate's practical platform configuration skills



### The JNCIP Certification

This slide details the JNCIP certification level.

## Juniper Networks Certified Internet Expert (JNCIE)

- One-day, lab-based exam
- Tests candidate's advanced configuration and design skills for essential and specialized technologies
- Testing centers: Sunnyvale, Amsterdam, Herndon, remote
- \$1,250 USD
- Prerequisite certification: JNCIP
- Currently only available in the M-series routers track
- Benefits provided to JNCIEs:
  - Crystal plaque and certificate
  - Logo usage
  - Worldwide recognition as an Internet Expert
- The most challenging and respected exam of its type in the industry

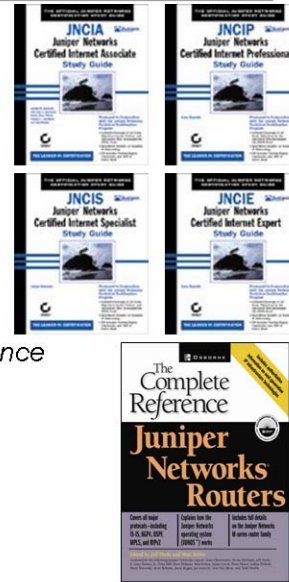


### The JNCIE Certification

This slide details the JNCIE certification level.

## Certification Preparation

- Training and study resources
  - JNTCP Web site
    - <http://www.juniper.net/certification>
  - Education Services training classes
    - <http://www.juniper.net/training>
  - Juniper Networks documents and white papers
    - <http://www.juniper.net/techpubs>
  - Sybex JNTCP preparation guides
    - Available at bookstores now
  - *Juniper Networks Routers: The Complete Reference*
    - Available at bookstores now
    - Covers M-series and T-series platforms
- Practical exams: lots of hands-on practice
  - On-the-job experience
  - Education Services training classes
  - Equipment access



### Prepping and Studying

This slide lists some options for those interested in prepping for Juniper Networks certification.

## Questions

---



### Any Questions?

If you have any questions or concerns about the class you are attending, we suggest that you voice them now so that your instructor can best address your needs during class.







# **Advanced Juniper Networks Routing in the Enterprise**

## **Chapter 2: JUNOS Policy**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - Identify terms, match conditions, and actions
  - Explain the way policies are evaluated
  - Write and apply a policy to control routing information
  - Write and apply a policy to filter traffic



### This Chapter Discusses:

- Terms, match conditions, and actions in a policy;
- The policy evaluation process;
- Writing and applying a policy to control routing information; and
- Writing and applying a policy to filter traffic.

## **Agenda: JUNOS Policy**

---

- Policy Language and Policy Evaluation Process Overview
- Routing Policy
- Firewall Policy
- Unicast Reverse-Path Forwarding Checks



### **Policy Language and Policy Evaluation Process Overview**

The slide lists the topics we cover in this chapter. We discuss the highlighted topic first.

## Policy Language Overview

- Policy language hierarchy:
  - A term is composed of a *from* statement and a *then* statement
  - A policy is composed of zero or more terms
  - Policy chains are composed of one or more policies



### Policy Language Hierarchy

There are many kinds of JUNOS policies. Some examples are firewall filters, routing policy statements, service filters, stateful firewall rules, and NAT rules. Although these policies serve different purposes and have different match and action conditions, they are all built using a common structure. Learning that common structure will enable you to better understand all JUNOS policies.

The fundamental building block of all policy evaluation is the term. A term contains a one or more match conditions and one or more actions. If all the match conditions are true, the action is taken. You use a policy to group together multiple terms and establish the order in which the router evaluates the terms. In some cases, you can form a policy chain to group multiple policies together and establish an order for their evaluation.

## Terms

- Contain *if...then* statements

- A *from* statement describes the match conditions
- A *then* statement describes the actions that should be taken if the *from* statement is matched
- Examples:

```
term discard-gre {
  from {
    protocol gre;
  }
  then discard;
}

term example-term {
  from {
    source-address {
      172.17.38.4/32;
    }
    destination-address {
      172.17.37.4/32;
      172.17.47.52/32;
      172.17.17.17/32;
    }
    protocol [ tcp udp ];
  }
  then accept;
}
```



### The Term

Terms are the basic building blocks of all JUNOS policy. They are essentially *if...then* statements. If all the match conditions specified in the *from* statement are true (or if no *from* statement is specified), then all the actions in the *then* statement are executed.

You give terms a name. The name has no effect on the evaluation of the term; rather, it gives you a way to provide a meaningful identifier that you can use when referring to the term.

When evaluating the *from* statement, the router performs the evaluation as a logical OR between arguments to a single match criterion and a logical AND between different match criteria. Put differently, for the *from* statement to be considered true, the item being evaluated must match at least one of the arguments to each match criterion given.

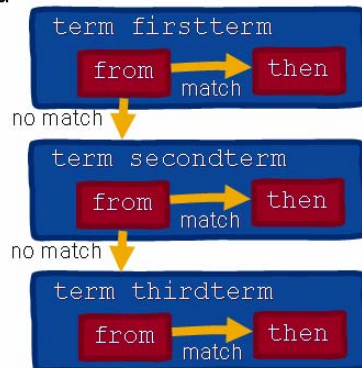
For example, consider the *example-term* from a firewall filter provided on the slide. To match this term, a packet must match the *source-address*, *destination-address*, and *protocol* conditions. To match the *source-address* condition, the packet's source address must be 172.17.38.4. However, to match the *destination-address* condition, the packet's destination address can be any of the listed addresses. Likewise, to match the *protocol* condition, the packet's protocol can be either of the listed protocols.

If all the conditions in the *from* statement are true, the router performs all the actions in the *then* statement. In this example, a single action is given: to *accept* the packet. So, for example, this term accepts a TCP packet from 172.17.38.4 to 172.17.37.4.

## Policy

- Contains terms

- Named with user-chosen identifiers
- Evaluated sequentially until a *terminating action* or end of policy is reached



### The Policy

Policies contain ordered groups of terms. You give policies a name, which you use to identify them when they are referenced elsewhere in the configuration.

When the router evaluates a policy, the router evaluates each term sequentially, in the order in which it appears in the policy. You can use the **insert** CLI command in configuration mode to modify the order in which terms appear in the policy. For example, at the `[edit firewall family inet filter filtername]` level, typing **insert term *secondterm* before term *firstterm*** places the term called *secondterm* immediately before the term called *firstterm*. You can use the **insert** command for all ordered data elements in the configuration, including terms within policies.

When the router evaluates a policy, it evaluates terms sequentially. If an item matches all the conditions in the *from* statement of a term, all the actions specified in the *then* statement of that term are executed. Provided that one of those actions is a *terminating action*, the evaluation of the policy stops. Some actions (such as the `next term` action) are not terminating actions, and the evaluation of the policy continues. In that case, later terms might overwrite attribute changes made by earlier terms because the router applies the actions for each matching term as the term is evaluated.

The actions that control the acceptance and rejection of items (`accept`, `reject`, `discard`, `service`, and `skip`) are terminating actions. Using these terminating actions results in a *first-match* policy evaluation because the router immediately executes the action and causes further evaluation of the policy.

## Policy Example

```
filter accept-vpn-to-Tokyo {
  term esp-ah-to-Tokyo {
    from {
      source-address {
        172.17.38.4/32;
      }
      destination-address {
        172.17.37.4/32;
      }
      protocol [ esp ah ];
    }
    then accept;
  }
  term ike-to-Tokyo {
    from {
      source-address {
        172.17.38.4/32;
      }
      destination-address {
        172.17.37.4/32;
      }
      protocol udp;
      port 500;
    }
    then accept;
  }
  term discard-others {
    then discard;
  }
}
```



### Policy Example

The sample firewall filter on the slide contains three terms. The term *esp-ah-to-Tokyo* matches packets with a source address of 172.17.38.4, a destination address of 172.17.37.4, and the Encapsulating Security Payload (ESP) protocol or the Authentication Header (AH) protocol (50 and 51, respectively). If a packet matches all three of these conditions, the packet is accepted and the evaluation of the policy ends.

If the packet does not match the first term, the router evaluates the term *ike-to-Tokyo*. Packets with a source address of 172.17.38.4, a destination address of 172.17.37.4, a protocol of UDP, and a UDP source or destination port of 500 match this term. If a packet matches all four of these conditions, the packet is accepted, and the evaluation of the policy ends.

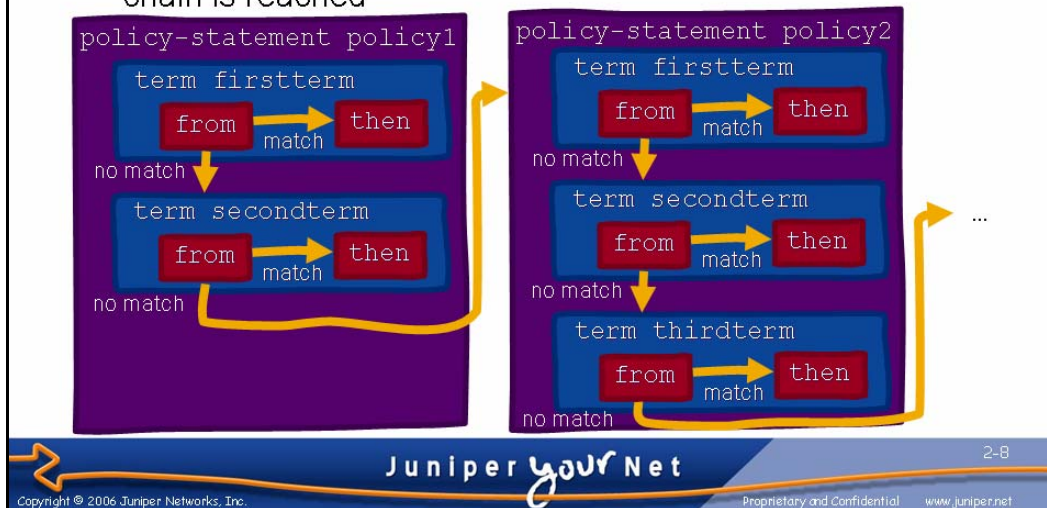
If the packet does not match either of the first two conditions, the router evaluates it against the final term (*discard-others*), and the packet is discarded.

Every policy has a default policy that the router uses if an item matches no terms in a policy. For firewall filters, the default policy is `then discard;`. Therefore, in this case, the final term is really unnecessary because the default policy causes the packet to be discarded anyway. We further discuss default policies later in this chapter.

## Policy Chains

- Contain policies

- Processed sequentially until a *terminating action* or end of chain is reached



### Policy Chains

In some cases, you can chain policies together. In these cases, the router sequentially processes through the terms of each policy in order until it reaches the end of the policy chain, or it reaches a terminating action, or it has no more terms to evaluate.

A policy chain allows you to write smaller policies that you reuse in a number of chains. For example, suppose you want to apply one BGP local preference to every route received from one provider's BGP sessions and a different BGP local preference to every route received from a different provider's BGP sessions, but you want to filter your own prefixes from the routes received from both providers. Instead of maintaining a single policy for each provider that both set the local preference and filters the routes, you can write three shorter policies: one that sets the correct local preference for the first provider, one that sets the correct local preference for the second provider, and one that filters your own prefixes. You then apply a policy chain to each session that calls the policy to set the correct local preference and calls the policy to filter your own prefixes.

This method has several advantages. It allows you to have shorter policies that are easier to read. You can also write policies in such a way that you usually must only change information in one location, making policy maintenance simpler. (For example, modifying the list of prefixes that should be filtered only requires updating a single policy.) Finally, it allows you some additional flexibility in controlling policy evaluation by providing a new command to control routing policy evaluation: the `next policy` action. This action causes the router to proceed to evaluating the next policy in the policy chain.



## Policy Chain Example

```

protocols {
  bgp {
    import [ localpref-80 accept-some-long-routes accept-short-routes ];
  }
}
policy-options {
  policy-statement localpref-80 {
    term set-localpref-80 {
      then {
        local-preference 80;
      }
    }
  }
  policy-statement accept-some-long-routes {
    term internal-long-prefixes {
      from {
        route-filter 10.216.216.0/22 prefix-length-range /25-/28;
      }
      then accept;
    }
  }
  policy-statement accept-short-routes {
    term 24-or-shorter {
      from {
        route-filter 0.0.0.0/0 prefix-length-range /1-/24;
      }
      then accept;
    }
    term 25-or-longer {
      then reject;
    }
  }
}

```

### Policy Chain Example

In the example on the slide, the router processes the three policies in the order in which they are listed in the policy chain that forms the import policy. (For readability reasons, they are listed in the same order in the `policy-options` section; however, the order in which the policies appear is irrelevant. The order in which the terms appear within policies *is* relevant because it controls the order of their evaluation, but the order listed in the policy chain determines the order in which the policies themselves are evaluated.)

In this case, the router starts by evaluating a route against the `localpref-80` policy. Because there is no `from` statement, all routes match, and the router takes the action specified in the `then` statement, setting the local preference to 80. Because this is not a terminating action and no more terms are in this policy, the router begins evaluating the next policy in the chain.

The router evaluates the route against the `accept-some-long-routes` policy next. If the route matches the `from` statement in the policy's only term, the router takes the action specified in the `then` statement, and it accepts the route, ending the evaluation of the policy chain. However, if the route does not match the `from` statement of this term, because this is the last term in this policy, the router moves on to evaluating the next policy in the chain.

*Continued on next page.*

### Policy Chain Example (contd.)

The router evaluates the route against the *accept-short-routes* policy next. If the route matches the *from* statement in the policy's *24-or-shorter* term, the router takes the action specified in the *then* statement and accepts the route, ending the evaluation of the policy chain. If the route does not match the *from* statement of this term, the router moves on to evaluating the next term of this policy.

The final term of this policy (*25-or-longer*) contains no *from* statement, so all routes match, and the router takes the action specified in the *then* statement, causing the route to be rejected.

For routes that are not accepted or rejected by the policy chain, the router applies the appropriate default policy when it reaches the end of the policy chain. Because an explicit default terminating action is at the end of this policy, the default policy is never reached. We discuss the default policy in more detail in a few pages.

## **Agenda: JUNOS Policy**

---

- Policy Language and Policy Evaluation Process Overview
- Routing Policy
- Firewall Policy
- Unicast Reverse-Path Forwarding Checks



### **Routing Policy**

The slide highlights the topic we discuss next.

## Why Use Routing Policy?

- Routing policy allows you to control the flow of routing information within the router by choosing to accept, reject, or modify attributes for:
  - Routes received via dynamic routing protocols
  - Routes sent via dynamic routing protocols
  - Routes installed in the forwarding table



### Routing Policy Uses

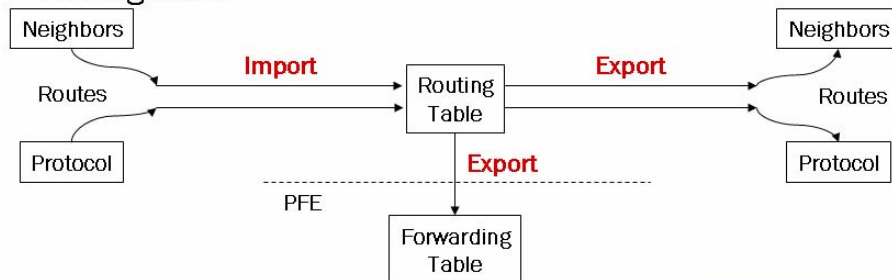
Routing policy allows you to control the flow of routing information within the router. It can be applied as information enters the routing table and as information leaves the routing table.

You can use routing policy to choose which routes you accept or reject from neighbors running dynamic routing protocols. You can also use routing policy to choose which routes you send to neighbors running dynamic routing protocols. Routing policy also allows you to modify attributes on routes as they enter or leave the routing table.

You can also use routing policy to control the flow of routing information into the forwarding table. This use allows you to control which routes are installed in the forwarding table and to control some of the attributes associated with those routes.

## Routing Policy Application (1 of 2)

- Routing policy controls the flow of routing information to and from the routing table
  - Import policies control the way routes are imported into the routing table
  - Export policies control the way routes are exported from the routing table



### Import and Export Policies

Policies that control the way routes are imported into the routing table are called *import policies*. The router applies these policies before the routes are placed in the routing table. Thus, an import policy can change the routes that are available in the routing table and affect the local route selection process.

Policies that control the way routes are exported from the routing table are called *export policies*. The router applies these policies as routes are exported from the routing table to dynamic routing protocols or the forwarding table. Only active routes are available for export from the routing table. Thus, while an export policy can choose which active routes to export and can modify attributes of those routes, it cannot cause inactive routes to be exported.

For example, suppose you have an OSPF route (preference 10) and a BGP route (preference 170) for the same prefix. An export policy can determine whether or not to send the active OSPF route and can modify attributes of the route as it is sent, but it cannot cause the inactive BGP route to be sent.

Export policies are applied as routes are exported from the routing table, so attribute changes do not affect the local routing table; rather, they are applied to the route as it is exported.

## Routing Policy Application (2 of 2)

- Routing policy can be applied at different levels:
  - Neighbor
  - Group
  - Protocol
- Only the most specific policy is applied (neighbor, then group, then protocol)



### Routing Policy Application Levels

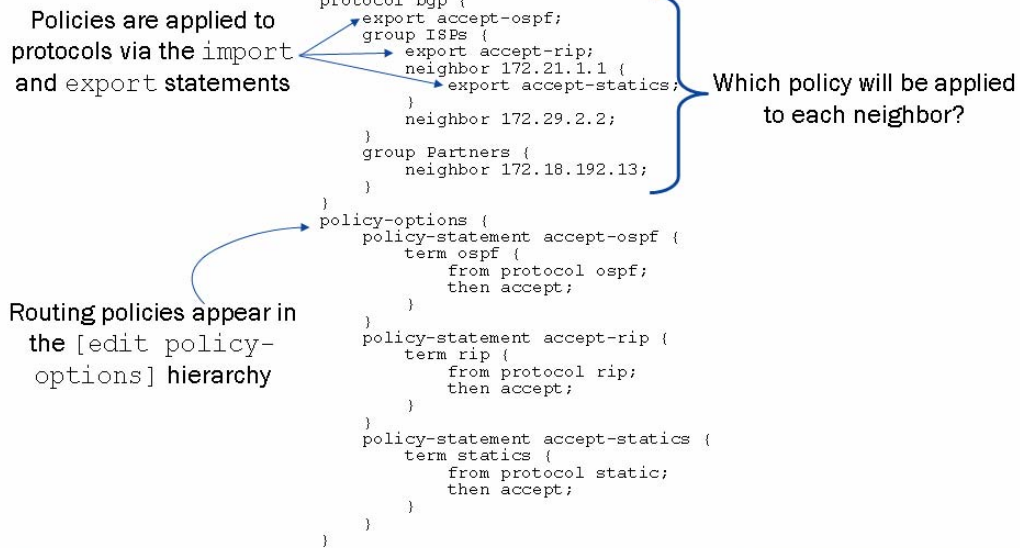
Depending on the routing protocol, you can apply import and export policies at multiple levels of the hierarchy. For example, you can apply import policies to BGP sessions at the neighbor, group, or protocol level of the configuration hierarchy.

Not all protocols allow policies to be configured at all these levels. For example, OSPF only allows protocol-level export and import policies because of the need to maintain a globally consistent link-state database. (This same requirement also limits the changes that can be made via an OSPF import or export policy.)

### Most Specific Policy

The router always applies the most specific (and only the most specific) import or export policy. Therefore, import or export policies applied at higher levels of the configuration hierarchy will be inherited at lower levels of the configuration if no policy is configured at that level. However, if a policy is configured at a lower level, the router only applies that policy.

## Policy Application Example



### Policy Application Example

The example on the slide illustrates the way the more specific portions of the configuration inherit import and export policies from the less specific portions of the configuration. It defines three policies—*accept-ospf*, *accept-rip*, and *accept-statics*—that accept all OSPF, RIP, and static routes, respectively. These policies are applied at three different levels of the configuration (protocol, group, and neighbor).

As the example shows, you configure a routing policy as a *policy-statement* in the [edit *policy-options*] hierarchy, and you apply it through an *export* or *import* statement.

Because the most specific policy that can be applied is at the neighbor level, the router uses the export policy *accept-statics* for the BGP neighbor 172.21.1.1. The policies at the protocol and group level are not applied to this neighbor because there is a neighbor-level policy, so the router does not send OSPF and RIP routes to this neighbor.

The neighbor 172.29.2.2 is part of the *ISPs* group, which has a group-level export policy. Because no export policy is defined at the neighbor level, the router uses the group-level export policy (*accept-rip*) for this neighbor. The policy defined at the protocol level is not used, so the router does not send OSPF routes to this neighbor.

The neighbor 172.18.192.13 is part of the *Partners* group, which does not have a group-level export policy. Because no export policy is defined at either the neighbor level or group level, the router uses the protocol-level export policy (*accept-ospf*) for this neighbor.

## Default Routing Policy

- Each protocol has a default import policy and a default export policy
  - Applied if no policy is specified
  - Applied at the end of any specified policy or policy chain

- Example:

```
export [ policyA policyB ];  
can be thought of as  
export [ policyA policyB default-policy ];
```

And:

```
export only-policy;  
can be thought of as  
export [ only-policy default-policy ];
```



### The Default Policy

Every protocol has a default import policy and a default export policy. The router applies these policies if no import or export policy is configured. In addition, if an export or import policy or policy chain is configured, the router applies the default policy at the end of the configured policy as if it were the last policy in a policy chain.

We examine each protocol's default policy on the following pages.



## Default Policies

Protocol	Import Policy	Export Policy
RIP	Accept all RIP routes from explicitly configured neighbors and import into <code>inet.0</code>	Reject everything
OSPF	Accept all OSPF routes and import into <code>inet.0</code>	Reject everything (protocol floods by default)
BGP	Accept all BGP routes and import into <code>inet.0</code> or <code>inet6.0</code>	Accept all active BGP routes



Juniper *your* Net

2-17

Copyright © 2006 Juniper Networks, Inc.

Proprietary and Confidential [www.juniper.net](http://www.juniper.net)

### Default Policies

The chart on the slide summarizes the default import and export policies for several common routing protocols.

The default policy for RIP is to import all routes learned from explicitly configured neighbors. The router ignores all routes learned from neighbors that are not explicitly configured. By default, no routes are exported to RIP neighbors, including routes received via RIP. Thus, to advertise *any* routes to RIP neighbors, you must configure an export policy. For RIP, you can configure import policies at the protocol level and neighbor level, while you can only configure export policies at the group level.

The default OSPF import policy is to import all OSPF routes. As a link-state protocol, OSPF maintains a consistent link-state database throughout each OSPF area by flooding link-state advertisements (LSAs). You cannot apply policy to affect the maintenance of the local link-state database or the flooding of LSAs. Additionally, you cannot apply policy that prevents internal (including interarea) routes from being installed in the routing table. (The nature of a link-state protocol is that it assumes that all routers have the same routing information for internal routes, which causes all routes to make consistent forwarding decisions. If you could block internal routes from entering the routing table, you could create routing loops or cause certain prefixes to become unreachable.) However, you can apply policy that blocks external routes.

*Continued on next page.*

## Default Policies (contd.)

The default OSPF export policy (which rejects everything) does not cause the router to stop flooding LSAs through the area. Rather, the router always floods LSAs throughout the OSPF area and that cannot be controlled by routing policy. The default export policy simply blocks additional routes from other sources from being advertised to OSPF neighbors. If you want to advertise other routes via OSPF, you must configure an explicit export policy.

Because link-state protocols rely on all routers having consistent link-state databases, you can only configure import and export policies at the protocol level.

BGP's default import policy is to accept all routes from BGP neighbors and install them in the routing table and to export all active BGP routes to all BGP neighbors.

Notice `inet.0` and `inet6.0` referenced on the slide. Different routing tables are available in each router, and they are used for different purposes. For example, MPLS routes are put into `inet.3`. The default IPv4 routing table is `inet.0`, and the default IPv6 routing table is `inet6.0`. We discuss the different routing tables again when we cover routing instances.

## Default Policy Application Example

```

protocol bgp {
  group Internal {
    neighbor 172.30.30.30;
  }
  group ISPs {
    export accept-rip;
    neighbor 172.21.1.1;
  }
  group Partners {
    export accept-only-statics;
    neighbor 172.18.192.13;
  }
}
policy-options {
  policy-statement accept-rip {
    term rip {
      from protocol rip;
      then accept;
    }
  }
  policy-statement accept-only-statics {
    term statics {
      from protocol static;
      then accept;
    }
    term reject-others {
      then reject;
    }
  }
}

```

Which routes will be exported to each neighbor?



### Default Policy Example

This slide helps illustrate the application of the default policy.

In this example, no explicit export policy is configured for the group *Internal*. Therefore, the router applies the default policy, and it advertises all active BGP-received routes to the BGP peer at 172.30.30.30.

An export policy is configured for the group *ISPs*. When you consider the default policy, you can think of this policy as being part of a policy chain like this: [ *accept-rip* default-policy ]. This export policy causes the router to advertise to the BGP peer at 172.21.1.1 all active RIP routes (because of the *accept-rip* policy) and all active BGP routes (because of the default policy).

An export policy is also configured for the group *Partners*. Again, when you consider the default policy, you can think of this policy as being part of a policy chain like this: [ *accept-only-statics* default-policy ]. This export policy causes the router to advertise to the BGP peer at 172.18.192.13 all active static routes (because of the *accept-only-statics* policy's term *statics*) and nothing else. The router denies all other routes (including all active BGP routes) because of the *accept-only-statics* policy's term *reject-others*. This term prevents any route from reaching the default policy.

## Common Selection Criteria

- Common match criteria for routing policy:
  - Prefix (`route-filter` or `prefix-list`)
  - Protocol (OSPF, static, BGP, etc.)
  - Routing protocol attributes
    - OSPF area ID, AS path, community
  - Next hop



### Common Selection Criteria

This slide shows some of the criteria you can use to select routes with *from* statements. You can select routes based on their prefix, protocol, some routing protocol attributes, or next hop. You can view the full list in the CLI interactive help and in the *JUNOS Policy Framework Configuration Guide*.

## Matching Prefixes

### ■ Prefix lists contain a list of prefixes

- Configured under [edit policy-options] hierarchy

Example:

```

policy-options {
  prefix-list rfc1918 {
    10.0.0.0/8;
    172.16.0.0/20;
    192.168.0.0/16;
  }
}

```

- Can be referenced in any firewall filter or routing policy term
- Routing policy supports exact-match (`prefix-list`) or optional match types and actions (`prefix-list-filter`):  
`prefix-list prefix-list-name;`  
`prefix-list-filter prefix-list-name match-type actions;`

### ■ Route filters match an individual route or groups of routes

- You can specify multiple route filters within a single term
- Not reusable—term-specific
- General syntax in the form of:  
`route-filter prefix/prefix-length match-type actions;`

## Matching Prefixes

You can select routes based on their prefix using a prefix list or a route filter.

Prefix lists are named lists of prefixes configured under the [edit policy-options] hierarchy. One of the advantages of prefix lists is that you can use them in multiple places. You can reference prefix lists in multiple terms in a single policy or in different policies. In addition, you can use prefix lists both for routing policies as well as firewall filters (but not stateful firewall rules). This reusability makes prefix lists attractive in some circumstances.

You can use prefix lists in two ways in the *from* statement of routing policies. When referenced in a `prefix-list` statement, routes only match if they *exactly* match one of the prefixes in the list. When referenced in a `prefix-list-filter` statement (added in JUNOS software Release 7.4), you can specify a match type of `exact`, `longer`, or `orlonger` to be applied to the listed prefixes. You can also specify an optional action to be taken if the filter matches. This action is executed immediately after the match occurs, and the *then* statement is not evaluated. We explain the match types in more detail on the following slides.

## Matching Route Filters

Route filters are lists of prefixes configured within a single routing policy term. Unlike prefix lists, they are not reusable but rather are specific to the term in which they are configured. They provide a few more match types for selecting prefixes. The following slides detail the available match types. Like the `prefix-list-filter` statement, you can specify an optional action to be taken if the `route-filter` statement matches. This action is executed immediately after the match occurs, and the *then* statement is not evaluated.

## Route Filter and Prefix List Match Types (1 of 2)

### ■ **exact:**

- Match the specified prefix and mask exactly
- No other routes are included

```
from route-filter 192.168.0.0/16 exact;
```

### ■ **orlonger:**

- Match the specified prefix and mask exactly
- Also match any routes that are subsets of the prefix and that have longer masks

```
from route-filter 192.168.0.0/16 orlonger;
```

### ■ **longer:**

- Do *not* match the specified prefix and mask exactly
- Match only the routes that are subsets of the prefix and that have longer masks

```
from route-filter 192.168.0.0/16 longer;
```



### **exact**

The match type **exact** means that only routes that match the given prefix exactly match the filter statement. For example, on the slide, only the prefix 192.168.0.0/16, and no other prefixes, match the filter statement.

### **orlonger**

The match type **orlonger** means that routes within the specified prefix with a prefix length greater than or equal to the given prefix length match the filter statement. So, the exact route 192.168.0.0/16 on the slide matches the statement. In addition, all routes that are subsets of 192.168.0.0/16 and that have prefix lengths between /17 and /32 also match. For example, the following prefixes match the statement: 192.168.0.0/16, 192.168.65.0/24, 192.168.24.89/32, 192.168.128/18, and 192.168.0.0/17. The following prefixes do not match the statement: 10.0.0.0/16, 192.167.0.0/17, 192.168.0.0/15, and 200.123.45.0/24.

### **longer**

The match type **longer** means that routes within the specified prefix with a prefix length greater than the given prefix length match the filter statement. (The **longer** and **orlonger** match types differ only in that the specified prefix itself matches the **orlonger** match type, but not the **longer** match type.) From the example on the slide, all routes that are subsets of 192.168.0.0/16 and have prefix lengths between /17 and /32 match, while 192.168.0.0/16 does not.

## Route Filter and Prefix List Match Types (2 of 2)

- **upto:**
  - Match the specified prefix and mask exactly
  - Also match any routes that are subsets of the specified prefix and that have a mask no longer than the second value specified

```
from route-filter 192.168.0.0/16 upto /24;
```
- **prefix-length-range:**
  - Match only routes that are subsets of the specified prefix and that have a mask between the two values specified (inclusive match)

```
from route-filter 192.168.0.0/16 prefix-length-range /20-/24;
```
- **through:**
  - Match the second specified prefix and mask exactly
  - Match the first specified prefix and mask exactly
  - Match all prefixes *directly* between the two prefixes

```
from route-filter 192.168.0.0/16 through 192.168.16.0/20;
```

### upto

The `upto` match type is similar to the `orlonger` match type, except that it provides an upper limit to the acceptable prefix length. The match type `upto` means that routes within the specified prefix with a prefix length greater than or equal to the given prefix's length, but less than or equal to the `upto` prefix length, match the filter statement. Thus, using the example on the slide, the exact route `192.168.0.0/16` matches the statement. All routes that are subsets of `192.168.0.0/16` and have prefix lengths between `/17` and `/24` (inclusive) also match. The following prefixes match the statement: `192.168.0.0/16`, `192.168.65.0/24`, `192.168.128.0/18`, and `192.168.0.0/17`. The following prefixes do not match the statement: `192.168.0.0/25`, `192.168.24.89/32`, `10.0.0.0/16`, `192.167.0.0/17`, and `200.123.45/24`.

*Continued on next page.*

### **prefix-length-range**

The `prefix-length-range` match type is similar to the `upto` match type, except that it provides both a lower and an upper limit to the acceptable prefix length. The match type `prefix-length-range` means that routes within the specified prefix with a prefix length greater than or equal to the first given prefix length, but less than or equal to the second prefix length, match the filter statement. Thus, using the example on the slide, all routes that are subsets of `192.168.0.0/16` and that have prefix lengths between `/20` and `/24` (inclusive) match. The following prefixes match the statement: `192.168.0.0/20`, `192.168.128.0/21`, and `192.168.64.0/24`. The following prefixes do not match the statement: `192.168.0.0/16`, `192.168.24.89/32`, `10.0.0.0/16`, `192.167.0.0/17`, `200.123.45/24`, and `192.168.128.0/18`.

### **through**

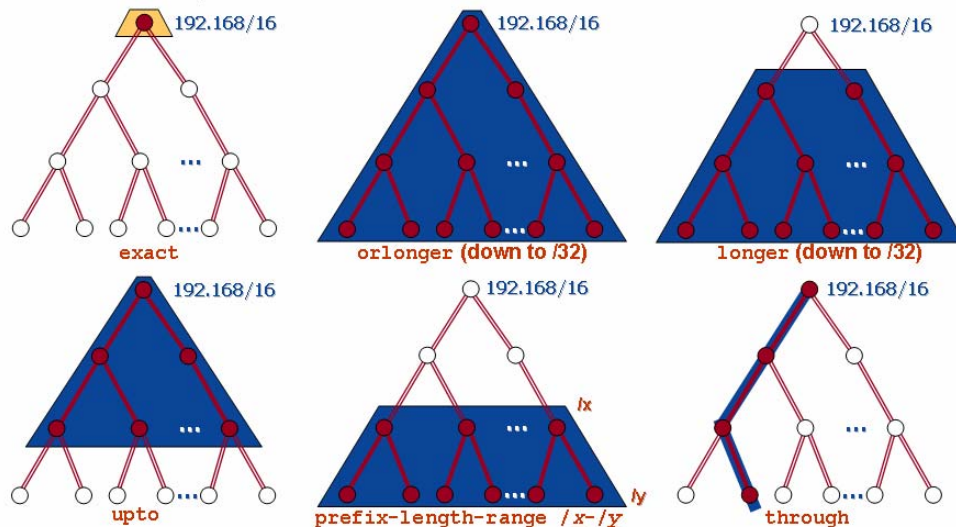
The `through` match type is very different than the other match types. The `through` match type matches all prefixes directly along the radix tree between two specified prefixes. The second prefix must be a subset of the first prefix. This match type matches the second (more specific) prefix and all prefixes of which it is a subset that are also equal to or more specific than the first prefix. In other words, this match type matches the second (more specific) prefix, the next most specific prefix of which it is a part, the next most specific prefix of which that one is a part, and so on, until reaching the first specified prefix. In the example on the slide, `192.168.16.0/20` matches. Because `192.168.16.0/20` is a subset of `192.168.0.0/19`, `192.168.0.0/18`, `192.168.0.0/17`, and `192.168.0.0/16` and all those prefixes are also equal to or more specific than `192.168.0.0/16`, they also match. No other prefixes match this example.

As another example, consider the statement from `route-filter` `192.168.0.0/16 through 192.168.232.0/21`. The following prefixes match that statement: `192.168.232.0/21`, `192.168.224.0/20`, `192.168.224.0/19`, `192.168.192.0/18`, `192.168.128.0/17`, and `192.168.0.0/16`.



## Match Type Summary

- Given a starting prefix of 192.168/16, what matches with each option?



### Route Filter Match Types

The diagram on the slide illustrates the different route filter match types.

There are some intricacies to the way that the router evaluates route filters when you use differing mask lengths. The router looks for a match between the configured prefix and mask in a route filter and a given route's prefix and mask. It looks for matches starting with the most specific route filter entry and ending with the least specific entry. Once it finds a match, it then evaluates the route against the route filter match type. If a match exists, the route matches the route filter. If the prefix does not match the route filter, the comparison fails even if the prefix might match a less specific entry in the route filter. We suggest you read the section on "How a Route List is Evaluated" in the *JUNOS Policy Framework Configuration Guide* before using route lists.

Also, you can use the **test policy** command to test the effectiveness of your policy (including route filters and prefix lists). When using this command, remember that the default policy of the **test policy** command is to accept all routes.

## Common Actions

### ■ Common actions in routing policy:

- Terminating actions
  - `accept`
  - `reject`
- Flow control
  - `next term`
  - `next policy`
- Modifying attributes
  - `as-path-prepend`
  - `community` (`add`, `delete`, `set`)
  - `load-balance`
  - `metric`
  - `preference`



### Common Actions

Some common routing policy actions include the terminating actions of `accept` and `reject`. These are called terminating actions because they cause the evaluation of the policy (and policy chain) to stop and the route to be accepted or rejected. The nonterminating equivalents of `default-action accept` and `default-action reject` do not cause policy evaluation to stop, but they do overrule the default policy's accept or reject determination.

Other common routing policy actions affect the flow of policy evaluation. The `next term` and `next policy` actions cause the router to evaluate the next term or next policy, respectively. A sample use of the `next policy` action follows:

```
policy-options {
  policy-statement set-localpref {
    term important-routes {
      from {
        prefix-list important-routes;
      }
      then {
        local-preference 200;
        next policy;
      }
    }
  }
}
```

*Continued on next page.*

**Common Actions (contd.)**

```

term less-important-routes {
    from {
        prefix-list less-important-routes;
    }
    then {
        local-preference 175;
        next policy;
    }
}
term all-others {
    then {
        local-preference 150;
    }
}
}

```

This policy itself does not accept or reject any routes; rather, it only sets local preference. The last term (`all-others`) sets the local preference for all routes to 150. However, the previous terms use the `next policy` statement to ensure that routes that match those terms are not evaluated against the remaining terms in the policy.

Other common actions are to prepend the AS path in BGP announcements, modify BGP communities, instruct the forwarding table to load-balance routes, modify an IGP metric or BGP MED (the `metric` statement), and modify route preference.

## Final Action

- One unnamed term allowed at the end of the policy:

```

policy-statement set-localpref {
  term important-routes {
    from {
      prefix-list important-routes;
    }
    then {
      local-preference 200;
      next policy;
    }
  }
  term less-important-routes {
    from {
      prefix-list less-important-routes;
    }
    then {
      local-preference 175;
      next policy;
    }
  }
  term all-others {
    then {
      local-preference 150;
    }
  }
}

```

→

```

policy-statement set-localpref {
  term important-routes {
    from {
      prefix-list important-routes;
    }
    then {
      local-preference 200;
      next policy;
    }
  }
  term less-important-routes {
    from {
      prefix-list less-important-routes;
    }
    then {
      local-preference 175;
      next policy;
    }
  }
  then {
    local-preference 150;
  }
}

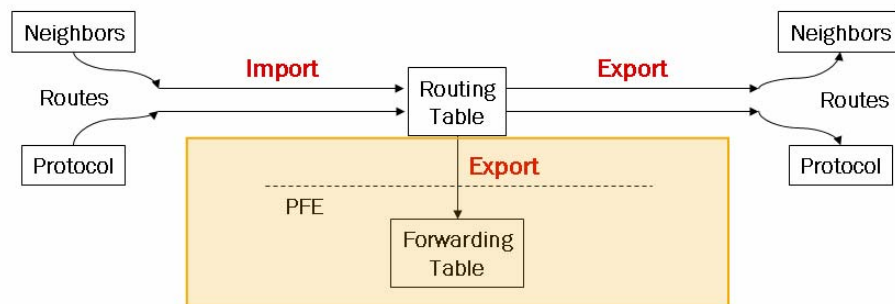
```

### Final Action

For routing policy only, you can define one unnamed term as a *final action*. The router always executes this term after all named terms, so it is only suitable for use in situations where the term should always be executed last, even when other terms are modified. This final action is very well suited for defining default actions.

## Forwarding Table Policy

- You can control how routes are exported to the forwarding table
  - Most frequently used for load-balancing equal-cost routes



### Forwarding Table Policy

You can also use policy to control routes as they are exported from the routing table to the forwarding table. Take care in applying this policy as differences between your routing and forwarding tables can cause routing loops or can cause certain prefixes to become unreachable. One of the common uses of forwarding table policy is to allow load balancing. By default, routes with multiple equal-cost next hops in the routing table are exported to the forwarding table with only a single next hop, which the router chooses randomly. You can modify that behavior with a forwarding table policy. Specifying the `load-balance per-packet` action causes all available next hops to be installed in the forwarding table. This installation causes traffic to be load-balanced in a per-flow fashion (over a maximum of 16 next hops). We show an example of this configuration on a later slide.

The default forwarding table export policy is to accept all routes.

## Sample Forwarding Table Policy

```
routing-options {  
    forwarding-table {  
        export load-balance-ospf;  
    }  
}  
policy-options {  
    policy-statement load-balance-ospf {  
        from protocol ospf;  
        then {  
            load-balance per-packet;  
        }  
    }  
}
```



### Sample Forwarding Table Policy

The policy on the slide sets all OSPF routes to be load-balanced. The default forwarding table policy accepts all routes.

## Routing Policy Example 1

```

protocol {
  rip {
    group v2peers {
      export local-statics;
    }
  }
}
policy-options {
  prefix-list local-routes {
    172.17.36.0/24;
    172.17.37.0/24;
    172.17.38.0/24;
    172.17.39.0/24;
  }
  policy-statement local-statics {
    term local-pl {
      from {
        protocol static;
        prefix-list local-routes;
      }
      then {
        metric 5;
        accept;
      }
    }
    term local-rf {
      from {
        protocol static;
        route-filter 172.17.36.0/22 prefix-length-range /24-/24;
      }
      then {
        metric 10;
        accept;
      }
    }
  }
}

```



### Example 1

In the example on the slide, the first term, (*local-pl*) accepts static routes to the four prefixes listed in the prefix list. The policy sets the metric to 5, and the router ceases further policy processing for those prefixes. The route filter in the second term matches exactly the same prefixes as listed in the prefix list; however, because these prefixes matched the first term, the router does not evaluate them against the second term. Because the route filter only matches those same prefixes, that term accepts no additional routes. The default policy for RIP is to reject all routes. Therefore, assuming that active static routes are in the routing table matching the four prefixes in the *local-routes* prefix list, this configuration results in exactly those four prefixes being advertised to the RIP neighbors in the *v2peers* group with a metric of 5.

## Routing Policy Example 2

```

protocol {
  bgp {
    export allow-local-routes;
  }
}
policy-options {
  prefix-list local-routes {
    172.17.36.0/22;
    172.17.40.0/22;
  }
  policy-statement allow-local-routes {
    term sub-aggregates {
      from {
        prefix-list-filter local-routes longer;
      }
      then {
        community set no-export;
        accept;
      }
    }
    term aggregates {
      from {
        prefix-list-filter local-routes exact accept;
      }
      then reject;
    }
    then reject;
  }
  community no-export members no-export;
}

```



### Example 2

In this example, all routes that are more specific and within the prefixes listed in the *local-routes* prefix list match the *prefix-list-filter* statement in the *sub-aggregates* term. The prefixes themselves do not match because the *longer* match type is used on the prefix list filter. All active routes in the routing table that match this prefix list filter—regardless of source—are accepted and sent with the *no-export* community.

The exact prefixes listed in the *local-routes* prefix list match the *prefix-list-filter* statement in the *aggregates* term. Only the exact prefixes match because the *exact* match type is used on the prefix list filter. Because of the *accept* action that is configured in the *prefix-list-filter* statement, the router accepts routes that match this statement and sends them without any attribute modification. The router ignores the actions in the *then* statement (including the *reject* action) because an action is attached to the prefix list filter.

Because the final action is *then reject*, the router rejects all other prefixes.



## Routing Policy Example 3

```

protocol {
  ospf {
    export some-prefixes;
  }
}
policy-options {
  prefix-list some-routes {
    172.17.36.0/22;
    172.17.38.0/23;
    172.17.39.0/24;
  }
  policy-statement some-prefixes {
    term from-a-prefixlist {
      from {
        prefix-list some-routes;
      }
      then {
        metric 100;
        external {
          type 1;
        }
        accept;
      }
    }
    term from-a-route-filter {
      from {
        route-filter 172.17.36.0/22 through 172.17.39.0/24;
      }
      then {
        metric 10;
        external {
          type 2;
        }
        accept;
      }
    }
  }
}

```



### Example 3

The first term (*from-a-prefixlist*) accepts the three prefixes listed in the prefix list on the slide. The policy causes the router to inject the routes into OSPF as Type 1 external routes with a metric of 100, and further policy processing for those prefixes will cease. The route filter in the second term matches exactly the same prefixes as listed in the prefix list; however, because these prefixes matched the first term, the router does not evaluate them against the second term. Because the route filter only matches those same prefixes, that term advertises no additional routes. The default policy for OSPF is to reject all routes. Therefore, assuming that active routes are in the routing table matching the three prefixes in the *some-routes* prefix list, this configuration results in exactly those three prefixes being advertised as OSPF Type 1 external routes with a metric of 100.

## **Agenda: JUNOS Policy**

---

- Policy Language and Policy Evaluation Process Overview
- Routing Policy
- Firewall Policy
- Unicast Reverse-Path Forwarding Checks



### **Firewall Policy**

The slide highlights the topic we discuss next.

## What Are Firewall Filters?

- Firewall filters allow you to:
  - Match packets in a stateless fashion (*from* statement)
  - Take actions on the selected packets (*then* statement)
- Always *compiled*
- Always checked in hardware (M-series routers)
- Easy to apply to local traffic to and from router



### Firewall Filters

Firewall filters follow the policy framework described earlier in this chapter. You use the *from* statement to list criteria used to select packets and the *then* statement to specify the actions that the router take on matching packets. Stateless firewall filters work on each individual packet in isolation to all others. Thus, unlike a stateful firewall, which tracks connections and allows you to specify an action to be taken on all packets within a flow, a stateless firewall filter has no concept of connections. The stateless nature of these filters can impact the way you write your firewall filters.

### Compiled ACLs

Other vendors' routers might process each packet through an access control list (ACL) sequentially, without optimization. Therefore, that method causes the router to spend more time processing each packet for each line of the ACL it checks prior to finding a match. With long ACLs or high-traffic environments, it might be necessary to construct ACLs to purposely cause as much traffic as possible to match as early as possible. This construction can be quite complex and still result in somewhat unpredictable processor utilization because the processor utilization varies not only on amount of traffic, but also the variety of traffic.

*Continued on next page.*

### ***Compiled ACLs (contd.)***

Other vendors sometimes try to alleviate this problem by allowing you to *compile* ACLs. When ACLs are compiled, the software converts each ACL into bit-pattern matches and uses this process to accomplish an efficient match. Processing packets through ACLs that are compiled takes a fairly consistent (and usually lower) amount of processing. Unlike some other vendors, the JUNOS software always *compiles* stateless firewall filters.

### **Hardware Processing**

Unlike some other vendors, M-series and T-series routers always perform ACL checks in hardware. This process results in a very efficient match. The J-series router performs ACL checks in the `fwdd` process; however, like all functions performed by the `fwdd` process, this function is designed to be processed efficiently. In either case, firewall filter processing still has the performance efficiencies that come from the way the software compiles firewall filters.

### **Filtering Local Traffic**

The JUNOS software makes it easy to apply firewall filters to *local* traffic, which is traffic destined to or originating from the router itself. You can filter all local traffic by applying a firewall filter to the `lo0` interface. If multiple routing instances are configured on the router, you filter local traffic on a per-routing-instance basis by applying a firewall filter to the `lo0` interface's unit assigned to each routing instance.

## Firewall Filter Application

- Firewall filters are applied to interfaces
  - Logical interfaces: matches input and output traffic on interface
  - 100: matches input and output traffic from the Routing Engine
- Firewall filter chains are allowed
  - Function the same as routing policy chains
- Default action is *discard* (if a firewall filter is configured)



### Filtering Traffic on Interfaces

Although you can use firewall filters to filter traffic at several points, their primary purpose is to filter traffic entering or exiting interfaces. You can apply them to all interfaces to filter traffic entering and exiting the interfaces. Additionally, you can apply them to the 100 logical interfaces to filter traffic destined for the router.

You apply firewall filters to interfaces in the [edit interfaces interfacename unit unitnumber family inet filter] hierarchy. To apply a single input or output filter, use the input filtername or output filtername configuration options. You can specify both input and output filters on the same interface.

### Firewall Filter Chains

You can configure the router to use multiple filters to filter traffic using the input-list or output-list configuration options in the [edit interfaces interfacename unit unitnumber family inet filter] hierarchy. When you use these options, you create a policy chain, and packets are processed through the filters in the same way as any other policy chain. As was discussed earlier, policy chains can provide several advantages (see “Policy Chains” on page 2-8).

*Continued on next page.*

## Default Action

The default action when a firewall filter is configured is `discard`. Therefore, if a packet fails to match any term within a firewall filter or chain of firewall filters, the router silently discards it.

Unlike routing policy, the default action is different when a firewall filter is configured than when no firewall filter is configured. If no firewall filter is configured, the default action is `accept`.

## Firewall Filter Application Example

```

interfaces {
  fe-0/0/0 {
    unit 0 {
      family inet {
        filter {
          input filter-in;
          output-list [ filter1 filter2 ];
        }
      }
    }
  }
}

firewall {
  family inet {
    filter filter-in {
      term block-some-packets {
        from {
          source-address {
            10.10.10.0/24;
          }
        }
        then {
          discard;
        }
      }
      term accept-others {
        then {
          count input-packets;
          accept;
        }
      }
    }
  }
}

filter filter1 {
  term count-packets {
    then {
      count output-packets;
      next term;
    }
  }
}

filter filter2 {
  term block-some-packets {
    from {
      destination-address {
        10.10.10.0/24;
      }
    }
    then {
      discard;
    }
  }
  term accept-others {
    then {
      accept;
    }
  }
}

```



Juniper your Net

2-39

Copyright © 2006 Juniper Networks, Inc.

Proprietary and Confidential www.juniper.net

### Firewall Filter Example

The example on the slide shows the two ways that you can apply firewall filters to an interface. The *filter-in* firewall filter is applied by itself, so the router processes all packets it receives on the *fe-0/0/0* interface through only this one filter and will discard any packets that fail to match a term in the filter with a terminating action. (Because no *from* clause is configured in the last term of the filter, the last term matches all packets and accepts them. Therefore, in this example, all packets match a term with a terminating action within this filter.)

The second way that you can apply firewall filters to an interface is as part of a chain. The router processes all traffic it is preparing to transmit on the *fe-0/0/0* interface through the firewall filter chain specified in the *output-list* configuration setting. In this example, the router processes all packets through the firewall filter *filter1* and the firewall filter *filter2* until the packet matches a term with a terminating action. If a packet passes through both filters without encountering a terminating action, the router discards the packet. (Once again, in this example, no *from* clause is configured in the last term of *filter2*, so this term matches all packets and accepts them. Thus, in this example, all packets match a term with a terminating action prior to reaching the end of the policy chain.)

Note that *from* clauses are used to specify all match conditions, whether source or destination. It is simply the *if...then* statement.

## Firewall Filter Matches

```

▶ Frame 3 (55 bytes on wire, 55 bytes captured)
▶ Ethernet II, Src: 00:0d:60:8b:6f:7f, Dst: 00:05:85:c7:53:d0
▶ Internet Protocol, Src Addr: 10.0.1.100 (10.0.1.100), Dst Addr: 10.0.1.70 (10.0.1.70)
▶ Transmission Control Protocol, Src Port: 1307 (1307), Dst Port: telnet (23), Seq: 0, Ack: 0,
▶ Telnet

```

Packet decode showing typical protocol fields

- Match packets in a stateless fashion
  - Each packet is processed independently of previous or subsequent packets in a given flow
- Can match based on most header fields
- Different categories of match conditions:
  - Numeric-range
  - Address
  - Bit field



### Stateless Filtering

The router processes each packet through the firewall filters independently of all other packets. This processing affects the way you craft firewall filters and also has implications on the information that is available to a router when it processes packets through these filters.

Because the router does not keep state information on connections, you must explicitly allow traffic in both directions for each connection that you want to permit. By contrast, stateful firewall filters only require you to permit the initial connection and then permit bidirectional communications for this connection.

The stateless nature of these filters also affects the information available to the router when processing these filters. For example, if you want to allow all *established* TCP sessions through a router, you can have the firewall filter permit all TCP packets that have the acknowledgement (ACK) flag set in the TCP header. However, looking for this match condition provides no guarantee that the session was actually properly established. This could instead be a packet that was maliciously crafted to have the ACK flag set for an unestablished TCP session.

### Match on Header Fields

You specify the criteria to be used to match packets in *from* clauses within firewall filter terms. You can use many header fields as match criteria. However, you must remember that all header fields might not be available to you due to the way firewall filters are processed.

*Continued on next page.*



## Match on Header Fields (contd.)

When you specify a header field, the JUNOS software looks for a match at the location in the header where that field should exist. However, it does not check to ensure that the header field makes sense in the given context. For example, if you specify that the software should look for the ACK flag in the TCP header, the software looks for that bit to be set at the appropriate location, but it does not check that the packet was actually a TCP packet. Therefore, you must account for how the software looks for a match when writing your filters. In this case, you would have the router both check that the packet was a TCP packet and whether the TCP ACK flag was set.

The stateless nature of firewall filters can affect the information available in the processing of fragmented packets. Processing fragments is trickier with stateless firewall filters than with a stateful firewall filter. The first fragment should have all the Layer 4 headers; however, subsequent fragments will not. Additionally, attempting to check Layer 4 headers in fragments produces unpredictable results. As was explained previously, the JUNOS software still attempts to evaluate the Layer 4 headers; however, the second and subsequent fragments do not contain these headers, so the matches are unpredictable.

## Categories of Match Conditions

Match conditions generally fall into three categories: numeric range, address, and bit-field match conditions. You can generally use the same evaluation options for each condition within the category. There are also several *synonyms*, which are match conditions that are equivalent to one or more of these match conditions.

## Numeric Range Filter Match Condition

- Match packet fields that can be identified by a numeric value
  - Port and protocol numbers
- Specify a keyword that identifies the condition and a value that a field in a packet must match
  - `source-port 1024-65535;`
  - `source-port ssh;`
  - `source-port [ telnet smtp ];`
- Keywords identifying available fields:
  - `destination-port, dscp, esp-spi, forwarding-class, fragment-offset, icmp-code, icmp-type, interface-group, ip-options, packet-length, port, precedence, protocol, source-port, *-except`



### Numeric Range Match Conditions

Use this category of match conditions to match packet fields that can be identified by a numeric value. For example, the Layer 4 protocol, source and destination port numbers, and packet length are all represented by numeric fields in packet headers.

### Configuring Numeric Range Matches

To configure a numeric range match condition within the *from* clause of a firewall filter, you specify the keyword that identifies the match condition and the value(s) that field must match. You can specify values in three ways: a single value, a range of values (for example, `source-port 1024-65535` matches all packets with a source port that is between 1024 and 65535, inclusive), or a list of values enclosed within square brackets (for example, `source-port [ telnet smtp ]` matches all packets with a source port that is either 23 [telnet] or 25 [smtp]).

For some match conditions (for example, `protocol`, `port`, and `icmp-type`), you can use strings to represent numeric values. You can use the question-mark (?) help to determine the available string representations.

As with all rules, there is an exception and, in this case, it is the `forwarding-class` option. With that match condition, you must specify the forwarding class by its string. You can only enter a single forwarding class or a list enclosed in square brackets.

*Continued on next page.*

## Keywords

The slide lists the keywords that specify match conditions. In any *from* clause, you can either use the `destination-port`, the `source-port` match condition, or both, or you can use the `port` match condition. You cannot use both the `port` match condition and also use either the `destination-port` or `source-port` match conditions.

You can also append the `-except` suffix to any of the keywords to specify that all values *except* those specified should match. For example, `source-port-except [ telnet smtp ]` matches all packets with source ports other than 23 (`telnet`) or 25 (`smtp`).

## Address Filter Match Condition

- IP source and destination addresses
- Longest match
  - Can specify `except` keyword to specifically not match prefixes
  - Implicit `0/0 except;` statement
- Keywords available:
  - `address, prefix-list`
  - `destination-address, destination-prefix-list`
  - `source-address, source-prefix-list`



### Address Matches

You can match packets based on source or destination IP addresses. You can match source addresses, destination addresses, or either field. You specify the addresses to be matched using the address/mask notation. You can use dotted-decimal notation or prefix-length notation to specify the mask. You can list multiple address/mask pairs per match condition. You can also use prefix lists to specify address/mask pairs to be matched.

### Longest-Match Evaluation

In addition to specifying the addresses that are to match within address match conditions, you can also specify addresses that should not match. You configure these exceptions using the `except` keyword. An implicit `0/0 except;` statement is in each address match condition.

When the router evaluates the addresses within a packet against address match conditions, it looks for the most specific address/mask combination that the packet matches (the most specific matching address/mask combination is called the *longest match*). If the longest match is an address/mask entry without the `except` keyword, the packet matches the address match condition. If the longest match is an address/mask entry with the `except` keyword, the packet does not match the address match condition.

*Continued on next page.*

## Longest-Match Evaluation (contd.)

When you use the prefix-list match conditions, you can specify multiple prefix lists. When you configure a prefix-list match condition, it is equivalent to configuring each prefix listed in the prefix list in the equivalent address match condition. You can use the `except` keyword when specifying a prefix list. If you use the `except` keyword when specifying a prefix list, it is equivalent to configuring each individual prefix with the `except` keyword in the equivalent address match condition.

## Configuring Address Match Conditions

You can specify address/mask combinations within an address match condition using the `address`, `source-address`, and `destination-address` match conditions. You can also reference prefix lists using the `prefix-list`, `source-prefix-list`, and `destination-prefix-list`.

When you specify prefix lists, it is equivalent to configuring the prefixes in the equivalent address match condition. (For example, if you configure the `destination-prefix-list` match condition, it is equivalent to configuring the prefixes in the specified prefix lists directly in the `destination-address` match condition.) You can use both the `prefix-list` match conditions and the equivalent address match conditions in the same *from* clause.

The `address` and `prefix-list` match conditions match either source or destination addresses. As with ports, in any *from* clause, you can either use any combination of the `destination-address`, `source-address`, `destination-prefix-list`, and `source-prefix-list` match conditions, or you can use any combination of the `address` and `prefix-list` match conditions. You cannot use both a combination of the `address` and `prefix-list` match conditions and also use the `destination-address`, `source-address`, `destination-prefix-list`, or `source-prefix-list` match conditions.

In general, you should *not* use the `address` or `prefix-list` match conditions. Usually, you can write more secure filters by instead specifying source and destination addresses. In addition, when a firewall filter term includes either the `address` or `prefix-list` match condition and a subsequent term includes the `source-address` or `source-prefix-list` match condition for the same address, the router might use the latter term to process packets prior to evaluating intervening terms. You can avoid this behavior by replacing all terms with the address match condition with two terms, one containing the `source-address` match condition and one containing the `destination-address` match condition.

## Address Filter Match Example

```
[edit]
lab@London# show firewall
family inet {
  filter reject-some-addresses {
    term unused-private-addresses {
      from {
        source-address {
          10.0.0.0/8;
          10.14.10.0/24 except;
          172.16.0.0/12;
          192.168.0.0/16;
          192.168.200.0/24 except;
        }
      }
      then {
        reject;
      }
    }
  }
}
```

Which source addresses will match?



### Address Filter Match Example

The slide provides an example of a firewall filter term that matches source addresses using the `source-address` match condition. This firewall filter matches all source addresses within the 10.0.0.0/8 prefix except those within the 10.14.10.0/24 prefix. It also matches all source addresses within the 172.16.0.0/12 prefix. And, it matches all source addresses within the 192.168.0.0/16 prefix except those within the 192.168.200.0/24 prefix.

## Prefix-List Address Filter Match Example

```
[edit]
lab@London# show policy-options
prefix-list internal-routes {
    10.14.10.0/24;
    192.168.200.0/24;
}
prefix-list rfc1918-addresses {
    10.0.0.0/8;
    172.16.0.0/12;
    192.168.0.0/16;
}
[edit]
lab@London# show firewall
family inet {
    filter reject-some-addresses {
        term unused-private-addresses {
            from {
                source-prefix-list {
                    rfc1918-addresses;
                    internal-routes except;
                }
            }
            then {
                reject;
            }
        }
    }
}
```

Which source addresses will match?



### Prefix-List Match Example

The slide provides an example of a firewall filter term that matches source addresses using the `source-prefix-list` match condition. Once you consider the way the prefixes are imported from the prefix lists, you will note that this filter is equivalent to the filter on the previous slide.

This firewall filter matches all source addresses within the 10.0.0.0/8 prefix except those addresses within the 10.14.10.0/24 prefix. It also matches all source addresses within the 172.16.0.0/12 prefix. And, it matches all source addresses within the 192.168.0.0/16 prefix except those within the 192.168.200.0/24 prefix.

## Noncontiguous Address Filters

- Similar to *wildcard masks* supported by ACLs in other OSs
- Specify with the address/mask notation
  - Mask is *not* a wildcard mask—a 1 indicates that the bit must match, a 0 indicates that the bit does not need to match
  - Can still use the `except` keyword
  - Examples:

```
destination-address {
  0.16.0.0/0.255.0.0;
  172.16.0.1/255.255.0.255;
}
destination-address {
  172.16.0.0/12;
  172.16.0.1/255.240.1.255 except;
}
```



### Wildcard Masks

Other operating systems allow you to match noncontiguous addresses by specifying an address/wildcard mask combination, providing a very flexible match condition. Wildcard masks are sometimes called the *inverse* of network masks. Unlike network masks, where a 1 indicates that a bit is *significant* and must match, wildcard masks use a 0 to indicate that a bit is significant. Therefore, to match all addresses in the 172.16.0.0/16 prefix, you specify a wildcard mask of 0.0.255.255.

For example, to match all addresses with an odd number in the third octet, you could specify an address of 0.0.1.0 and a wildcard mask of 255.255.254.255. For a more realistic example, assume the 10.10.0.0/16 prefix was divided into /24 subnets and the gateway was always at the .1 address. To match all gateway addresses, you could specify an address of 10.10.0.1 and a wildcard mask of 0.0.255.0.

### Noncontiguous Address Filters in JUNOS Software

The same functionality is available in JUNOS software, but without the complication of using wildcard masks. In JUNOS software, you use a normal mask. In other words, to match all addresses in the 172.16.0.0/16 prefix, you specify a mask of 255.255.0.0. In addition, you can still use the `except` keyword when specifying noncontiguous addresses, providing additional flexibility.

*Continued on next page.*



## Noncontiguous Address Filters in JUNOS Software (contd.)

In the examples on the slide, the first destination-address match condition matches all destination addresses with a 16 in the second octet and all addresses in the 172.16.0.0/16 prefix with a 1 in the fourth octet.

For the second example on the slide, assume that the 172.16.0.0/12 prefix is divided into /23 subnets and that the first usable address in each subnet is the gateway. This example matches all addresses in the 172.16.0.0/12 prefix *except* the gateway addresses. The 255.240.1.255 mask specifies that the first 12 bits are significant (they specify the prefix) and that the last 9 bits are also significant (they are the host portion of addresses in a /23). The bits in the middle define the subnet and can be any value. Therefore, the mask contains zeros for these bits. The end result is that this address/mask combination matches the first usable address in each /23 within the 172.16.0.0/12 prefix.

## Bit-Field Match Condition

- Match on specific bits in certain packet fields
- You can specify bit fields with symbolic names or numeric values
- Bit matching for `ip-options`, `fragment-flags`, and `tcp-flags`
  - Note: Specification of a bit field does *not* imply the corresponding protocol
- Grouping (...), negation (!), and support for logical AND (& or +) and logical OR (| or ,) functions
  - Example: `tcp-flags "(syn & !ack) | rst"` matches any packet that is the initial TCP packet in a stream or is a TCP reset



### Bit-Field Match Conditions

Bit-field match conditions allow you to select packets using information contained in bit fields within the header. A common example is the TCP flags in the TCP header. The TCP flags are each one bit, so looking for a specific flag within the TCP flags header field requires performing a bit-by-bit comparison.

### Symbolic Names

Bit-field matches can use symbolic names for the most common bit configurations. Bit-field match keywords always map to a single bit value. You also can specify bit fields as hexadecimal or decimal numbers. The symbolic names and their hexadecimal equivalent are listed in the *JUNOS Policy Framework Configuration Guide*.

### Bit Matching

To specify the bit-field value to match, enclose the value in quotation marks (double quotes). For example, the match condition `tcp-flags "rst"` matches all packets with the RST bit in the TCP flags field set.

*Continued on next page.*

## Bit Matching (contd.)

Configuring a match condition that specifies that a Layer 4 protocol does not cause JUNOS software to check that the packet uses that protocol. In other words, just using the `tcp-flags` match condition in no way assures that only IP packets with TCP segments inside will be examined by the firewall filter. If an IP packet containing OSPF information (for example) happens to have the right combination of bits where the TCP flags would be in a TCP segment, an unintended match will occur. To prevent unintended matches, use the `protocol` match condition in the *from* clause to ensure the packet uses the intended protocol.

Additionally, the router does not check that the packets are the first fragment. Layer 4 headers are generally only available in the first fragment of fragmented packets. Unless the *from* clause also includes a match condition that restricts the router to examining the first fragment of fragmented packets, the router looks for the specified bits where they would exist in the first fragment (or an unfragmented packet). This process can cause unpredictable matches in the second and subsequent fragments. To prevent unpredictable matches, use the `fragment-offset 0;` match condition in the *from* clause to ensure that the router only inspects unfragmented packets and the first fragments of fragmented packets. You must also permit or deny subsequent fragments in a separate term; however, you cannot use Layer 4 header fields when determining whether to permit or deny these fragments.

## Logical Operators

You can use logical operators when operating on bit fields. These operators have a distinct precedence from high to low when they are combined in the same match condition.

The following table lists the bit-field logical operators from highest precedence to lowest:

Logical Operators

Logical Operator	Description
( ... )	Grouping of bit-field match conditions
!	Negation of bit field
& or +	Logical AND between bit-field matches
or ,	Logical OR between bit-field matches

For example, `tcp-flags "syn & !ack";` matches all packets that both have the TCP SYN flag set and also do not have the ACK flag set. (In other words, it matches the very first packet in the TCP connection.) The match condition `tcp-flags "(syn & !ack) | rst";` matches all packets that either have the TCP RST flag set, OR that both have the TCP SYN flag set and also do not have the ACK flag set. (In other words, this match condition matches the very first packet in the TCP connection or a TCP reset.)

## Text Synonyms

- Use text synonyms for common matches
  - `first-fragment`: Matches the first fragment of a fragmented packet (synonym for `fragment-offset 0;` and `fragment-flags more-fragments;`)
  - `is-fragment`: Matches fragments other than the first fragment (synonym for `fragment-offset-except 0;`)
  - `tcp-established`: Synonym for `tcp-flags "(ack | rst)";`
  - `tcp-initial`: Synonym for `tcp-flags "(syn & !ack)";`



### Text Synonyms

Several text synonyms are available for your use and are documented on the slide. These text synonyms take no arguments. As usual, the `tcp-established` and `tcp-initial` match conditions do not cause the router to check that the packet is a TCP packet. When using these text synonyms, use the `protocol tcp;` match condition in the *from* clause to ensure that the packet is a TCP packet.

## Common Actions

- Common actions in firewall filters:
  - Terminating actions:
    - `accept`
    - `discard`
    - `reject`
  - Flow control:
    - `next term`
  - Action modifiers:
    - `count`, `log`, `syslog`
    - `forwarding-class`, `loss-priority`
    - `policer`
    - `sample`



### Common Actions

You specify actions in the *then* clause of a term. You can specify terminating actions, an action that modifies the flow of firewall filter evaluation, and action modifiers.

Terminating actions cause the policy evaluation to stop. The `accept` action causes the router to accept the packet and continue the input or output processing of the packet. The `discard` action causes the router to silently discard the packet, without sending an Internet Control Message Protocol (ICMP) message to the source address. The `reject` action causes the router to discard the packet and send a message back to the source address. The default message the router sends is an ICMP message with the destination unreachable message type and administratively prohibited message code. You can use an optional argument to the `reject` action to cause the router to send an ICMP message with a different message code or to cause the router to send a TCP reset instead of an ICMP message. If you specify the `tcp-reset` option, the router responds to TCP packets with a TCP reset, but it sends no message in response to non-TCP packets.

You can use the `next term` action to specify that the router should continue processing further terms in the filter or filter chain.

You can specify one or more action modifiers with any terminating or flow-control action. If you specify an action modifier, but not action, the router implies an action of `accept`. You can use the `count`, `log`, and `syslog` action modifiers to record information about packets. The `forwarding-class` and `loss-priority` action modifiers are used to specify class-of-service (CoS) information, which we explain in that chapter. The `policer` action modifier allows you to invoke a traffic policer. The `sample` action modifier specifies that the router should sample the packets. Traffic sampling is outside the scope of this course.

## Counting and Logging

- Count matching packets with the `count` counter action modifier
  - Access counter with **show** commands:
    - `show firewall filter filter` (shows all counters configured in a filter)
    - `show firewall filter filter counter counter`
- Log matching packets with the `log` action modifier
  - Access traffic log with the `show firewall log` command:

```
lab@router> show firewall log
Log :
Time      Filter      Action Interface      Protocol Src Addr      Dest Addr
22:34:02  classify-pak A    local             UDP          172.17.37.4    172.17.38.4
21:57:53  classify-pak A    local             ICMP         172.17.37.4    172.17.38.4
21:57:47  pfe              A                fe-2/0/1.11   GRE          172.17.37.4    172.17.38.4
21:57:46  pfe              A                fe-2/0/1.11   GRE          172.17.37.4    172.17.38.4
21:57:45  pfe              A                fe-2/0/1.11   GRE          172.17.37.4    172.17.38.4
21:57:17  classify-pak A    local             ICMP         172.17.37.4    172.17.38.4
21:57:16  classify-pak A    local             ICMP         172.17.37.4    172.17.38.4
```



### Counting Packets

You can count packets by specifying the **count** action modifier with a named counter. Counters maintain a cumulative packet and byte count. Counters are specific to the filter, so the router keeps separate statistics for counters with identical names that exist in separate filters. By default, the router keeps only one set of statistics for each counter in a filter, so if the same filter is applied to multiple interfaces, all matching packets from all interfaces where the filter is applied increment the same counter. You can access counter statistics with the commands shown on the slide. You can reset counter statistics with the CLI command `clear firewall filter filter` command. You can specify an optional **counter** counter argument to reset the statistics for a single counter.

You can also configure the router (on a per-filter basis) to keep interface-specific statistics for counters. When you configure the router in this way, the router creates a new counter for every logical interface and traffic direction where the filter is applied.

### Logging Packets

You can log packets by specifying the **log** action modifier. You can display the logged packets using the `show firewall log` CLI command. Sample output is shown on the slide. A filter name only appears if the packet is switched by the Routing Engine. Otherwise, a dash (-) or `pfe` appears instead of the filter name to indicate that the packet was switched by the PFE.

## Policers

- Instead of allowing or dropping packets that meet match conditions, you can use a filter to identify traffic that is to be policed (rate-limited)
  - You can also apply a policer directly to an interface to rate-limit all traffic associated with that protocol family
- Traffic that matches the filter is then policed according to an average bandwidth and a burst size
  - Can specify bandwidth as a percentage of interface speed
- When traffic exceeds the policing parameters, you can specify an action and action modifiers



### Filters Identify Traffic for Policing

In addition to dropping or accepting packets, firewall filters can also be used to police or rate-limit traffic. Rate policing enables you to limit the amount of traffic that passes into or out of an interface. Firewall filters that use rate policing still employ normal match conditions, such as addresses, protocols, ports, and so on, to determine which traffic on an interface is subject to rate limiting. As usual, lack of a *from* clause matches all packets that did not match an earlier firewall filter term. If the first term in a firewall filter lacks a *from* clause and contains a policer, all packets on the interface (input or output, as the filter is applied) are subject to rate policing.

However, JUNOS software also accommodates interface-based policers that you apply directly to a given protocol family on a given logical unit of a particular interface. Such policers accommodate Layer 2 VPN traffic as well as the MPLS and IPv6 families and operate without the need for a calling firewall filter. We detail interface-based policers on a subsequent page.

### Rate Limits

Policing employs the *token-bucket algorithm*, which enforces a limit on average bandwidth while allowing bursts up to a specified maximum value. You configure two rate limits for the traffic: bandwidth, which is the number of bits per second permitted on average, and maximum burst size, which is the bursts of data that exceed the given bandwidth limit, but only up to a total number of bytes given by this value.

*Continued on next page.*

## Rate Limits (contd.)

The preferred method for determining the maximum burst size is to multiply the interface's speed by the amount of time bursts should be allowed at that bandwidth level. For example, to allow bursts on a Fast Ethernet link for 5 milliseconds (a reasonable value), use the following calculation:

burst size = bandwidth (=100,000,000 bits/sec) x allowable burst time (=5/1000s)

This calculation yields a burst size of 500,000 bits. You can divide this number by 8 to convert it to bytes, which gives you a burst size of 62500 bytes.

You specify the bandwidth as a number of *bits* using the **bandwidth-limit** statement or as a percentage of interface bandwidth using the **bandwidth-percent** statement. You specify the maximum burst size as a number of *bytes* using the **burst-size-limit** statement.

## Excess Traffic

When a packet matches a term that has a policer in the *then* clause, the router first determines if the packet exceeds the policer. If the packet does *not* exceed the policer, the router performs the actions in the firewall filter's *then* clause as if the policer were not configured. If the packet *does* exceed the policer, the router takes the actions in the policer's *then* clause. If the policer's *then* clause does not result in the packet being discarded, the router takes the remainder of the actions in the firewall filter's *then* clause. In cases where action modifiers are defined in both the policer's *then* clause and the firewall filter's *then* clause, the router uses the policer's action modifiers.

For example, the following firewall filter polices all TCP traffic that exceeds 10 Mbps with a 62500-byte burst size. It places traffic that exceeds these limits in the best-effort forwarding class, while traffic that conforms to these limits is placed in the assured-forwarding forwarding class:

```
firewall {
  policer class-example {
    if-exceeding {
      bandwidth-limit 10m;
      burst-size-limit 62500;
    }
    then forwarding-class best-effort;
  }
  family inet {
    filter example1 {
      term policer-example {
        from {
          protocol tcp;
        }
        then {
          policer class-example;
          forwarding-class assured-forwarding;
          accept;
        }
      }
    }
  }
}
```



## Policing Example

```
[edit firewall]
lab@router# show
policer p1 {
  if-exceeding {
    bandwidth-limit 400k;
    burst-size-limit 100k;
  }
  then discard;
}
family inet {
  filter limit-ftp {
    term ftp {
      from {
        source-address {
          1.2.3.0/24;
        }
        protocol tcp;
        destination-port [ ftp ftp-data ];
      }
      then {
        policer p1;
        count count-ftp;
      }
    }
  }
}
```

### ■ Example:

- `bandwidth-limit`
  - In bits per second
  - 30,520 bps to 4.29 Gbps
- `burst-size-limit`
  - In bytes per second
  - Minimum should = 10 times MTU (low speed) or bandwidth times 3-5 milliseconds (high speed)



## Policing Example

In the example on the slide, we define a policer called *p1* that discards traffic that exceeds the defined average bandwidth of 400 kbps and the defined burst-size limit of 100 KB. Once we define this policer, we can call it from any firewall filter. By default, the router treats each invocation of the policer separately, and tracks statistics separately for each term that references the policer. You can think of the policer definition as simply defining a set of parameters, which we can choose to reference in any firewall filter term.

We call it in the filter *limit-ftp* to police FTP traffic from the specified block of clients. If the traffic exceeds the limits, the router discards it. If the traffic does *not* exceed the limits, the router accepts it and counts it in the counter called *count-ftp*. Traffic that exceeds the policer is not counted.

As the slide shows, you can use the k, m, and g abbreviations to indicate one thousand, one million, and one billion bytes or bits, respectively.

## Interface-Based Policers

```

interfaces {
  interface-name {
    unit logical-unit-number {
      family inet {
        filter {
          input filter-name;
          output filter-name;
        }
        policer {
          input policer-template;
          output policer-template;
        }
      }
    }
  }
}

```



### Interface-Based Policers

One way to do interface policing (rate limiting) of input and output traffic is to configure a firewall filter with a then `policer` action modifier that is applied to the desired interface. The purpose of interface-based policers is to allow a mechanism to support policing at the protocol family level, independent of any firewall filters that might, or might not be, applied to that interface.

Let's say you want to limit the bandwidth of an attached device to 30-Mbps input and output with a supported burst size of 200 KB. With interface policers, you can apply a policer to an interface under a particular protocol family in a manner that is independent of any firewall filters. All traffic associated with the related family is subjected to the action of the policer because no match conditions exist for an interface policer.

Keep in mind that you can also call the policer from within firewall actions in addition to having your policer separate from any firewall actions. You can also define different policers to be applied to the desired interface as input or output policers.

Because the router evaluates an interface policer before any firewall invokes policers, it is possible to police the same traffic twice at ingress and twice at egress, using a combination of interface and firewall-invoked policers in both the input and output directions.

## Filters Within Filters

- You can include a filter within another filter by referencing it in a term

- Can only include one level of filters

- Example...

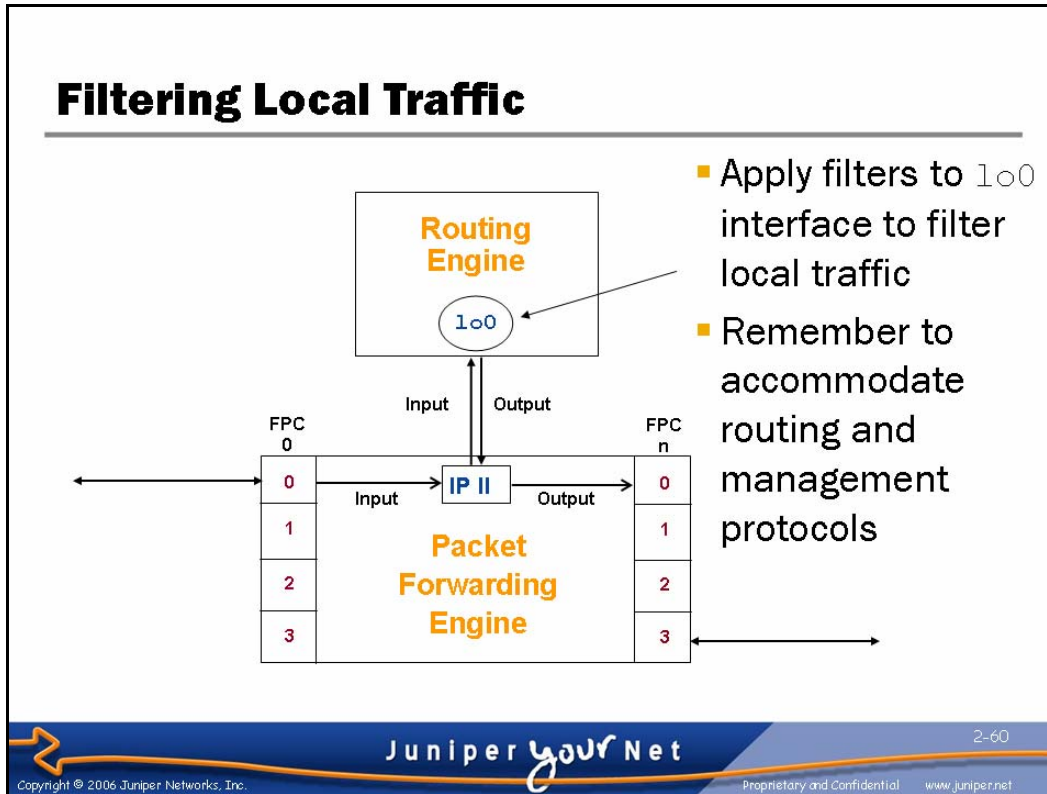
```
[edit firewall]
lab@router# show
family inet {
  filter limit-ftp {
    term include-generic {
      filter generic-terms;
    }
    term ftp {
      from {
        source-address {
          1.2.3.0/24;
        }
        protocol tcp;
        destination-port [ ftp ftp-data ];
      }
      then accept;
    }
  }
  filter generic-terms {
    term block-spoofed {
      from {
        source-prefix-list {
          internal-routes;
        }
      }
      then {
        discard;
      }
    }
  }
}
```

### Filters Within Filters

You can include a different firewall filter within a firewall filter term. When you do that, the packet is processed through that term as if the entire referenced filter had been included at that point. You cannot include firewall filters that themselves include other filters.

### Example

In the example on the slide, the firewall filter *limit-ftp* includes the firewall filter *generic-terms*, which contains terms (or, in this case, just a single term) meant to be used in multiple firewall filters. The router processes incoming packets through the *limit-ftp* filter as if all the terms in the *generic-terms* filter had been included where term *include-generic* exists.



### Filtering Local Traffic

Transit firewall filters act on the packets that flow directly from one interface to another on the router. These filters can protect sites from unauthorized access and other threats. But what about protecting core routers from unauthorized management access and other harmful effects? This is the idea behind applying a firewall filter to protect the Routing Engine. These filters are applied on the Packet Forwarding Engine (PFE) before traffic ever reaches the control plane.

Traditionally, two types of tools are used in a layered fashion to protect core routers. The first line of defense is the router's remote access management policy, which is essentially an IP address list. Management access to the router (for example, using Telnet or SNMP) requires an allowed source IP address. After the source IP address is verified, a second tool, such as passwords or one-time passwords, provides a second layer of security.

The overall benefits of applying a firewall filter to 100 on the Routing Engine and other interfaces are the following:

- The added security to protect routers from unauthorized access;
- Permanent filters do not impact forwarding performance; and
- On M-series routers, hardware filtering does not impact the Routing Engine, while on J-series routers, the `fwdd` process performs filtering in the real-time thread without disturbing the control plane processes.

*Continued on next page.*

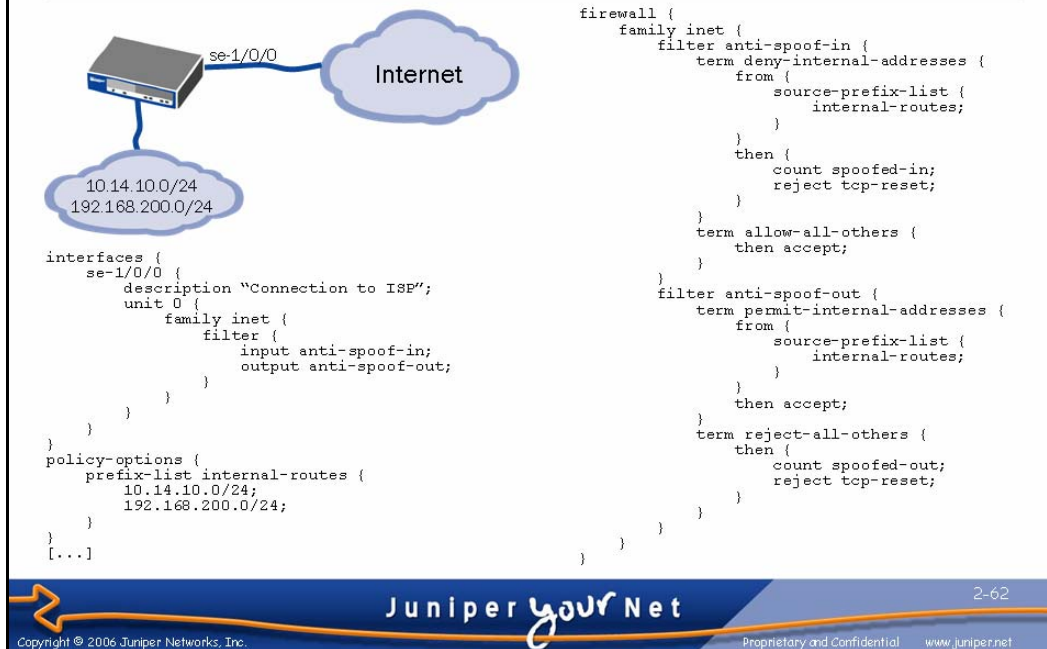
### Filtering Local Traffic (contd.)

As the diagram on the slide shows, *input* filters applied to a physical interface affect traffic inbound on a router interface, while input filters applied to the `lo0` interface affect traffic inbound to the RE from the PFE.

### Remember to Accommodate Routing Protocols

No automatic *holes* are created in the `lo0` firewall filter. Therefore, in addition to allowing management traffic, you must also allow routing protocol and other control traffic to reach the RE. The default silent discard can be the cause of unintentional effects. Also, you cannot sample local traffic or generate cflowd information export for local traffic.

## Firewall Filter Example 1



### Firewall Filter Example 1

In the example on the slide, the list of internal routes is defined in the prefix list *internal-routes*. This prefix list is used to create two antispoofing firewall filters. The filter *anti-spoof-in* blocks packets with an internal source address from entering the network. The filter *anti-spoof-out* blocks packets *without* an internal source address from leaving the network. In both cases, the router counts offending packets. Additionally, the router responds to offending TCP packets with a TCP reset and silently discards all other offending packets.

We discuss a better and easier way to prevent spoofing (unicast RPF) in a few pages.

## Firewall Filter Example 2

```

interfaces {
  fe-0/0/0 {
    unit 0 {
      family inet {
        filter {
          group 1;
        }
      }
    }
  }
  fe-2/0/0 {
    unit 0 {
      family inet {
        filter {
          group 1;
        }
      }
    }
  }
  lo0 {
    unit 0 {
      family inet {
        filter {
          input protect-re;
        }
      }
    }
  }
}

policy-options {
  prefix-list internal-routes {
    10.14.10.0/24;
    192.168.200.0/24;
  }
  prefix-list bgp-peers {
    apply-path "protocols bgp group <*> neighbor <*>";
  }
}
[...]

firewall {
  family inet {
    filter protect-re {
      term allow-telnet-ssh {
        from {
          source-prefix-list {
            internal-routes;
          }
          protocol tcp;
          port [ ssh telnet ];
        }
        then accept;
      }
      term allow-bgp {
        from {
          source-prefix-list {
            bgp-peers;
          }
          protocol tcp;
          port 179;
        }
        then accept;
      }
      term allow-ospf {
        from {
          interface-group 1;
          protocol ospf;
        }
        then accept;
      }
      term permit-ping {
        from {
          protocol icmp;
          icmp-type echo-request;
        }
        then accept;
      }
    }
  }
}

```



### Firewall Filter Example 2

The slide shows an example of a fairly simple filter to protect the Routing Engine. The `interfaces` configuration shows that we have configured two Fast Ethernet interfaces to be in `group 1` under the `family inet` filter hierarchy. This configuration allows you to later use the `interface-group` match condition in a firewall filter to determine whether the packet was received on one of the interfaces in the specified group. We use this information in the `allow-ospf` term to allow OSPF traffic from interfaces where we expect to have OSPF neighbors.

The `bgp-peers` prefix list uses the `apply-path` syntax to automatically include IP information from another section of the configuration. In this case, it automatically includes the IP addresses of all BGP neighbors from all groups. This configuration allows us to automatically update our prefix list (and, therefore, our firewall filter) every time we configure a new BGP peer, easing configuration maintenance.

The firewall filter also allows inbound pings in the term `permit-ping`, which allows others to ping the router. However, this filter is incomplete because it blocks some important items. For example, it does *not* allow replies to pings (`echo-reply`) or traceroutes (`time-exceeded` and `unreachable`), which prevents the router from successfully using these tools. It also does not allow the router to initiate SSH, Telnet, or FTP connections. The filter also blocks the Network Time Protocol (NTP). A more complete filter to protect the RE is available in Appendix C.

## Firewall Filter Example 3

```
[edit]
lab@router# show interfaces se-1/0/0
description "Connection to ISP";
unit 0 {
    family inet {
        filter {
            input filter-internet-in;
        }
    }
}

[edit]
lab@router# show policy-options
prefix-list internal-routes {
    10.14.10.0/24;
    192.168.200.0/24;
}

[edit]
lab@router# show firewall family inet
filter filter-internet-in {
    term allow-established-sessions {
        from {
            destination-prefix-list {
                internal-routes;
            }
            fragment-offset 0;
            tcp-established;
            protocol tcp;
        }
        then accept;
    }
    term allow-webserver-connections {
        from {
            destination-address {
                10.14.10.200/32;
            }
            protocol tcp;
            destination-port [ 80 443 8080 ];
        }
        then accept;
    }
    term deny-udp-icmp-frags {
        from {
            is-fragment;
            protocol [ icmp udp ];
        }
        then {
            discard;
        }
    }
    term allow-tcp-frags {
        from {
            is-fragment;
            protocol tcp;
        }
        then {
            accept;
        }
    }
    term allow-udp {
        from {
            destination-prefix-list {
                internal-routes;
            }
            protocol udp;
        }
        then accept;
    }
    term allow-some-icmp {
        from {
            destination-prefix-list {
                internal-routes;
            }
            protocol icmp;
            icmp-type [ echo-reply time-exceeded unreachable ];
        }
        then accept;
    }
}
[...]
```



### Firewall Filter Example 3

This firewall filter is applied to the packets received from the ISP. No firewall filter is applied outbound, so all traffic is allowed outbound; however, because the inbound firewall filter can prevent connections from being established by blocking packets, some of this firewall filter actually defines the connections that should be allowed *outbound*. This *reverse thinking* is one of the drawbacks of using stateless filters to control bidirectional traffic.

The filter starts by allowing all *established* TCP sessions, which are TCP packets with the ACK or RST flag set. This filter allows all outbound TCP sessions to establish and also matches packets in established inbound TCP streams. Because the Layer 4 headers are not in the second and subsequent fragments, the filter only checks packets that have a fragment offset of 0. The second term allows inbound connections to a Web server. Note that only the initial packet of the TCP session reaches this term. Subsequent packets match the *allow-established-sessions* term. The third term denies ICMP and UDP fragments, while the fourth term allows TCP fragments. The fifth term allows all inbound UDP, which also allows the return traffic of outbound UDP connections and inbound traceroute. The sixth term allows inbound ICMP echo replies (which effectively permits *outbound* pings) and ICMP TTL exceeded and unreachable messages (which effectively permits outbound traceroutes as well as allowing errors about outbound traffic to reach its source inside the network).



## **Agenda: JUNOS Policy**

---

- Policy Language and Policy Evaluation Process Overview
- Routing Policy
- Firewall Policy
- Unicast Reverse-Path Forwarding Checks



### **Unicast Reverse-Path Forwarding Checks**

The slide highlights the topic we discuss next.

## Unicast RPF Check

- Automates antispoofing filters based on routing table
- Two modes:
  - Strict (default): Accept the packet if:
    - The source address of the packet matches an active route
    - The selected next hop to that active route is the interface on which the packet arrived
  - Loose: Accept the packet if:
    - The source address of the packet matches a prefix in the routing table
  - Packets always match *loose* mode with a default route



### Automated Antispoofing Filters

The unicast reverse-path-forwarding (RPF) checks validate that packets are received on interfaces where the router would expect to receive such traffic. By default, the router expects to receive traffic on a given interface if it has an active route to the packet's source address and the packet was received on the interface that is the next hop for the active route to the packet's source address.

For example, if the router receives a packet with a source address of 10.10.10.10 on interface `fe-2/0/0.0` and the router is configured to perform the unicast RPF check on that interface, it examines its routing table for the best route to 10.10.10.10. If this route lookup returns a route for 10.10.10.0/24 with a next hop of `fe-2/0/0.0`, the packet passes the unicast RPF check and is accepted. You can combine both unicast RPF and firewall filters on the same interface.

### Loose Versus Strict

By default, the router uses the strict mode RPF check. You can instead configure it to use the *loose* mode RPF check, which only checks to ensure a valid route to the source address in the routing table. However, in networks with a default route, there is always a valid route to every IP address; so, using a loose mode check does not make sense in this environment. In general, using the default *strict* mode provides the best results.

## Unicast RPF Caveats

- **Active vs. feasible paths:**
  - By default, only active paths to a prefix are checked, which can cause drops when multiple paths exist
  - To consider all feasible paths, use the `set routing-options forwarding-table unicast-reverse-path feasible-paths` command
- **Fail filters:**
  - If you specify a fail filter, it processes all traffic that fails the RPF check
  - Allows you to accept, log, count, etc. denied traffic
- **DHCP/BOOTP**
  - Denied by default by RPF
  - To allow, create a fail filter that permits traffic with a source address of 0.0.0.0 and destination address of 255.255.255.255

### Active Versus Feasible Paths

By default, when the router performs its RPF check, it only considers the active routes to a given destination. In networks where there is perfectly symmetric routing, the default behavior of only considering active routes should work. However, in cases where there is the possibility of asymmetric routing (different forward and reverse paths), this can cause legitimate traffic to be dropped. To alleviate this issue, you can cause the router to consider all *feasible* routes to a destination when it performs the RPF check. In this mode, the router considers all routes it received to a given prefix, even if they are not the route the router has currently chosen to be its active route to the destination. In networks where the possibility of asymmetric routing exists, you should activate this option.

### Fail Filters

When the router decides that a packet has failed the RPF check, it discards it by default. However, if you specify an optional fail filter, the router processes packets that fail the RPF check through that filter prior to discarding them. In the fail filter, you can perform all the actions and action modifiers you could in any other firewall filter, including accepting the traffic despite the packet failing the RPF check. (Notably, if you choose to log packets in an input firewall filter, but the packets then fail the RPF check, they are not logged. To log these packets, you must log them in an RPF fail filter.)

*Continued on next page.*

## **DHCP/BOOTP and Unicast RPF**

DHCP and the bootstrap protocol (BOOTP) requests will always fail the RPF checks. To allow these requests, you must configure a fail filter that permits traffic with a source address of 0.0.0.0 and destination address of 255.255.255.255. We show an example of this on the next page.

## RPF Example

```

firewall {
    family inet {
        filter filter-internet-in {
            term allow-established-sessions {
                from {
                    protocol tcp;
                    tcp-established;
                }
                then accept;
            }
            term allow-udp {
                from {
                    protocol udp;
                }
                then accept;
            }
            term allow-some-icmp {
                from {
                    protocol icmp;
                    icmp-type [ echo-reply time-exceeded unreachable ];
                }
                then accept;
            }
        }
        filter rpf-dhcp {
            term dhcp {
                from {
                    source-address {
                        0.0.0.0/32;
                    }
                    destination-address {
                        255.255.255.255/32;
                    }
                }
                then accept;
            }
        }
    }
}

interfaces {
    fe-0/0/0 {
        description "Internal segment";
        unit 0 {
            family inet {
                rpf-check fail-filter rpf-dhcp;
            }
        }
    }
    se-1/0/0 {
        description "Internet Connection";
        unit 0 {
            family inet {
                rpf-check;
                filter {
                    input filter-internet-in;
                }
            }
        }
    }
    fe-2/0/0 {
        description "Internal segment";
        unit 0 {
            family inet {
                rpf-check fail-filter rpf-dhcp;
            }
        }
    }
}

```



Juniper your Net

2-69

Copyright © 2006 Juniper Networks, Inc.

Proprietary and Confidential www.juniper.net

## RPF Example

In the example on the slide, RPF is enabled in strict mode on all interfaces and only considers the active paths to any prefix. On the internal Fast Ethernet interfaces, it includes a fail filter that permits DHCP and BOOTP. On the Internet-facing interface, a firewall filter is also applied to input traffic. Because RPF provides the antispoofing function, no source-address filters are applied to the firewall filter to prevent spoofing.

## Review Questions

---

1. What is the significance of the *from* and *then* clauses of policies?
2. What is a term?
3. What is a policy chain?
4. What is the significance of the default policy?
5. Explain the difference between prefix lists and route filters.
6. What is the purpose of a unicast RPF check?



### This Chapter Discussed:

- Terms, match conditions, and actions in a policy;
- The policy evaluation process;
- Writing and applying a policy to control routing information; and
- Writing and applying a policy to filter traffic.

## Lab 1: Policy Configuration

---

- Configure per-packet load-balancing
- Use routing policy to control the BGP announcements to a provider
- Protect the router with a stateless firewall filter
- Configure RPF



### Lab 1: Policy Configuration

The slide lists the objectives for this lab.







# **Advanced Juniper Networks Routing in the Enterprise**

## **Chapter 3: BGP**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - Describe three common routing policies used in the enterprise environment
  - Explain the way attribute modifications affect routing decisions
  - Implement a given routing policy for both inbound and outbound traffic using BGP



### This Chapter Discusses:

- Implementing both internal BGP (IBGP) and external BGP (EBGP) sessions;
- The interaction between interior gateway protocols (IGPs) and BGP; and
- Implementing a given routing policy for both inbound and outbound traffic using BGP.

## Agenda: BGP

---

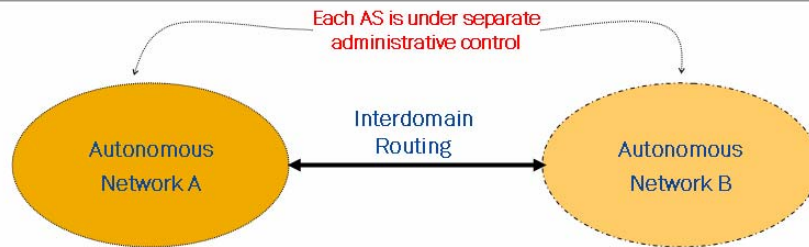
- BGP Overview
- IBGP Implementation
- EBGp Implementation
- BGP-IGP Interaction



### BGP Overview

The slide lists the topics we cover in this chapter. We discuss the highlighted topic first.

## What Is BGP?



- **BGP 4:**
  - Is an interdomain routing protocol
  - Supports CIDR and route attributes that accommodate complex routing policy
  - Is a *path-vector* protocol that uses incremental updates and reliable TCP transport
  - Views the Internet as a collection of autonomous systems
  - Normally requires explicitly defined *peers* for added security and control
  - Is an IETF standard defined in RFC 4271 (supersedes RFC 1771)

### What Is BGP?

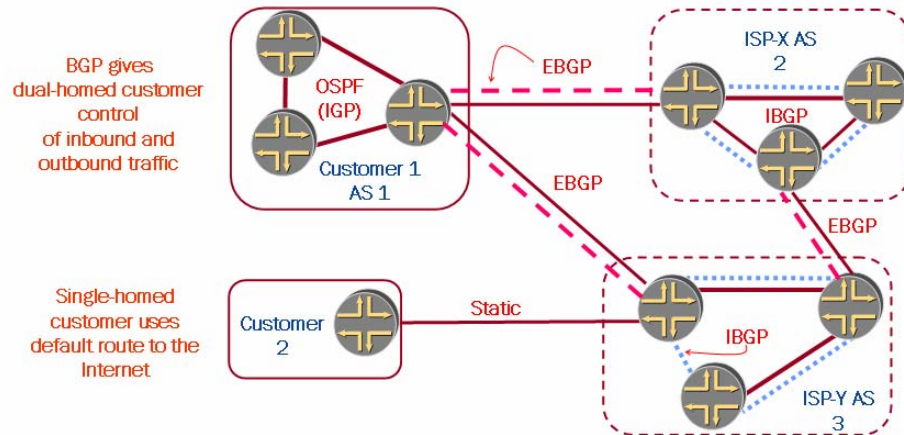
BGP is an interautonomous system (inter-AS) routing protocol and is sometimes called a *path-vector routing protocol* because it uses an AS path, used as a vector, to prevent interdomain routing loops. The term *path vector*, in relation to BGP, means that BGP routing information includes a series of AS numbers, indicating the path that a route takes through the network.

BGP exchanges routing information among ASs or domains. An AS is a set of routers that operate under the same administration. BGP routing information includes the complete route to each destination. BGP uses the routing information to maintain a database of network layer reachability information (NLRI), which it exchanges with other BGP systems. BGP uses the NLRI to construct a graph of AS connectivity, thus allowing BGP to remove routing loops and enforce policy decisions at the AS level.

BGP is a classless routing protocol, which supports prefix routing, regardless of the class definitions of IPv4 addresses. BGP routers exchange routing information between peers. The peers must be connected directly for inter-AS BGP routing (unless certain configuration changes are done).

BGP version 4 (BGP4) is essentially the only exterior gateway protocol (EGP) currently used in the Internet. It is defined in RFC 4271, which made the former standard of more than 10 years, RFC 1771, obsolete.

## When Should I Use BGP?



- Single-homed customers normally use a static default route
- Multihomed customers benefit from BGP route selection intelligence and policy controls

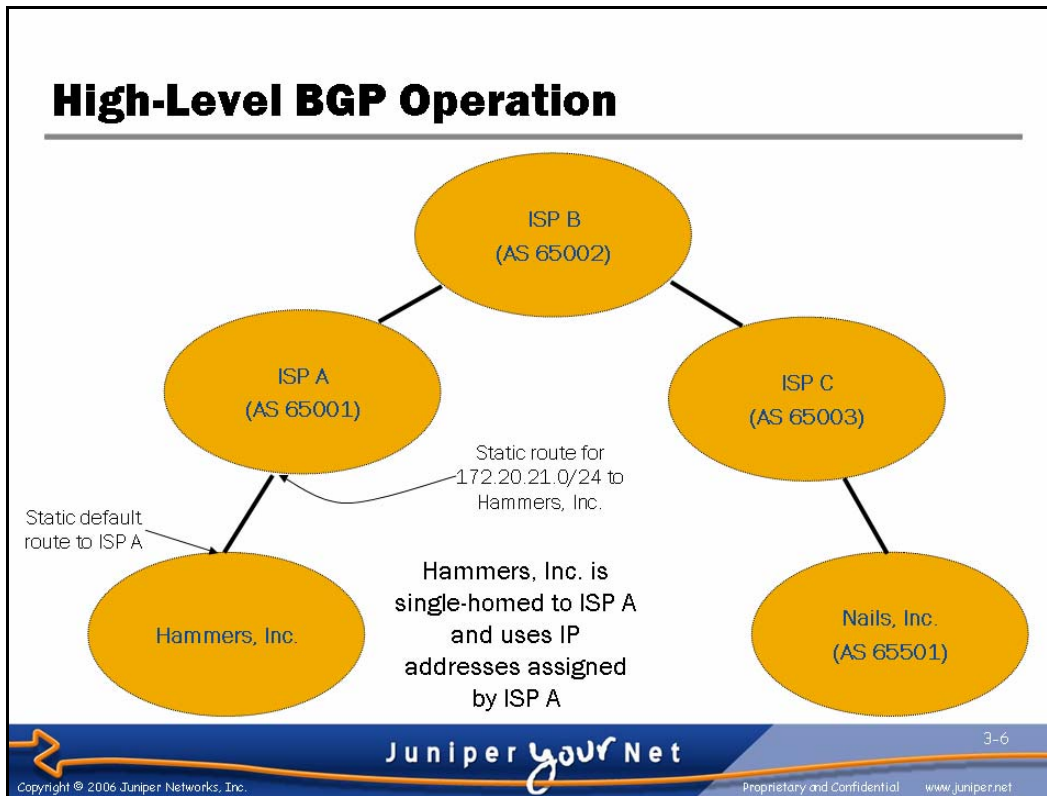
### Single-Homed Networks

Networks with a single upstream connection receive little benefit from running a dynamic routing protocol with their Internet service provider (ISP). These customers typically use a static default route to send all external traffic towards the Internet. Their provider also typically uses a static route to direct traffic destined for the customer's addresses to the customer. Normally, a single-homed network uses addresses assigned by the provider from the provider's aggregate. Because these addresses are assigned to the provider and can only be used by the customer while they are a customer of the provider, they are called *nonportable* addresses. Using these addresses allows the provider to announce a single aggregate route for many customer networks, reducing global routing table growth. (At the time of this writing, the Internet routing table contained more than 180,000 routes.)

### Multihomed Networks

BGP is normally used when a network has multiple upstream connections, either to a single ISP or to multiple ISPs. BGP's policy controls provide the ability to optimize inbound and outbound traffic flows based on a network's technical and business constraints. Like all routing protocols, BGP can also dynamically detect and route around link and node failures.

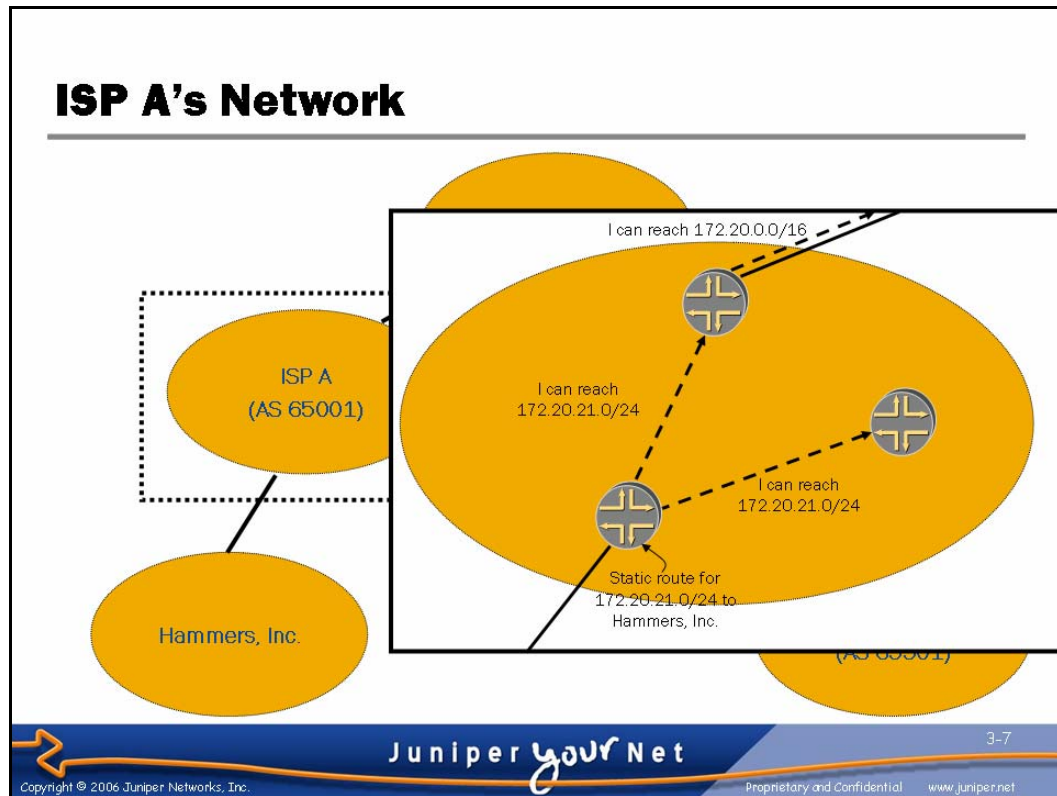
Networks that are multihomed to a single ISP likely use nonportable addresses assigned by the provider. Networks that are multihomed to multiple ISPs likely use portable addresses assigned directly by the regional address registry.



### High-Level BGP Example

The example on the slide explains the operation of BGP at a very high level. Let's consider the way traffic is routed to Hammers, Inc. Hammers, Inc. is a single-homed connection with a single connection to ISP A. ISP A has assigned Hammers, Inc. a prefix (172.20.21.0/24) from its aggregate address range (172.20.0.0/16).

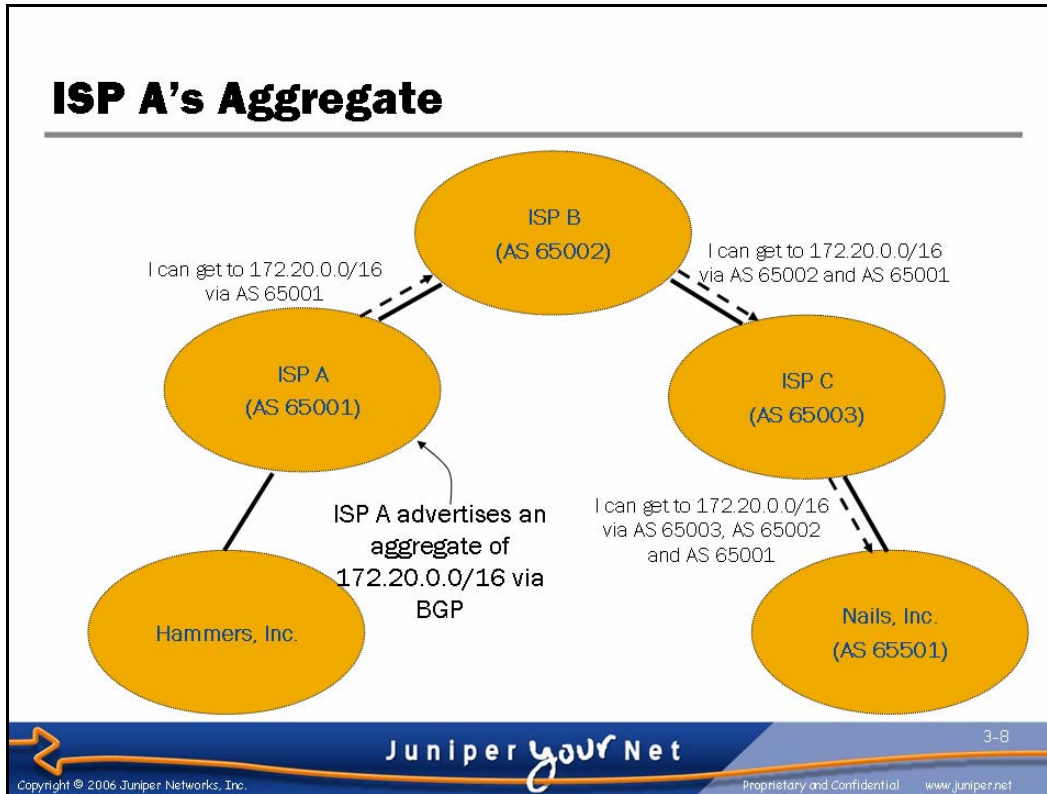
Because Hammers, Inc. has a single-homed network, it has a static default route to reach all destinations on the Internet via its single connection to ISP A. Likewise, ISP A has a static route to reach Hammers, Inc.'s prefix.



### ISP A's Network

The slide highlights a portion of ISP A's network. Internally, ISP A maintains reachability information for each prefix within its aggregate address range. Therefore, every router in ISP A has knowledge about the /24 prefix assigned to Hammers, Inc. This reachability information can be maintained by either an IGP or by IBGP.

Even though ISP A has reachability information about each prefix internally, it only advertises the aggregate prefixes externally. Because other networks use the same path to reach all prefixes available on ISP A's network, other networks do not need the more specific information. To reduce the size of the global routing table, ISPs typically do not transmit the prefixes of their statically routed customers to their peers; rather, they just transmit the aggregate prefixes from which their addresses are assigned.

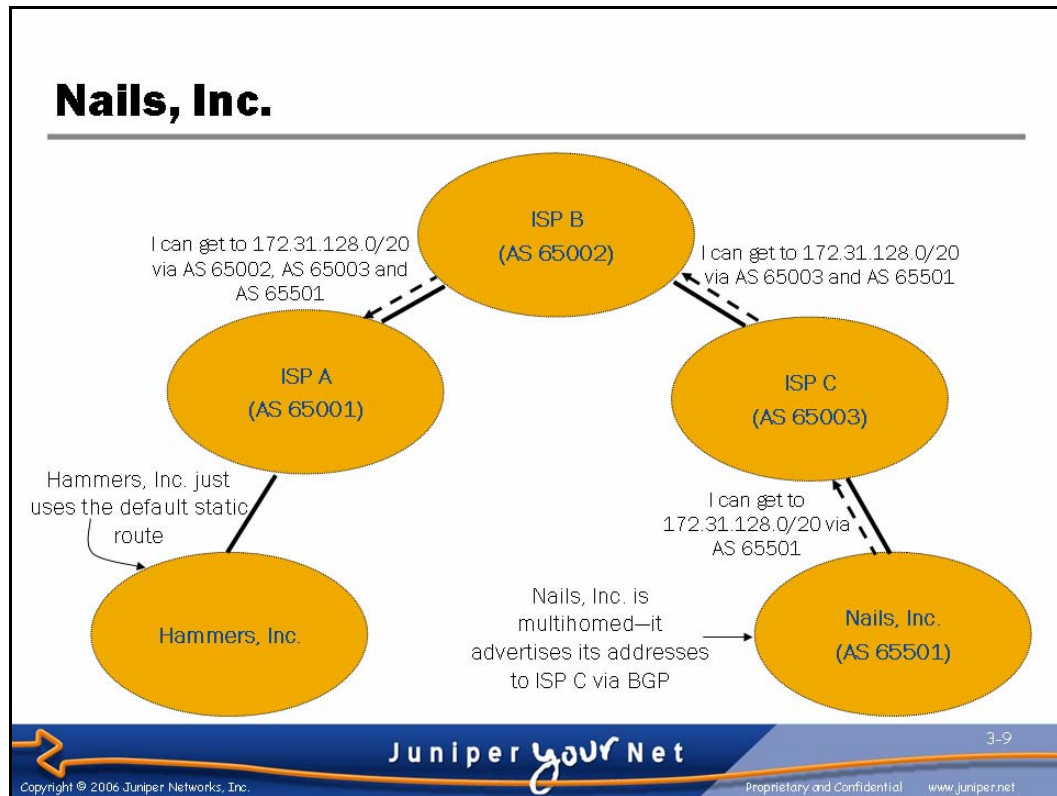


### ISP A Advertises Its Aggregate

ISP A advertises its aggregate address range of 172.20.0.0/16 via BGP along with some information about the path to reach that route. One of these *path attributes* is the AS path, which is a list of the autonomous systems through which the path to this aggregate passes. By examining the AS path, ISP B knows that the 172.20.0.0/16 network was originated within ISP A. (As was mentioned previously, ISP A does not also advertise Hammers, Inc.'s specific prefix.)

ISP B then advertises the 172.20.0.0/16 prefix to ISP C. It updates the path attributes, including the AS path, when it transmits the route. ISP C further advertises this prefix to Nails, Inc., again updating the path attributes when it transmits the route.



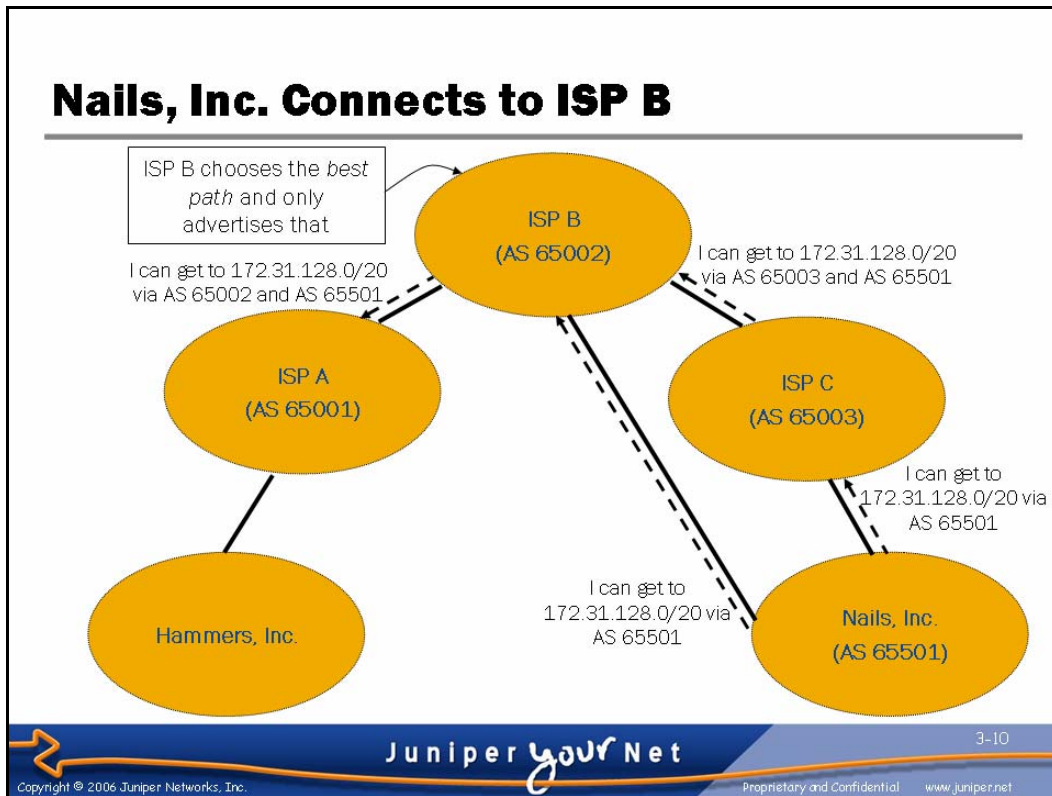


### Nails, Inc.

Nails, Inc. has a multihomed network. For simplicity's sake, we only show its connection to a single ISP on this slide; however, it is multihomed and has a portable /20 prefix assigned directly to it.

It advertises its prefix to ISP C (one of its ISPs) with an AS path indicating that it was originated locally. ISP C sends the advertisement to ISP B, who sends it to ISP A, with each ISP updating the path attributes as it sends the route.

ISP A does not have a BGP session to Hammers, Inc., so Hammers, Inc. does not receive any routing information for Nails, Inc.'s prefix. However, receiving the route information is not necessary because Hammers, Inc. has a static default route that directs all Internet-bound traffic to ISP A. Once the traffic reaches ISP A, ISP A follows the BGP-received route to Nails, Inc.



### Nails, Inc. Becomes Multihomed

Nails, Inc. decides to add a connection to ISP B. Therefore, Nails, Inc. now advertises its prefix to both its providers. In this example, ISP B receives routing information for Nails, Inc.'s prefix both from ISP C and directly from Nails, Inc. ISP B chooses one of the paths as the *best path* and places a corresponding route for the prefix in the routing table. It then advertises the prefix with the associated path attributes to ISP A. Because ISP B chose the path directly to Nails, Inc. as the *best path*, it advertises the path attributes associated with that advertisement to ISP A. Note that it advertises an AS path that reflects that it can directly reach Nails, Inc. and does not include any information about ISP C. Because the path through ISP C was *not* chosen as the *best path*, ISP B does not send ISP A any of the path attributes associated with the advertisement from ISP C.

Should ISP B cease to hear the announcement about Nails, Inc.'s prefix directly from Nails, Inc. (for example, because the circuit fails), it will begin using the path it received from ISP C and will send updated announcements to its peers to reflect the new path.

Although not shown, Nails, Inc. now also receives two advertisements for ISP A's aggregate. It chooses one of those as the *best path* and installs a corresponding route in the routing table.

Now, the question is: *How did ISP B choose the best path?* Truly understanding the answer to that question reveals BGP's flexibility and complexity. We spend a great deal of time in this chapter and the next chapter considering how routers select BGP routes.

## BGP Updates and Attributes

- BGP compares the AS path and other attributes to select the best path
  - Accommodates administrative control over route selection
- A BGP update message can contain one path advertisement and its associated attributes
  - Multiple prefixes can be advertised in one update if they share the same BGP attributes
  - Can also list one or more routes that are no longer reachable



### BGP Is a Policy-Based Routing Protocol

The primary purpose of BGP is *not* to find the shortest path to a given destination; rather, its purpose is to find the *best* path. Each AS determines the best path to a prefix by taking into account its own outbound routing preferences, the inbound routing preferences of the route's originator (as updated by ASs along the path between the source and destination ASs), and some information that is collected about the path itself. All this information is contained in *path attributes* that describe the path to a prefix. The path attributes contain the information that BGP uses to implement the routing policies of source, destination, and transit ASs.

### BGP Update Messages

BGP update messages describe a single path and then list multiple prefixes that can be reached through this same path. BGP peers assume that this information is unchanged unless a subsequent update advertises a new path for a prefix or lists the prefix as unreachable. Updates can list any prefixes that are no longer reachable, regardless of the path associated with those prefixes. BGP peers use update messages to ensure that their neighbors have the most up-to-date information about BGP routes.

BGP uses TCP to provide reliable communication, which ensures that their neighbors never miss an update. A system of keepalives also allows each BGP peer to ensure its neighbor is still functioning properly. If a neighbor goes down, the BGP speaker deletes all routes to the peer and updates its other peers accordingly.

## BGP Active Route Selection Summary

### ■ Selection summary:

1. Can the BGP next hop be resolved?
2. Prefer the highest local-preference value
3. Prefer the shortest AS-path length
4. Prefer the lowest origin value
5. Prefer the lowest MED value
6. Prefer routes learned using EBGp over routes learned using IBGP
7. Prefer routes with the lowest IGP metric
8. Prefer paths with the shortest cluster length
9. For EBGp-received routes, prefer the current active route; otherwise, prefer routes from the peer with the lowest RID
10. Prefer routes from the peer with the lowest peer ID

### Summary of BGP Active Route Selection

When a BGP router learns the same route from multiple BGP speakers, it must decide which route to install into the routing table as an active route. The following steps provide a summary overview of the BGP active route selection algorithm. Note that some details are omitted here in the interest of brevity:

1. The router first must verify that it has a current route in the `inet.0` routing table to the IP address in the BGP next-hop attribute field.
2. The router then compares routes for the *highest* local preference (the only choice based on a higher, rather than lower, value).
3. The router evaluates the AS-path attribute next, where a shorter path is preferred. This attribute is often a common tiebreaker for routes.
4. The router evaluates the origin code. The lowest origin code is preferred.
5. If any of the remaining routes are advertised from the same neighboring AS, the router checks the multiple exit discriminator (MED) attributes for a lowest value. The absence of a MED value is interpreted as a MED of 0.
6. If multiple routes remain, the router prefers any routes learned via an EBGp peer over routes learned via an IBGP peer. If all remaining routes were learned through EBGp, the router skips to Step 9.
7. If the remaining routes were learned through IBGP, the router uses the path with the lowest IGP cost to the IBGP peer.

*Continued on next page.*

## Summary of BGP Active Route Selection

8. The router then examines the cluster-list attribute for the shortest length. The cluster list is similar in function to an AS path.
9. The router prefers the route advertised from the peer with the lowest router ID. However, for EBGp-received routes only, the router prefers the current active route when comparing routes received from different neighboring ASs.
10. The router prefers routes from the router with the lowest peer ID.

Note that the local-preference value (which configures each AS) is more important than the length of the AS path used to get to the destination prefix. This example is an illustration of the way BGP is designed to give great flexibility in enforcing a routing policy, rather than to choose the shortest path.

We return to this path selection algorithm later and use examples to describe the way you can implement inbound and outbound routing policies.

## Agenda: BGP

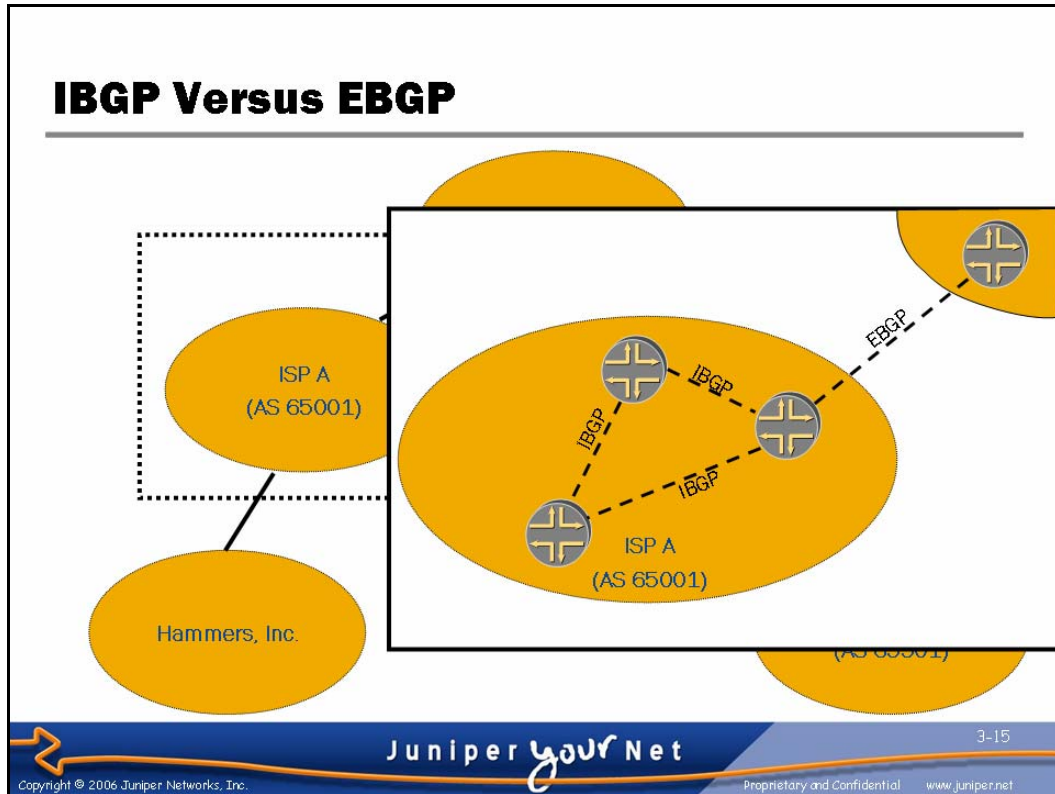
---

- BGP Overview
- IBGP Implementation
- EBGp Implementation
- BGP-IGP Interaction



### IBGP Implementation

The slide highlights the topic we discuss next.



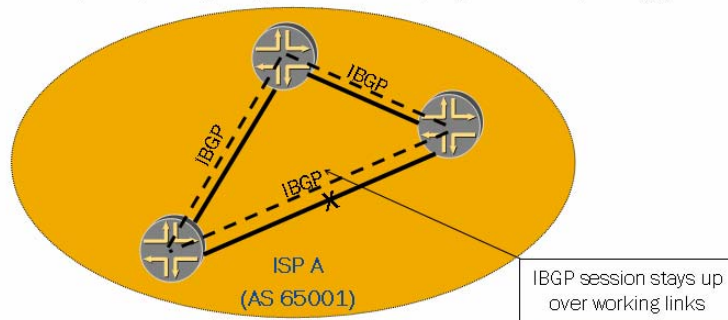
### IBGP Versus EBGP

When you establish a BGP relationship with a peer, your BGP session is either an internal BGP (IBGP) or external BGP (EBGP) session. Determining which kind of session you are establishing is quite simple. If the two peers are in the same AS, the BGP session is an IBGP session. If the two peers are in different ASs, the BGP session is an EBGP session.

The type of relationship affects the path selection algorithm, the way routes are propagated, and the guidelines and constraints that affect session establishment.

## Loopback Peering

- IBGP sessions are usually established between loopback addresses
  - Maintains IBGP session regardless of physical topology
  - Uses IGP to maintain reachability
  - One session per-peer, regardless of physical topology



### Loopback Peering

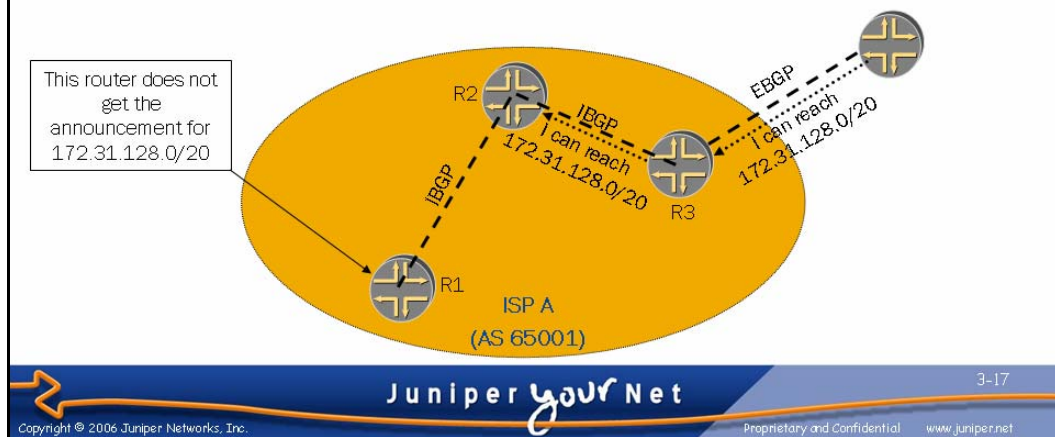
You only maintain one IBGP session between each internal peer. For the most part, IBGP peering sessions really have no connection to the physical topology. The IGP is used to maintain reachability between the loopback addresses regardless of the physical topology, allowing the IBGP sessions to stay up even when the physical topology changes.

The physical topology is relevant in one respect: each router along the path between BGP speakers must have enough information to make consistent routing decisions about packet forwarding. In many cases, this requirement means that all routers along all possible physical paths between BGP speakers must run BGP; however, in some enterprise networks, this requirement is not necessary. We cover this in more detail in a few pages.



## IBGP Route Propagation

- BGP speakers do not propagate IBGP-received routes to other IBGP peers
  - Requires a full mesh of BGP speakers, which provides loop prevention



### IBGP Route Propagation

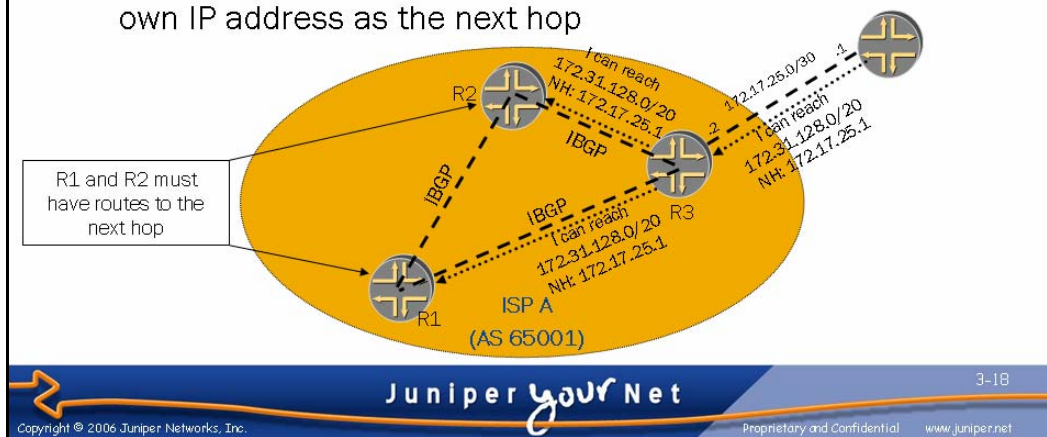
IBGP speakers only send their IBGP peers EBGP-received announcements and announcements they originated themselves. For all BGP speakers in an AS to have consistent routing information, there must be a full mesh of IBGP sessions between all BGP speakers. Without this full mesh, some BGP speakers will end up with incomplete routing information.

In the example on the slide, there is not a full mesh of IBGP sessions. R3 receives the announcement for 172.31.128.0/20 via an EBGP session. Because it is the best route it has for that prefix, it propagates the route to its IBGP peer R2. R2 determines that route to be its best path to 172.31.128.0/20; however, it does *not* send the route to its IBGP peer R1. Because it received the route via IBGP, it cannot send the route to any IBGP peers. Therefore, R1 ends up without any routes to 172.31.128.0/20. This situation can be alleviated by adding an IBGP session between R1 and R3. (It is irrelevant whether a physical path exists between the two routers.)

This situation can also be alleviated by using route reflectors or confederations, both of which can reduce or alleviate the full mesh requirement; however, in most enterprise networks these solutions are unnecessary, so they are beyond the scope of this class.

## IBGP Next-Hop Propagation

- By default, the BGP next-hop attribute is unchanged as a route propagates through an AS
  - Put external interfaces in IGP, or...
  - Use `next-hop self` in a policy to cause the router to use its own IP address as the next hop



### IBGP Next-Hop Propagation

By default, the next-hop attribute attached to a route is unchanged as it passes through an AS. The slide illustrates this default behavior and the design decision this behavior causes. Because routers can only use the BGP routes if they already have a route to the next hop, you must either configure the routers to advertise external interfaces via the IGP or configure the routes to change the next-hop attribute attached to BGP routes.

When EBGP speakers send routes to a peer, they set the next-hop attribute to the interface they share with that peer. In this example, R3 receives routes from its EBGP peer with the next-hop attribute set to 172.17.25.1. R3 sends this route to R1 and R2 without changing the next-hop attribute. Therefore, to use this route, they either know how to reach 172.17.25.1 through the IGP, or R3 must send the routes with a different next hop.

You can send the appropriate external routes into the IGP, if wanted; however, using the `next-hop self` action in a policy has some advantages.

## Changing the Next Hop

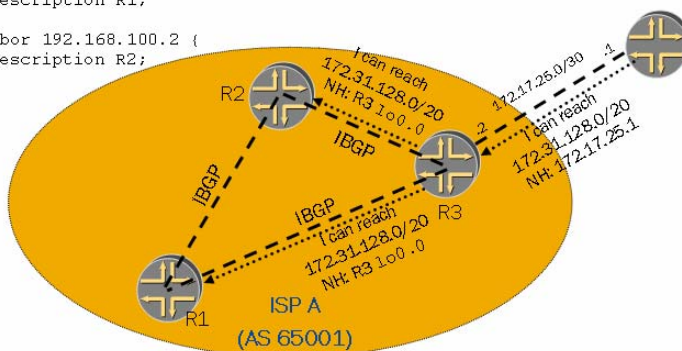
### ■ Example (on R3):

```

protocols {
  bgp {
    group IBGP-peers {
      export next-hop-self;
      peer-as 65001;
      local-address 192.168.100.3;
      neighbor 192.168.100.1 {
        description R1;
      }
      neighbor 192.168.100.2 {
        description R2;
      }
    }
  }
  [...]
}

policy-options {
  policy-statement next-hop-self {
    then {
      next-hop self;
    }
  }
}

```



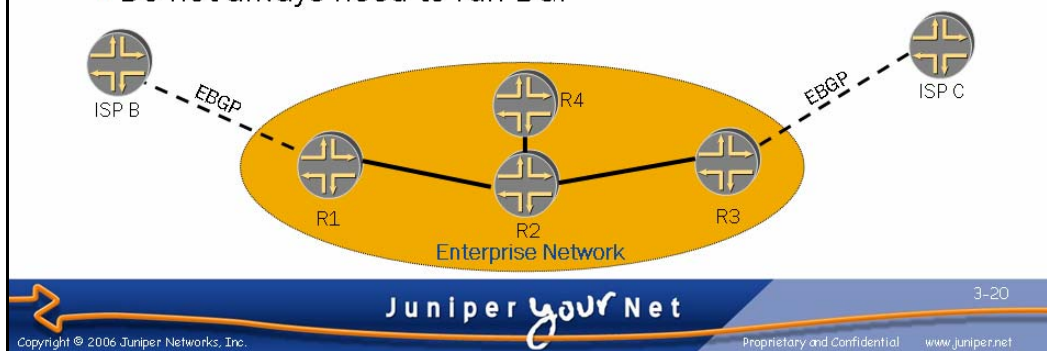
### Using next-hop self

Using the `next-hop self` action in a policy causes the router to send BGP routes to its peers using the same IP address it uses to establish that BGP session. For the BGP session to remain established, the peer must have a route to that IP address. Therefore, using `next-hop self` guarantees that a router's peers will be able to reach the next hop of the routes that router sends, as long as the BGP session remains established.

Using the `next-hop self` action is fairly easy; the slide shows an example.

## Which Routers Should Run BGP?

- Any router which requires an EBGP session
- Any router topologically between BGP speakers that needs BGP information to make forwarding decisions
  - Need enough information to make consistent forwarding decisions without creating forwarding loops
  - Do not always need to run BGP



### Deciding Where to Run BGP

All routers that require an EBGP session must run BGP. Every router running BGP within an AS should be part of a full mesh of IBGP sessions between all BGP speakers within the AS so that they can make consistent routing decisions between them.

### Deciding Where NOT to Run BGP

Deciding what other routers should run BGP is a bit less obvious. In general, you should not run BGP on routers that do not require the information to forward packets appropriately. For example, routers with a single connection to the core of the network can probably do with an IGP-supplied (or even static) default route. However, some routers might need BGP information to forward traffic correctly.

In the diagram on the slide, R1 and R3 must run BGP because they have EBGP peers; however, does R2 need to run BGP as well? The question is whether R2 needs the BGP routing information to make consistent forwarding decisions. For example, assume that this customer wants to use both of its upstream providers. To implement this policy, both R1 and R3 are receiving only a default route from their service provider. They are, in turn, advertising this default route to the network via an IGP. In this scenario, R1 and R3 will both send any Internet-bound traffic that R2 sends them directly to their service providers. So, if R2 follows either (or both) of the default routes, the end result works as intended. Thus, in this particular case, R2 does not need to run BGP.

*Continued on next page.*

### Deciding Where NOT to Run BGP (contd.)

However, if R1 or R3 received more specific routes, R2 would need to run BGP to ensure that consistent routing decisions were made. For example, assume that R1 receives a prefix from its ISP that R3 does not receive from its ISP. R1 tells R3 about this prefix via their IBGP session. R3 tries to send this traffic to R1 via R2; however, if R2 is not also running BGP, it has no way to make an accurate routing decision. It might send the traffic back to R3, causing a routing loop. The best way to resolve that situation is to run BGP on R2 so that all the routers in the AS can make consistent routing decisions.

In general, you should always run BGP on all routers along any possible path between routers that have EBGP sessions. You should only violate this general rule in limited circumstances, where you can guarantee that the intermediate routers can make accurate routing decisions. Usually, there is no need to run BGP on routers that are not along any possible path between routers that have EBGP sessions.

## Agenda: BGP

---

- BGP Overview
- IBGP Implementation
- EBGp Implementation
- BGP-IGP Interaction



### EBGP Implementation

The slide highlights the topic we discuss next.

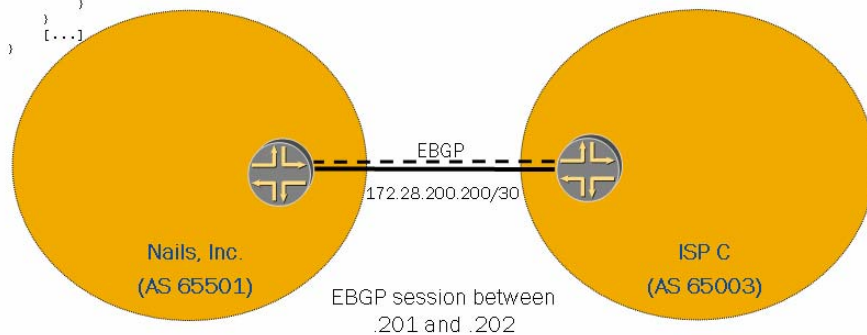
## Interface Peering

- EBGp sessions are usually established between two directly connected addresses

```

protocols {
  bgp {
    group ISPs {
      export to-ISP;
      neighbor 172.28.200.201 {
        description ISP-C;
        peer-as 65003;
      }
    }
    [...]
  }
}

```



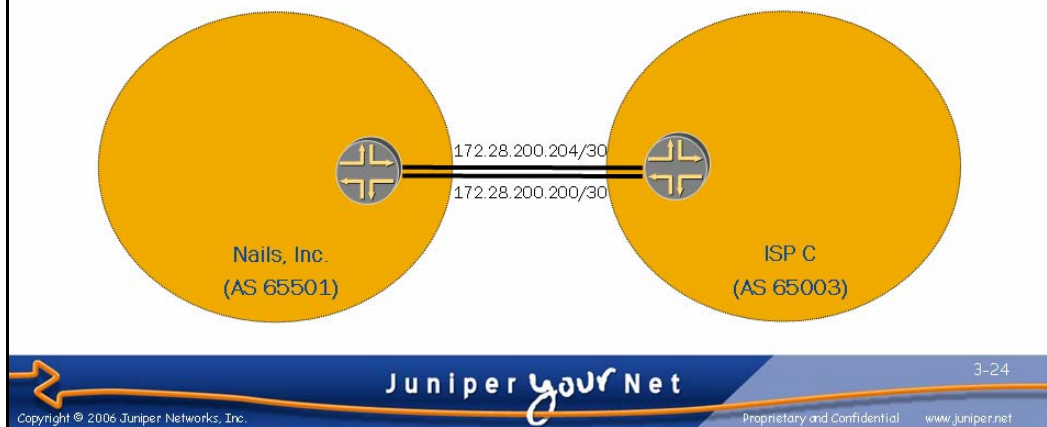
### Interface Peering

Recall that EBGp sessions are simply BGP sessions between two routers in different ASs. When two EBGp peers have a single path between them, EBGp sessions are usually established over the shared subnet between two peers, using the IP addresses assigned to the interfaces on that subnet as the session endpoints. By establishing the EBGp session using the IP address assigned to the interfaces on the shared subnet, you gain many advantages. One of these advantages is that you prevent either AS from needing to maintain any routing information about the other AS (besides what it received through BGP). You also ensure that all traffic flows over this particular shared subnet.



## Multiple Connections Between EBG Peers

- Two options:
  - One EBG session per physical connection + `multipath`
  - One loopback-loopback session + static routes
- Remember, no per-prefix load balancing by default



### Multiple Connections Between EBG Peers

When two EBG peers are connected via multiple paths, two ways exist to connect the EBG peers. You can run multiple EBG sessions, one per connection. When you do this, the BGP path selection algorithm always chooses to use the route received over the EBG session to the peer with the lowest IP address. Usually, when you have multiple connections to a provider, you want to load-share traffic over the connections. If you want to load-share traffic between the multiple connections, you must configure *multipath operation* at the group or neighbor level. When you enable multipath operation, multiple EBG routes from the same next-hop AS tied upon reaching Step 9 of the route selection process described on page 3-12 are installed in the routing table as equal-cost next hops to the prefixes.

The other way to establish EBG connectivity over multiple paths between two EBG peers is to establish an EBG session using the loopback addresses on the two peers as the endpoints. Because neither AS has routing information for the other router's loopback address, you must configure static routes for the loopback address of the peer on each router, using the multiple paths as the next hops. By varying the route preference per next hop, you can enable primary and secondary paths, you can enable equal-cost paths, or you can configure equal-cost primary paths with equal-cost secondary paths.

*Continued on next page.*



## Load Sharing

Remember that JUNOS routers do not load-share over multiple paths by default. To enable load sharing, you must configure an export policy for the forwarding table. We explained this process in “Forwarding Table Policy” on page 2-29.

## EBGP Export Policy

- Unlike IBGP sessions, you usually want to apply an export policy
  - Advertise aggregates
  - Block provider routes
  - Block internal details
  - Implement inbound routing policy
- Export policy should define what you *want* to send, not what you *do not* want to send



### EBGP Policy

You *must not* apply export policies to IBGP peers, because you want to ensure that you have consistent routing information within a routing domain. However, EBGP sessions are an entirely different matter.

You almost *always* want to apply an export policy to control the announcements to EBGP peers. There are several reasons for this. First, the default BGP export policy is to advertise all BGP-received routes to BGP peers. However, in enterprise networks, it is often the case that the very prefixes enterprises want to advertise (the aggregate blocks from which their internal addresses come) are not already known via BGP; rather, they are usually created via an **aggregate** statement, a **generate** statement, or a static discard route. To advertise these prefixes to your EBGP peers, you must use an export policy to advertise them.

Second, a lot of routes exist that you want to block from being sent to EBGP peers. Because the default BGP export policy advertises all BGP-received routes to all BGP peers, it advertises all the routes you receive from each ISP to all your other ISPs. If you have EBGP sessions with partners, you also advertise your ISPs' routes to your partners and your partners' routes to your ISPs. These scenarios are colloquially known as accidentally *leaking* routes. In essence, you could become an ISP for your partner, raising your ISP costs. Or, if you leaked a full routing table from one ISP to another ISP, you could become an ISP for your ISP, overloading your Internet connection and incurring the ire of your ISP, not to mention bringing some embarrassment to your organization. To avoid this situation, you must conduct careful route filtering.

*Continued on next page.*

## EBGP Policy (contd.)

Third, a lot of routes exist that you just do not want to send. In general, you only want to advertise the most specific prefix necessary. For example, many organizations only advertise their aggregate prefixes and do not advertise any more specific routes within those prefixes. If any of the specifics are known via BGP, these routes are accepted by the default BGP export policy, so they must be blocked by a configured export policy.

Finally, the way you establish an inbound routing policy is by modifying the path attributes of the routes you send to your EBGP peers. You can influence the way traffic flows into your network by setting communities, MEDs, and prepending the AS path. You use an export policy to accomplish these settings.

## Explicit Route Definition

For most enterprises, it is best to start an export policy by explicitly defining what the enterprise *should* send. For example, you can define a prefix list that explicitly lists all routes that you want to send. You can then write a policy that rejects all routes except the ones on the list, and include that simple policy as the first policy in a chain of export policies applied to each peer. You can then modify attributes in intervening policies and end the list with a default accept policy, which applies only to the routes not discarded in the first policy (that is, only those that are in the prefix list). Over time, this process will prove to be much more accurate and leave less possibility for error than trying to filter the routes you do not want to send.

## EBGP Import Policy

- Unlike IBGP sessions, you usually want to apply an import policy
  - Implement outbound routing policy
- Generally, you do not want to filter routes from your ISP, but you do want to filter routes from non-ISPs
  - Exceptions:
    - Default-only from ISP
    - Partial routes from ISP
    - Limited resources
    - Transition



### EBGP Import Policy

You generally do not want to apply an import policy to an IBGP session for the same reasons you should not use an export policy on an IBGP session. However, you generally *do* want to apply an import policy to EBGP sessions. You implement your outbound routing policy by configuring an import policy that chooses which prefixes to accept from each BGP neighbor and changes the BGP path attributes. If you do not define an import policy, the router accepts all valid routes and assigns them the default local preference (100). The outbound route selection decision is then based exclusively on the route's path attributes as set by the origin, modified along the path, or both. In some cases, that decision might be acceptable, but often, enterprises want richer outbound traffic policies than that.

### Filtering EBGP Peers

In general, you *do* want to filter routes from partners or other non-ISPs with whom you peer. This prevents them from accidentally beginning to send you routes other than the ones you are expecting to receive (possibly, even a full Internet table) and ensures that you only send them traffic that you have agreed to exchange with them.

In general, you *do not* want to filter routes from your ISPs. Even if you normally expect to follow a different path to a particular prefix, you usually want to retain the route from the ISP to use as a backup.

*Continued on next page.*

## Filtering EBGP Peers (contd.)

Like every rule, this one has some exceptions. In particular, if you only expect to receive a subset of routes from your ISP (for example, only a default route or routes for their customers), you might want to enforce that by rejecting other prefixes to ensure predictable and consistent routing. Additionally, when you have limited resources (particularly, memory), it is important that you enforce restrictions on the number of prefixes your ISP sends you. If you do not, a temporary flood of routes from your ISP could lead to your router running out of memory, causing it to become unstable. You can enforce prefix limits by using the **family inet unicast prefix-limit** configuration statement (at the `protocols bgp group`, or `neighbor` level), filtering the routes they send you, or both. (The **prefix-limit** configuration command is outside the scope of this course, but it should be used with extreme care and full understanding, especially on EBGP sessions to your ISP.)

Also, it is sometimes helpful to filter your ISP's connection during transitional periods. For example, if you transition from receiving only a default route to receiving full Internet routes from your ISP, you might want to temporarily filter the EBGP session with your ISPs to deny everything except the default route. Then, you can ask your ISP to begin sending you both a default route and also a full Internet routing table at their convenience. You can verify that you are receiving (and denying) the full Internet routing table (using the **show route receive-protocol bgp x.x.x.x** command). Then, during your own maintenance window, you can change your filter to deny the default route and accept all other routes. As you monitor the effects of the change during the maintenance window, if things do not work as expected, you can restore the filter and reattempt the transition at a future time. Once you complete the transition, you can have your ISP stop sending you the default route. This method can prove especially useful when the enterprise's and ISP's maintenance windows do not match, or when a change must be made to multiple ISPs' connections at the same time.

In general, when filtering EBGP peers, you always want to ask your peers to filter the routes they send you, but you also want to verify that your peers are filtering the routes they send you by also configuring filters on your own routers.

## Agenda: BGP

---

- BGP Overview
- IBGP Implementation
- EBGp Implementation
- BGP-IGP Interaction

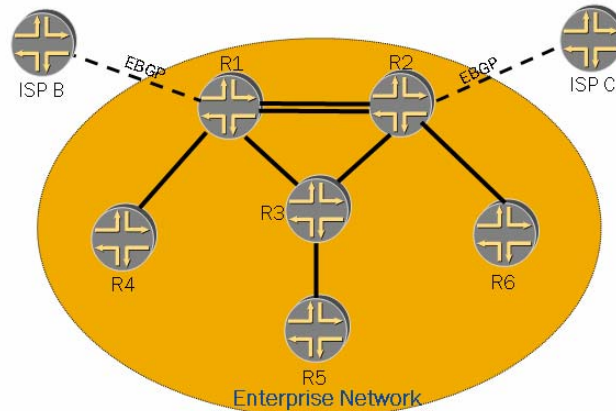


### BGP-IGP Interaction

The slide highlights the topic we discuss next.

## BGP-IGP Interaction Overview

- Problem: You have a few routers with great routing information
  - How do the other routers know about these routes?



### BGP-IGP Interaction: The Problem

Recall that we earlier discussed not needing to run BGP everywhere. However, not running BGP everywhere creates a bit of a problem: because only routers running BGP have external routing information, how do other routers know how to reach external locations? For example, in the diagram on the slide, R1 and R2 have full external routing information. However, how does R3, R4, or R5 get appropriate reachability information?

The exact details of the solution can vary slightly depending on your network's requirements; however, we discuss some general principles here. We also discuss this topic again in some of the examples at the end of this chapter.

## BGP-IGP Interaction—General Principles

- Do not export all BGP routes into the IGP
  - Can export *specific* routes, if necessary
- If multiple paths, use generated routes
  - Only generate routes when local session is up
  - Use least-specific routes possible (for example, default route)
  - Export only generated routes to IGP
  - Purpose is to draw traffic to BGP-speaking core that has full routes



### Exporting BGP Routes into the IGP

In general, you do not want to export BGP routes into the IGP. In particular, you *do not* want to export a full Internet table from BGP into the IGP. Usually, you have a *core* of routers that connects to the ISPs and receive some sort of routing information from the ISPs. These routers, which run BGP, have full routing information and use it to choose the correct exit point for traffic. However, the other routers do not need this routing information; rather, they must only know that they can reach any destination by sending the traffic to this set of BGP-speaking routers. Thus, merely exporting a default route into the IGP suffices to provide the necessary reachability information for the remainder of the network.

The *corner cases* are where the BGP-speaking routers are not close to each other topologically, but rather are scattered throughout the network. If all the ISP connections are concentrated into a core of routers, but a very limited number of prefixes (such as a few partner routes) are available outside of this core, it is acceptable to export the few routes outside the core into the IGP by using a default route to draw the remainder of the traffic into the core. If a significant number of BGP-received routes are outside the core, or the ISP connections are so scattered that there really is no core, you should consider running BGP on most of the multihomed routers in the network. However, in no case should you export a full Internet table into the IGP.

*Continued on next page.*



## Route Generation

When all the connections to the Internet are on a single router, you can simply configure a static default route on the border router and export that route into the IGP. However, when Internet connections are on multiple routers, it is ideal to have the routers only advertise a default route when they have a working connection to the Internet available locally, which you can accomplish with the use of generated routes.

You can configure the router to generate a route when a certain set of conditions is met. In this case, we look for BGP-received routes from our neighbor. Ideally, we configure the router to only generate a default route when it receives routes from an EBGp peer and when those routes have an AS path that indicates that they traversed one or two ASs beyond the EBGp peer. By having the router look for these conditions, we ensure that the router only generates a default route when our ISP's connection to the Internet is healthy, too.

You can also use generated routes when you desire a primary/secondary configuration, which you can accomplish by modifying metrics on the generated route when you export it to the IGP. If the primary router loses its connection to the Internet, it stops creating the generated route, and the IGP begins using the secondary route. Note that you should closely coordinate your IGP and BGP configurations to match your outbound routing policy. Thus, if your outbound routing policy is to have a primary/secondary relationship, you should establish this relationship both in BGP and in your IGP configuration. On the other hand, if your outbound routing policy is to load-share, you should establish this policy both in BGP and your IGP.

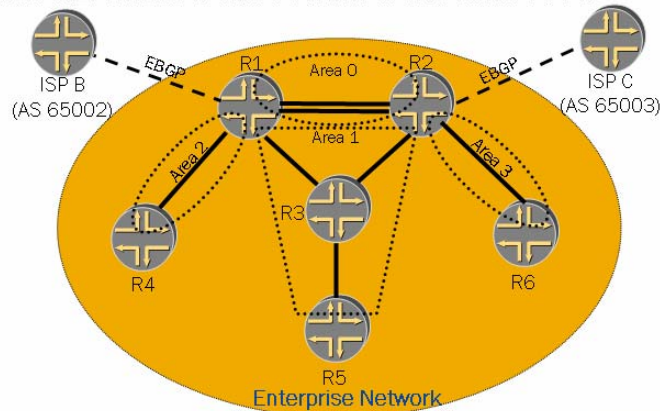
In general, you want to export a default route to the IGP. In rare cases, you might want to encourage some load sharing by establishing a primary/secondary relationship for large blocks. For example, you can have one router advertise 0.0.0.0/1 to the IGP with a low metric and advertise 128.0.0.0/1 into the IGP with a higher metric. You can then have another router do the opposite: advertise 0.0.0.0/1 with a high metric and 128.0.0.0/1 with a low metric. This configuration encourages traffic to be split between the two routers within the AS; however, once the traffic arrives at these routers, it follows the path dictated by the BGP routes. Therefore, it does not necessarily guarantee that traffic will be split across external links.

Generated routes are similar to aggregate routes except that they automatically inherit the next hop of the primary contributing route. This route is the contributing route with the lowest route preference (most preferred) and the lowest numerical prefix value.

## BGP-IGP OSPF Example: Overview

### ■ Overview of example:

- Traffic flow: R1-R2
  - No requirement to support R1-R3-R2
- R1 and R2 receive full routes from their ISPs

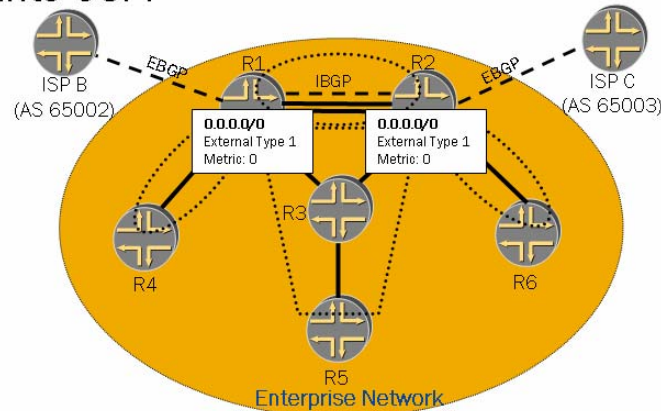


### BGP-IGP Interaction: OSPF Example Overview

In the example on the slide, we show generating default routes in an OSPF network. This diagram shows the EBGP sessions and the OSPF areas. The dual links between R1 and R2 are physically diverse. There is no requirement for traffic to travel between R1 and R2 through R3. For this example, we assume that both R1 and R2 are receiving full routes from their ISP.

## BGP-IGP OSPF Example: Solution

- IBGP between R1 and R2
- Both R1 and R2 generate default routes and export them into OSPF



### IBGP Setup

Because both R1 and R2 have EBGP sessions, they must run BGP. Because traffic between R1 and R2 will never flow through R3, there is no requirement that it have full routing information, and it therefore does not need to run BGP. Thus, only one IBGP session is needed: an IBGP session between R1 and R2.

### Default Route Generation

On the slide, both R1 and R2 generate default routes when they receive BGP routes from their directly connected ISPs. They export them to OSPF as Type 1 external routes (so the metric increments along the path) with a starting metric of 0. In this example, R3 ends up with equal-cost default routes, and we must configure a forwarding table filter to load-share traffic using both connections.

If, instead, we want to establish a primary/secondary traffic flow for outbound traffic, we can have either router export its route with a higher metric. (In this case, an OSPF Type 2 external route is likely more appropriate because it causes the lower-metric route to always be chosen, regardless of topology.)

## BGP-IGP OSPF Example: Configuration

### ■ R1 sample configuration excerpts:

```

routing-options {
    generate {
        route 0.0.0.0/0 {
            policy [ ISPB-routes reject-all ];
        }
        [...]
    }
    protocols {
        bgp {
            group isp {
                [...]
                neighbor 172.17.55.45 {
                    description ISP-B;
                    peer-as 65002;
                }
                [...]
            }
            ospf {
                export default-to-ospf;
                [...]
            }
        }
    }
}

policy-options {
    policy-statement ISPB-routes {
        term match-ISPB-routes {
            from {
                protocol bgp;
                neighbor 172.17.55.45;
            }
            then accept;
        }
    }
    policy-statement default-to-ospf {
        term match-default {
            from {
                protocol aggregate;
                route-filter 0.0.0.0/0 exact;
            }
            then {
                metric 0;
                external {
                    type 1;
                }
                accept;
            }
        }
    }
    policy-statement reject-all {
        then reject;
    }
}

```



### Configuration Excerpt

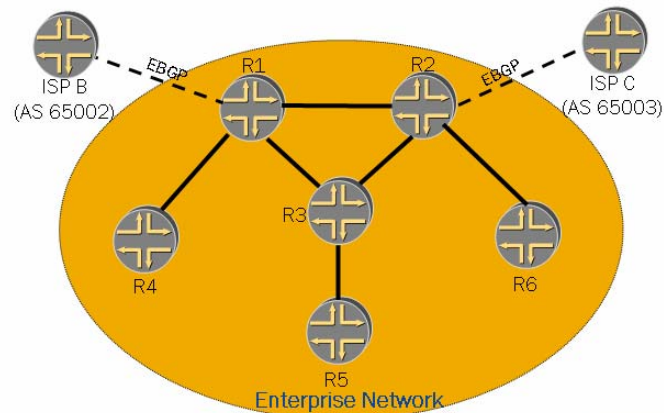
The slide shows a sample configuration excerpt from R1. The sample configuration has been trimmed to only show the relevant parts. The `generate` statement causes a route to be created when the routing table contains any active routes that match the policy. The generated route will have a next hop of ISP B.

The `ISPB-routes` policy matches all BGP routes received from ISP B that are active in the routing table, while the `reject-all` policy causes all other routes to be ignored.

The `default-to-ospf` policy causes the generated route to be exported to OSPF. This policy matches `protocol aggregate` because generated routes are treated like aggregate routes.

## BGP-IGP RIP Example: Overview

- Overview of example:
  - R1 and R2 receive full routes from their ISPs
  - R1 and R2 traffic flow

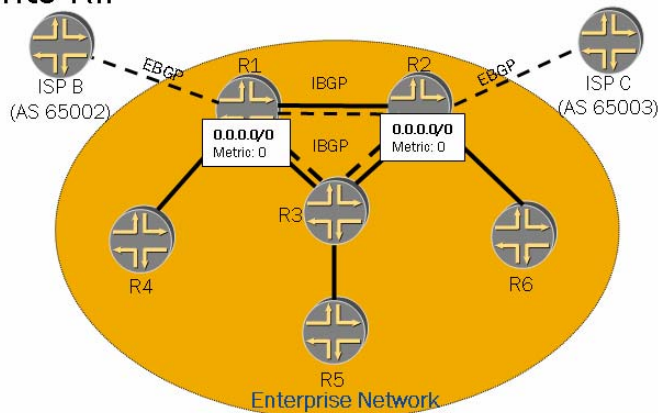


### BGP-IGP Interaction: RIP Example Overview

This example is nearly identical to the previous example, except the IGP in this example is RIP, and redundant R1-R2 links no longer exist. Therefore, a requirement is that the R1-R3-R2 path be available as a backup to the direct R1-R2 link.

## BGP-IGP RIP Example: Solution

- IBGP between R1, R2, and R3
- Both R1 and R2 generate default routes and export them into RIP



### IBGP Configuration

Because both R1 and R2 have EBGP sessions, they must run BGP. Because traffic between R1 and R2 might flow through R3, R3 must be able to make routing decisions consistent with R1 and R2, so it must also run BGP. Therefore, three IBGP sessions are needed: one between R1 and R2, one between R1 and R3, and one between R2 and R3.

### Default Route Generation

In this example, both R1 and R2 generate default routes when they receive BGP routes from their directly connected ISPs. They export them to RIP with a metric of 0. In this example, R3 ends up with equal-cost default routes through RIP; however, if it has more specific routes via BGP, it is probably not necessary to configure the router to load-balance this default route because it makes routing decisions on a per-prefix basis using the more specific information available via BGP. (Note that R3 makes more accurate routing decisions in this configuration because it always sends traffic to the router attached to the ISP that BGP would choose as the preferred exit for that given destination.)

If, instead of load-sharing traffic between R1 and R2, we want to establish a primary/secondary traffic flow for outbound traffic, we can have either router export its route with a higher metric.

## BGP-IGP RIP Example: Configuration

### ■ R1 sample configuration excerpts:

```

routing-options {
  generate {
    route 0.0.0.0/0 {
      policy [ ISPB-routes reject-all ];
    }
    [...]
  }
  protocols {
    bgp {
      group isp {
        [...]
        neighbor 172.17.55.45 {
          description ISP-B;
          peer-as 65002;
        }
        [...]
      }
      rip {
        group peer-routers {
          export [ default-to-rip rip-routes ];
          [...]
        }
      }
    }
  }
}

policy-options {
  policy-statement ISPB-routes {
    term match-ISPB-routes {
      from {
        protocol bgp;
        neighbor 172.17.55.45;
      }
      then accept;
    }
  }
  policy-statement default-to-rip {
    term match-default {
      from {
        protocol aggregate;
        route-filter 0.0.0.0/0 exact;
      }
      then {
        metric 0;
        accept;
      }
    }
  }
  policy-statement reject-all {
    then reject;
  }
}

```



### Configuration Excerpt

The slide shows a sample configuration excerpt from R1. The sample configuration has been trimmed to only show the relevant parts. The `generate` statement causes a route to be created when the routing table contains any active routes that match the policy. The generated route will have a next hop of ISP B.

The `ISPB-routes` policy matches all BGP routes received from ISP B that are active in the routing table, while the `reject-all` policy causes all other routes to be ignored.

The `default-to-rip` policy causes the generated default route to be exported to RIP. The `rip-routes` policy (not shown) matches and accepts RIP-received routes. (This policy is necessary because RIP's default policy does not accept any routes, including RIP-received routes.)

## Review Questions

1. What is the difference between IBGP and EBGP?
2. What are the advantages of loopback peering for IBGP sessions?



### This Chapter Discussed:

- Implementing both internal BGP (IBGP) and external BGP (EBGP) sessions;
- The interaction between interior gateway protocols (IGP)s and BGP; and
- Implementing a given routing policy for both inbound and outbound traffic using BGP.





# **Advanced Juniper Networks Routing in the Enterprise**

## **Chapter 4: Enterprise Routing Policies**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - Describe three common routing policies used in the enterprise environment
  - Explain the way attribute modifications affect routing decisions
  - Implement a given routing policy for both inbound and outbound traffic using BGP



### This Chapter Discusses:

- Three common routing policies used in the enterprise environment;
- The way attribute modifications affect routing decisions; and
- Implementing a routing policy for both inbound and outbound traffic using BGP.

## **Agenda: Enterprise Routing Policies**

---

- Enterprise BGP Deployment
- Case Study: Primary/Secondary Routing Policy



### **Enterprise BGP Deployment**

This slide lists the topics we discuss in this chapter. We discuss the highlighted topic first.

## Common Enterprise Routing Policies

- Inbound routing policies:
  - Topology driven
  - Primary/secondary
  - Load-shared per prefix
- Outbound routing policies:
  - Topology driven
  - Primary/secondary
  - Load-shared per prefix



### Common Enterprise Routing Policies

Many different possibilities exist for inbound and outbound routing policies, but most enterprise routing policies fall into roughly three categories: topology driven, primary/secondary, and load-shared per prefix. We explain these categories in following pages.

### Inbound and Outbound Routing Policies

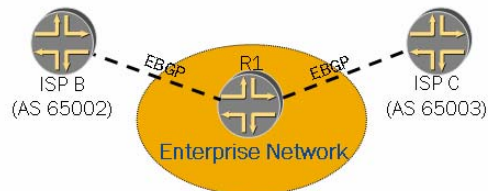
Organizations choose routing policies to balance many interests, among them performance, reliability, and cost. Those decisions are generally policy decisions made by managers. It is the job of the network engineers to choose the correct routing policies to implement those policy decisions.

It is possible to use different inbound and outbound routing policies to fill the needs of the organization. For example, you can use a topology-driven outbound routing policy while using a load-shared per-prefix inbound routing policy.

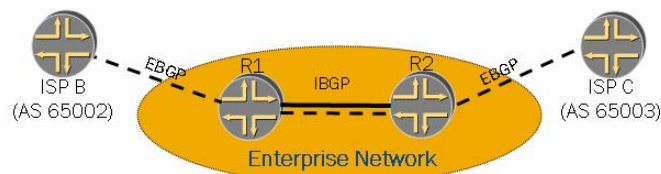
Note that you can only influence—but not fully control—the way inbound traffic reaches your AS. Because BGP is a policy-driven protocol and each AS controls its own outbound routing policy, you can again only influence—but not control—the routing decisions other ASs make. To produce the desired result, you must think about your announcements from the perspective of other ASs (both ISPs and other enterprises). You must think about the likely ways the path attributes you attach to your announcements will affect their routing decisions.

## Common Enterprise Topologies

- Multiple providers, one router:



- One or more providers, multiple routers:



### Common Enterprise Topologies

Enterprise networks that have only a single ISP connected to a single router generally do not need to worry too much about routing policy. (In fact, they really do not even need to run BGP.) Other enterprise networks can be summarized by one of the topologies shown on the slide. The effect of outbound routing policies might differ slightly depending on which of the topologies you have.

### Routing Policies and Reference Topologies

In the following slides, we look at each of the routing policies, describe the effect of that routing policy on the topologies shown on the slide, and discuss how to implement it in the topologies.

## Common ISP Routing Policies

- Use local preference to prefer certain routes
  - Usually prefers customer routes over peer routes by default
  - Usually can be modified by use of communities
- Filter all routes by length
  - Usually do not accept routes  $> /24$
  - Can accept *long* routes from customers but cannot announce them
- Filter customer routes by prefix, AS path, or both
  - Sometimes automatic (routing registry), sometimes manual



### Local Preference

There are many practices that are fairly common among ISPs. Understanding these practices can affect the way we implement routing policies. Among these practices is the use of local preference to prefer certain routes over others. Usually, ISPs assign default local preferences such that customer routes are preferred over routes received from peers (or upstream ISPs). ISPs also usually accept certain communities from customers that cause them to set a different local preference. Usually, several options are available between the default customer and peer values, and usually, at least one option is available that is less than the default peer value. We can use these communities to further influence inbound traffic flow.

### Prefix Length

Most ISPs does not accept routes that are more specific (longer) than a  $/24$  from either customers or peers. Some ISPs accept longer routes from customers, but they do not announce them to other customers or peers.

### Filter Customer Routes by Prefix, AS Path, or Both

It is almost a universal practice among ISPs to apply rather strict import filters to their BGP sessions with customers. Most filters match on prefixes. Some filters match the AS path (either in addition to, or instead of, a prefix match).

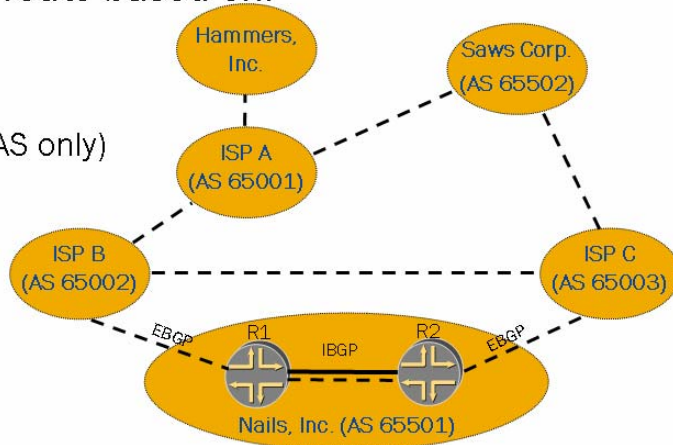
*Continued on next page.*

### Filter Customer Routes by Prefix, AS Path, or Both (contd.)

Those filters that match on prefixes can perform an `exact` match, an `orlonger` match, or an `upto` match. Some ISPs provide automated mechanisms for updating these filters, while some rely on a manual process. You should understand your ISP's filtering policies because they might impact the way you implement a routing policy.

## Topology-Driven Routing Policies

- Accept all routes without attribute modification
- Choose an active route based on:
  - AS path
  - Origin
  - MED (same next AS only)
  - Closest exit



### Topology-Driven Routing Policies

In these routing policies, all routes are accepted without attribute modification. Thus, the BGP path selection algorithm looks primarily at topological factors (such as AS path, MED, and the IGP metric) to determine the best route to send the traffic. This model should generally produce the best performance but is only appropriate when you have no preference about the way traffic enters or exits your AS.

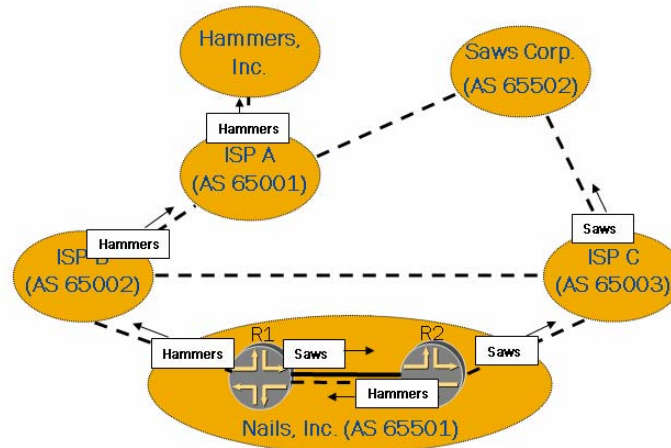
### Path Selection

In this model, the router first tries to determine the best path for traffic to reach its destination. The router first looks at the AS path length, which should give some indication of the total distance to the destination. If that length is tied between routers from the same next-hop AS, it compares MEDs to determine which path the next AS wants it to use (hopefully, the next AS is setting its MEDs to accurately reflect internal metrics).

If the router determines that two or more BGP-received routes are still *tied* in these values, the router must break the tie between these equally well preferred routes. Once it reaches this point in the decision algorithm, it simply tries to choose the closest exit. It does this by preferring EBGP-received paths over IBGP-received paths. If there are no EBGP-received paths, it next tries to break the tie by choosing the IBGP path with the closest exit, as determined by IGP metrics. (See Steps 6 and 7 in “Summary of BGP Active Route Selection” on page 3-12.)



## Topology-Driven Outbound Routing Policy

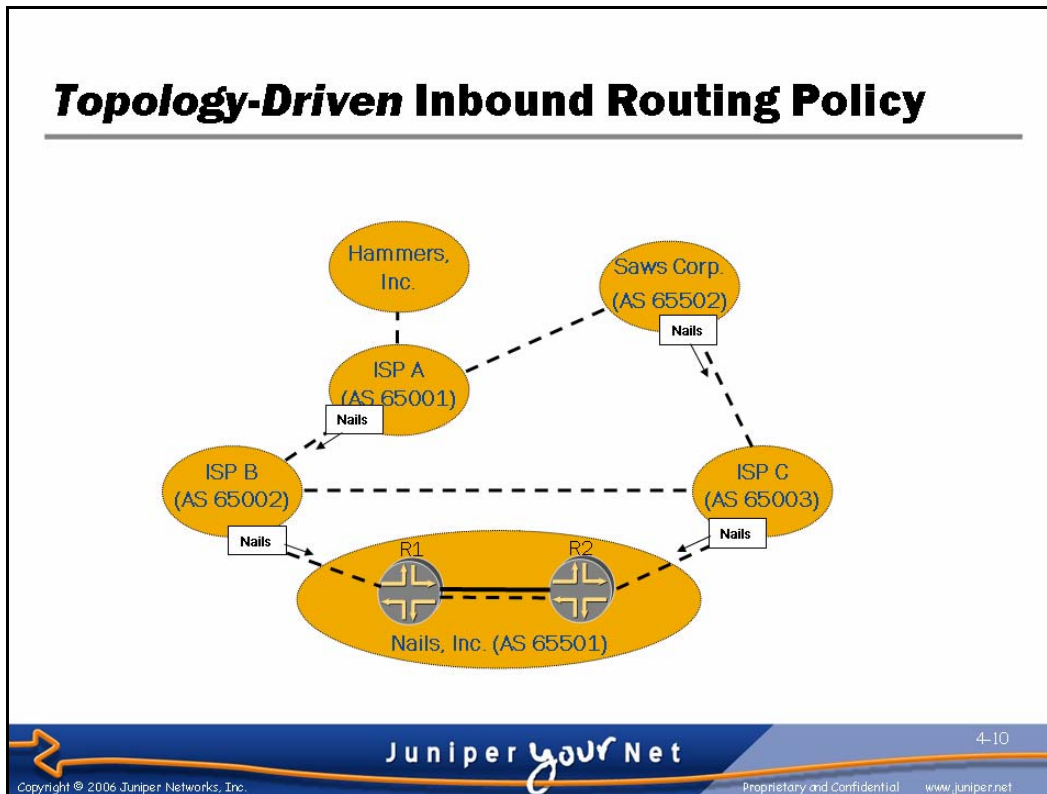


### Outbound Policy

This slide shows an example of a *topology-driven* outbound routing policy. We assume that Nails, Inc. is receiving a full Internet routing table from both providers.

In this example you see that AS 65501 prefers to send traffic to the topologically closest exit. Note that the routers in AS 65501 make consistent exit decisions, causing R1 to send all traffic destined for Saws to R2, and R2 to send all traffic destined to Hammers through R1. Thus, outbound traffic might flow between R1 and R2, which might raise the bandwidth requirements for this connection.

Also, note that the quality of the ISPs involved is important in this model. If R1 is connected to a tier 2 or 3 ISP while R2 is connected to a tier 1 ISP, the likelihood is that many paths will be shorter through the ISP connected to R2. This scenario would result in most traffic exiting the network via R2, essentially creating a primary/secondary configuration.



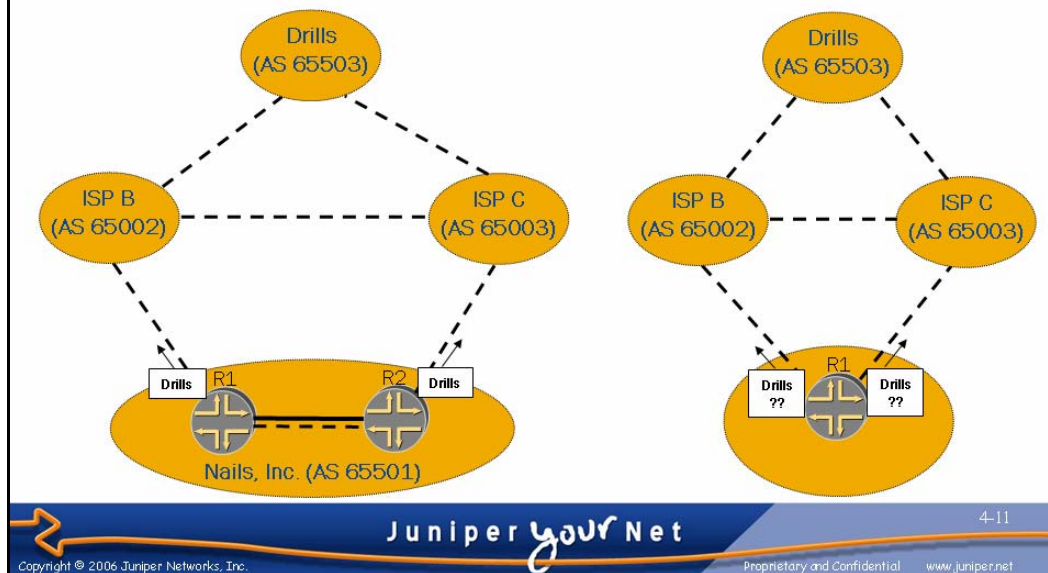
### **Inbound Routing Policy**

This slide shows a *topology-driven* inbound routing policy from Nails, Inc.'s perspective. Nails, Inc. advertises its prefixes without any modification. Assuming the other networks shown are also using a topology-driven outbound routing policy, the result is shown on the slide.

Remember that you cannot *control* inbound routing, but only *influence* it. Therefore, note that Saws Corporation could decide to send its traffic via ISP A, if it wanted, which would result in all the traffic from both Hammers and Saws arriving via R1.

## Multiple Versus Single Routers

- When all else is equal, routers choose the closest exit



### Path Selection

If two paths to a destination have equal AS path lengths and MED values (if the next AS is the same), the router simply tries to choose the closest exit from the AS. Once it gets to this point, one of the differences between having multiple border routers and a single border router appears.

In the case of multiple border routers, each border router prefers one of the EBGP-received announcements over any of the IBGP-received announcements. Thus, traffic bound for Drills that reaches R1 is sent via ISP B, and traffic bound for Drills that reaches R2 is sent via ISP C. This scenario should result in load-sharing, depending on the IGP configuration.

In the case of a single border router, the router prefers the oldest EBGP-received announcement, which results in the router preferring the most stable path to a destination. However, in periods of great stability (or immediately following a period of high instability, such as a router reboot), it also generally results in the router preferring a single connection for most routes with an equal AS path. This behavior results in somewhat imbalanced load sharing; however, this behavior should result in better performance over time (because it favors stable paths).

## Primary/Secondary Routing Policies

- Multiple variations:
  - *Strict* primary/secondary: always prefer the primary connection unless it is down, no traffic over secondary connection
  - *Loose* primary/secondary: prefer the primary connection unless it is down, except allow certain traffic over the secondary connection
- Equal-bandwidth links and *strict* primary/secondary provide assured redundancy



### Primary/Secondary Variations

A true primary/secondary routing policy has a single preferred ISP, over which all traffic flows while the connection to that ISP is up. Only when that connection is down does traffic flow over the secondary connection.

Variations on the primary/secondary routing policy do exist that allow certain traffic to flow over the secondary link, even when the primary link is up. For example, you might want to allow traffic to or from customers of the secondary ISP to use that connection even when the connection to the primary ISP is functioning normally.

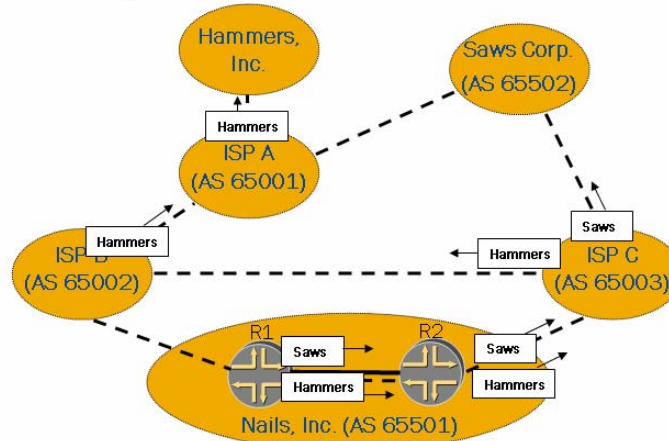
### Assured Redundancy

The main benefit to a primary/secondary routing policy is to provide reliability through redundancy. To be assured of *true* redundancy, you must always have enough bandwidth available to the secondary provider to fully accept the load on the primary provider. The easiest way to guarantee this availability is to use connections of the same bandwidth to both the primary and secondary providers and to use a *strict* primary/secondary routing policy. This method ensures that you always have enough bandwidth in place on the secondary link to fully accept the load on the primary link.

Implementing a strict primary/secondary model can also allow you to reduce the costs of the secondary connection. It is usually possible to negotiate a lower monthly fixed fee for a secondary circuit in return for accepting a usage-based bill. Provided that there are not extended outages to the primary circuit, this fixed fee will likely result in a lower cost for the secondary circuit than if you purchased two circuits and used them both.

## Primary/Secondary Outbound Routing Policy

- Enforce *correct* exit via local preference
- Use for single and multiple border routers



### Local Preference

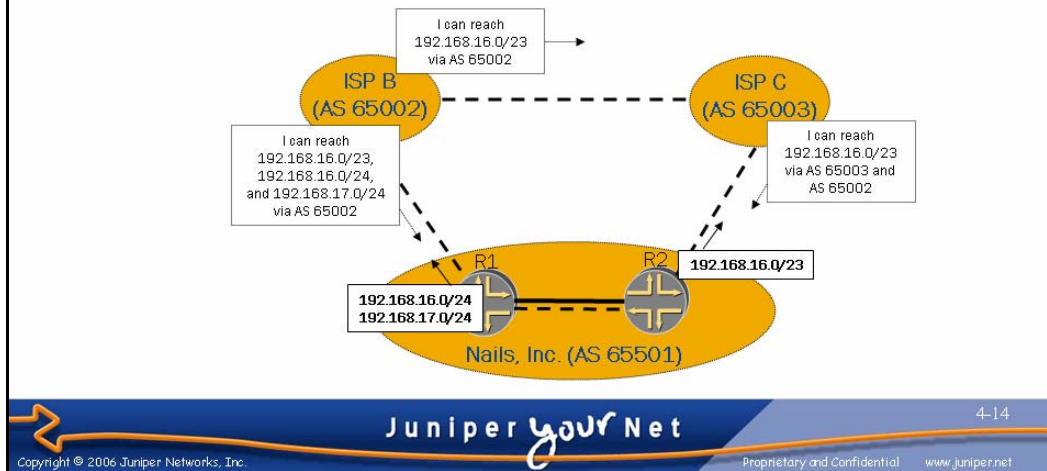
The best way to enforce a primary/secondary outbound routing policy is by modifying the local-preference path attribute. The router always selects the route with the highest local preference (regardless of AS path) as its preferred route to a destination. Therefore, setting a higher local preference on the routes using the primary path and a lower local preference on the routes using the secondary path should cause the router to always prefer the routes using the primary path. However, this setting does not necessarily mean that all traffic will flow through the primary path.

### Single Versus Multiple Border Routers

The primary/secondary configuration is the same whether you have a single border router or multiple border routers. The routers communicate local-preference values between them, resulting in a consistent routing decision.

## Strict Primary/Secondary Outbound Routing Policy

- To enforce strict primary/secondary, receive only a default



### More Specific Routes

It is very typical for ISPs to have slightly different sets of routes in their routing tables. For a variety of reasons (prefix length filtering, aggregation, and so forth), it is common for each provider to have a handful (or more!) more-specific routes that other providers do not have. In the example on the slide, ISP B is sending both a /23 prefix as well as the two more specific /24 prefixes to AS 65501. ISP C is only receiving a /23 aggregate prefix from ISP B and, therefore, is only sending the /23 prefix to AS 65501.

Because AS 65501 is configured to assign a higher local preference to routes received from ISP C, both R1 and R2 prefer the announcement for the /23 prefix that is being received from ISP C. However, because AS 65501 only has one announcement to each of the /24 prefixes, it prefers that announcement and installs routes to send traffic for the two /24 prefixes to ISP B, which effectively results in ISP B being the primary provider for the /23 prefix.

To avoid this situation and to enforce a strict primary/secondary outbound routing policy, it is best to only receive a default route from each provider. Receiving the default route via BGP (instead of configuring a static route) causes the route to disappear when the link to your ISP becomes unusable. Additionally, this configuration matches your intention of always following the path to the primary ISP when it is available.

*Continued on next page.*

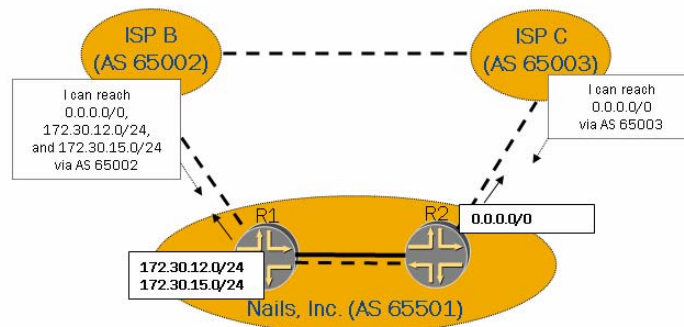
### More Specific Routes (contd.)

If you choose to receive only default routes, you lose a small amount of reliability. If AS 65501 receives full routes and instability exists within ISP C's network, ISP C *might* stop advertising some specific routes, at which point AS 65501 would begin using the path through ISP B (provided it is not experiencing the same instability). However, in this scenario, there is no guarantee that ISP C will withdraw any announcements. (There have been examples where ISPs have continued to advertise full Internet routing tables.) As usual, this scenario comes down to a trade-off between reliability and cost: the cost of sending small amounts of traffic to the secondary ISP versus the small reliability gain.



## Loose Primary/Secondary Outbound Policy

- To allow loose primary/secondary, receive a default from both the primary and secondary provider, but also allow specific routes from the secondary that you want to prefer (such as customers of that ISP)



### Loose Primary/Secondary

In a *loose* primary/secondary routing policy, you make one ISP your primary ISP and another your secondary ISP, but you want to allow traffic to select prefixes to use your secondary ISP even in normal operations.

To implement a loose primary/secondary outbound routing policy, you accept a default route from both providers, but you also accept announcements for some specific prefixes from your secondary ISP. Because you do not receive announcements for those specific prefixes from your primary ISP, your routers use the announcements you receive for these prefixes from your secondary ISP.

In the example on the slide, AS 65501 is receiving the specific routes for some of ISP B's customers. AS 65501 is preferring to send traffic to these prefixes through ISP B.



## Primary/Secondary Outbound: Configuration Example—Strict

### ■ *Strict* sample configuration excerpts:

```

protocols {
    bgp {
        group primary-isp {
            [...]
            import [ localpref-80 default-only ];
        }
        [...]
        group secondary-isp {
            [...]
            import [ localpref-70 default-only ];
        }
        [...]
    }
}

policy-options {
    policy-statement localpref-80 {
        then {
            local-preference 80;
        }
    }
    policy-statement localpref-70 {
        then {
            local-preference 70;
        }
    }
    policy-statement default-only {
        term match-default {
            from {
                route-filter 0.0.0.0/0 exact;
            }
            then accept;
        }
        then reject;
    }
}

```

### Sample Configuration: Strict Primary/Secondary Outbound

The slide provides a sample configuration to enforce a strict primary/secondary relationship. The routing policies set the local preference appropriately and accept only a default route.

## Primary/Secondary Outbound: Configuration Example—Loose

### ■ Loose sample configuration excerpts:

```

protocols {
  bgp {
    group primary-isp {
      [...]
      import [ localpref-80 default-only ];
    }
    [...]
    group secondary-isp {
      [...]
      import [ localpref-70 isp-b-customers default-only ];
    }
    [...]
  }
}

```

This policy accepts the routes for ISP B's customers. Because they are accepted from ISP B, but not the primary ISP, the secondary ISP will be used for these routes.

```

policy-options {
  policy-statement localpref-80 {
    then {
      local-preference 80;
    }
  }
  policy-statement localpref-70 {
    then {
      local-preference 70;
    }
  }
  policy-statement isp-b-customers {
    from community isp-b-customer-routes;
    then accept;
  }
  policy-statement default-only {
    term match-default {
      from {
        route-filter 0.0.0.0/0 exact;
      }
      then accept;
    }
    then reject;
  }
  community isp-b-customer-routes members 65002:8000;
}

```



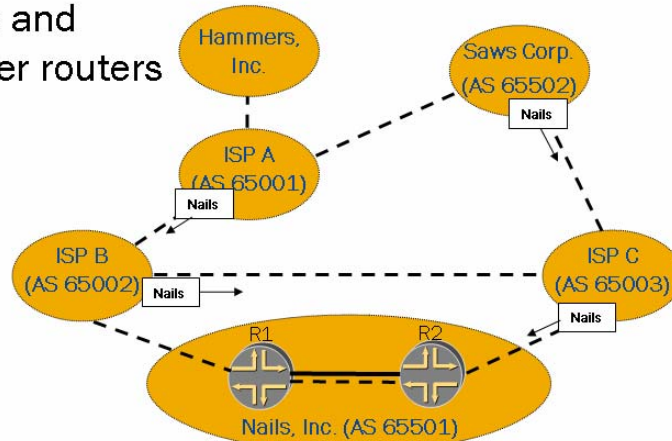
### Sample Configuration: Loose Primary/Secondary Outbound

The configuration example shown on this slide is the same as the strict primary/secondary example, except that we added the *isp-b-customers* policy to the import policy chain on the *secondary-isp* group. The *isp-b-customers* policy matches on a community that our provider adds to the path attributes of all BGP announcements originated by their customers.

Because we allow the more-specific routes to be received from the secondary ISP only, the routers choose the secondary ISP as the best path for those prefixes. Because these routes are more specific than the default route that is received from the primary ISP, the router follows these routes to the secondary ISP when it receives them. If the secondary ISP fails, it follows the default route received from the primary ISP.

## Primary/Secondary Inbound Policy

- Enforce *correct* entrance via *both* communities to set local preference *and* AS path prepending
- Use for single and multiple border routers



### Local Preference and AS Path Prepending

To ensure the correct operation of a primary/secondary inbound routing policy, it must be configured correctly. In particular, you must pay attention to normal operation, failure conditions, and failback (restoration from failure) conditions. To ensure correct primary/secondary inbound routing policy operation, you must often set the local preference within the provider's network through the use of communities and also prepend the AS path of the announcements sent over the secondary path. The case study at the end of the chapter illustrates the reason for this.

### Single Versus Multiple Border Routers

Where a single primary and a single secondary path exist, you configure the primary/secondary routing policy in the same way whether there are one or two border routers. When multiple, equally preferred primary and secondary paths exist, this routing policy takes on a mixture of the primary/secondary and *topology-driven* routing policies. Because each situation is different, we cannot begin to provide design guidance for each and every combination; however, we can describe to you a way to approach the problem. The following slides, which describe the operation of the primary/secondary inbound routing policy, should show you how to think about these design scenarios.

## Primary/Secondary Inbound: Configuration Example

### ■ Sample configuration excerpts:

```

protocols {
  bgp {
    group primary-isp {
      [...]
      export routes-to-ISP;
    }
    [...]
    group secondary-isp {
      [...]
      export [ set-backup routes-to-ISP ];
    }
    [...]
  }
}
policy-options {
  prefix-list announce-to-ISP {
    172.31.128.0/20;
  }
  policy-statement routes-to-ISP {
    from {
      prefix-list announce-to-ISP;
    }
    then accept;
  }
  policy-statement set-backup {
    then {
      community set ISP-B-localpref-70;
      as-path-prepend "65501 65501 65501";
    }
  }
  community ISP-B-localpref-70 members 65002:70;
}

```



### Sample Configuration: Primary/Secondary Inbound

In the configuration on the slide, the announcements to the secondary ISP are prepended to Nails, Inc.'s own AS number three times. Additionally, a community is set that causes ISP B to assign a local preference value lower than the value they assign to the routes they receive from their peers and upstream providers. Note that Juniper Networks routers send communities by default, so no additional configuration is necessary to send communities to BGP peers.

## Load-Shared per-Prefix Routing Policies

- Direct traffic to certain prefixes over certain links to effect load sharing
  - Is a variation on primary/secondary routing policy, but on a per-prefix basis
  - Requires monitoring and adjustment to ensure that desired load sharing is achieved
- Use same configuration for single or multiple border routers
  - On multiple routers, keep configurations synchronized



### Load-Shared per-Prefix Routing Policies

A load-shared per-prefix routing policy is really a primary/secondary routing policy, but you choose a different provider as primary for each prefix (or set of prefixes). Because different providers should be used for traffic to certain prefixes, this should lead to load sharing over multiple providers. Maintaining any sort of traffic parity over the various providers requires fairly consistent monitoring and adjustment during setup, as well as ongoing monitoring and adjustment as traffic patterns change.

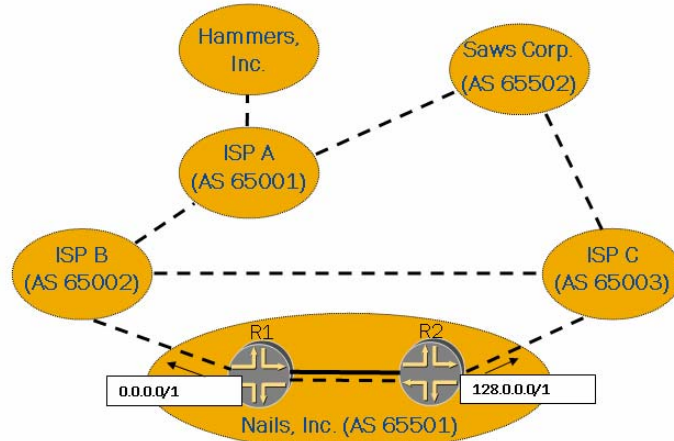
This traffic model is appropriate where cost dictates using multiple available links to the fullest extent possible. This model sacrifices the 1:1 redundancy found in the *strict* primary/secondary model and the performance found in the *topology-driven* model.

### Single Versus Multiple Routers

Once again, the configuration is the same whether you use one or more border routers; however, it is important to keep configurations synchronized between multiple border routers to ensure that each router is correctly marking the routes it receives.

## Load-Shared per-Prefix Outbound Policy

- Set a higher local preference for the specific routes within a large range for each provider



### Load-Shared per-Prefix Outbound Routing Policy

To implement a load-shared per-prefix outbound routing policy, you set a higher local preference for the specific routes within a large range for each provider. For example, this slide shows that Nails, Inc. chose to set a higher local preference on the routes within 0.0.0.0/1 from ISP B and to set a higher local preference on the routes within 128.0.0.0/1 from ISP C. If either ISP failed, the routes from the other ISP would be used, providing reachability. However, without 1:1 redundancy, we cannot guarantee that there would not be performance problems during a failure.

## Load-Shared per-Prefix Outbound: Configuration Example

### ■ Sample configuration excerpts:

```

protocols {
  bgp {
    group isp-b {
      [...]
      import isp-b-import;
    }
    [...]
    group isp-c {
      [...]
      import isp-c-import;
    }
    [...]
  }
}

policy-options {
  policy-statement isp-b-import {
    term primary {
      from {
        route-filter 0.0.0.0/1 orlonger;
      }
      then {
        local-preference 80;
        accept;
      }
    }
  }
  policy-statement isp-c-import {
    term primary {
      from {
        route-filter 128.0.0.0/1 orlonger;
      }
      then {
        local-preference 80;
        accept;
      }
    }
  }
}

```



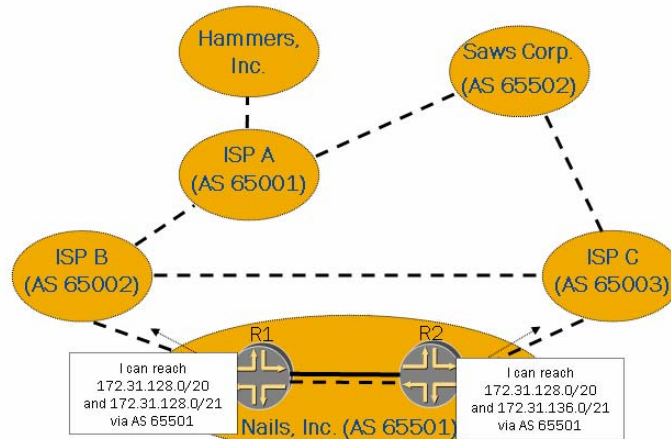
### Sample Configuration: Load-Shared Per-Prefix Outbound

We can use the sample configuration on the slide to implement the scenario described on the previous page.



## Load-Shared per-Prefix Inbound Policy

- Announce entire aggregate to all providers, but also announce more specific prefixes to each provider



### Load-Shared per-Prefix Inbound Routing Policy

To implement a load-shared per-prefix inbound routing policy, you take advantage of longest-match routing. You announce your aggregate prefix to all your providers, but you also announce a more specific prefix to each provider. As long as all providers are functioning, all routers on the Internet follow the more specific routes, resulting in traffic for each prefix using a different provider. However, if a provider malfunctions and does not successfully advertise one of the more specific routes, the routers on the Internet will follow the aggregate prefixes to the closest ISP announcing it. Again, without 1:1 redundancy, there is no way to guarantee good performance during failure conditions.

Note that your more-specific prefixes must be no more specific than /24 prefixes for those announcements to propagate through the Internet. Also note that some ISPs might enforce stricter filtering guidelines; however, if you are assigned portable address space by a regional registry, your aggregate prefixes should be accepted by all ISP filters on the Internet, which is yet another reason to announce the aggregate prefix along with the specific prefixes.



## Load-Shared per-Prefix Inbound: Configuration Example

### ■ Sample configuration excerpts:

```

protocols {
  bgp {
    group isp-b {
      [...]
      export [ isp-b-export accept-aggregates reject-all ];
    }
    [...]
    group isp-b {
      [...]
      export [ isp-b-export accept-aggregates reject-all ];
    }
    [...]
  }
}

policy-options {
  prefix-list aggregates {
    172.31.128.0/20;
  }
  prefix-list isp-b-specifics {
    172.31.128.0/21;
  }
  prefix-list isp-c-specifics {
    172.31.136.0/21;
  }
  policy-statement accept-aggregates {
    from {
      prefix-list aggregates;
    }
    then accept;
  }
  policy-statement isp-b-export {
    from {
      prefix-list isp-b-specifics;
    }
    then accept;
  }
  policy-statement isp-c-export {
    from {
      prefix-list isp-c-specifics;
    }
    then accept;
  }
  policy-statement reject-all {
    then reject;
  }
}

```



### Sample Configuration: Load-Shared per-Prefix Inbound

We can use the sample configuration on the slide to implement the scenario described on the previous page.

## **Agenda: Enterprise Routing Policies**

---

- Enterprise BGP Deployment
- Case Study: Primary/Secondary Routing Policy



### **Case Study: Primary/Secondary Routing Policy**

The slide highlights the topic we discuss next.

## Primary/Secondary Case Study Overview

- Follow the engineers at Florists, Inc. as they try to implement a strict primary/secondary inbound routing policy
- Discover why either AS path prepending or communities to set local preference alone does not provide a strict primary/secondary inbound routing policy
- Think through the effects of routing policies
  - Route servers, looking glasses, etc.



### Case Study Overview

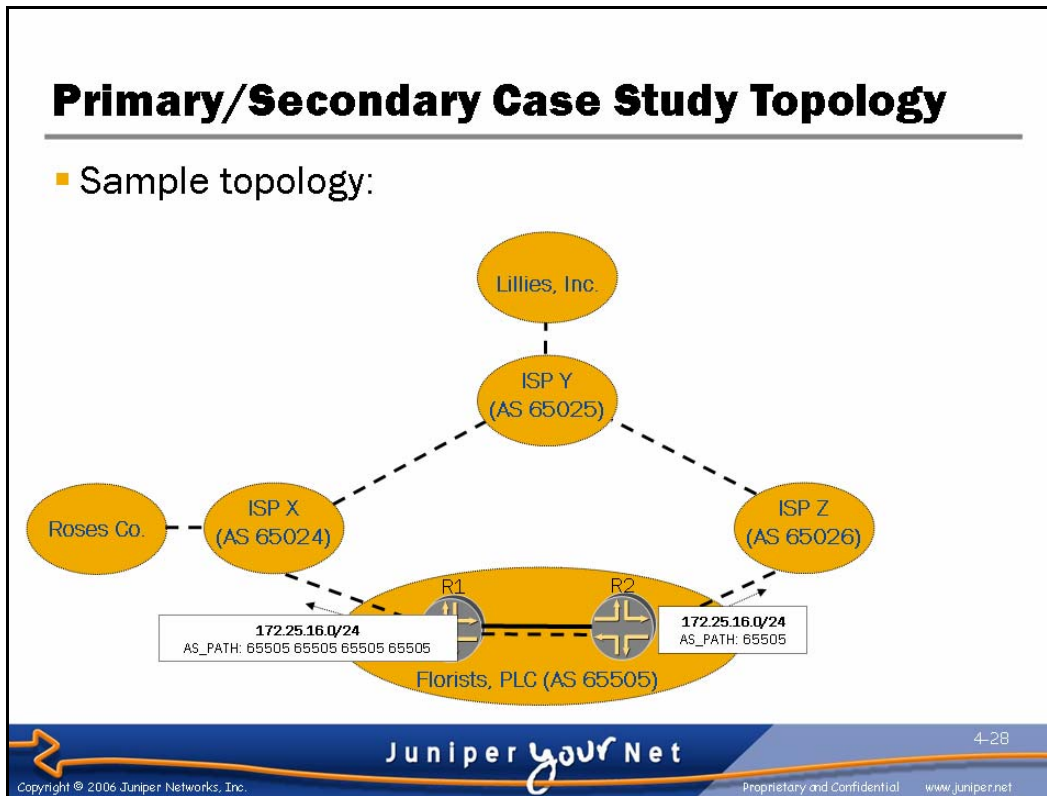
This case study follows the engineers at Florists, PLC as they try to implement a strict primary/secondary inbound routing policy. First, they try AS path prepending alone. Then, they try using communities to modify their provider's local preference. Finally, they try using both together. This is not really a true case study of a single organization, but rather it is the combination of several organizations' experience and testing.

### Implementing a Strict Primary/Secondary Inbound Routing Policy

The following case study illustrates why you must use AS path prepending and modification of the provider's local preference via communities together to ensure the reliable operation of a primary/secondary inbound routing policy.

### Evaluating Routing Policies

Prior to implementing any routing policies—and, especially, inbound routing policies—you must thoroughly consider all the ramifications of the policy. You must consider normal operations, various failure conditions, and the recovery from those failure conditions. One of the methods you can use to evaluate the possible effects of routing policies is to use route servers or looking glasses to investigate the way your providers are interconnected. These same tools will let you verify the proper operation of your inbound routing policy. You can find a listing of route servers and looking glasses at <http://www.traceroute.org/>.



### Sample Topology

The case study is based on the topology shown on the slide. In this topology, we find an enterprise (Florists, PLC), that has two connections: one to ISP X and one to ISP Z. It wants inbound traffic to use the path through ISP Z whenever possible. Inbound traffic should only use the connection to ISP X when the path through ISP Z is unavailable.

We also show two companies that are partners of Florists, PLC: Roses Co. and Lillies, Inc. Florists, PLC regularly exchanges data with these two entities, and we can use them as a reference when checking the inbound traffic flow.

## Primary/Secondary Case Study: Routing Policy

- ISPs X, Y, and Z all have a similar routing policy
  - Default LOCAL\_PREF for customer routes: 100
  - Default LOCAL\_PREF for peer/upstream routes: 80
  - Accept certain communities from customers:
    - (AS):110 – Set LOCAL\_PREF=110
    - (AS):100 – Set LOCAL\_PREF=100
    - (AS):90 – Set LOCAL\_PREF=90
    - (AS):80 – Set LOCAL\_PREF=80
    - (AS):70 – Set LOCAL\_PREF=70
  - Example: 65026:110 causes ISP Z (AS 65026) to set a LOCAL\_PREF of 110 for that particular route.



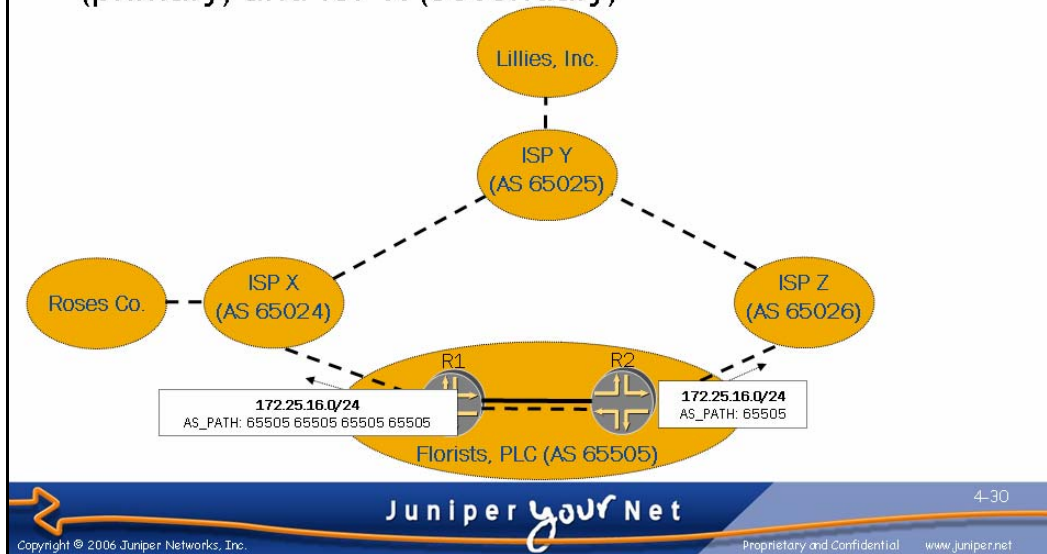
### ISP X's, Y's and Z's Routing Policies

When trying to influence inbound traffic, you should understand your providers' routing policies as thoroughly as possible and also understand how your providers are attached to one another. This knowledge helps you to understand the way that various modifications to the path attributes affect inbound traffic.

In this case study, ISPs X, Y, and Z all have a similar routing policy. All ISPs assign a local preference value of 100 for routes received from customers and a value of 80 for routes received from peers or upstream ISPs. All the ISPs accept certain communities from their customers only and modify the local-preference value according to the chart on the slide.

## AS Path Prepending Example (1 of 4)

- Florists, PLC advertises its prefix to both ISP Z (primary) and ISP X (secondary)

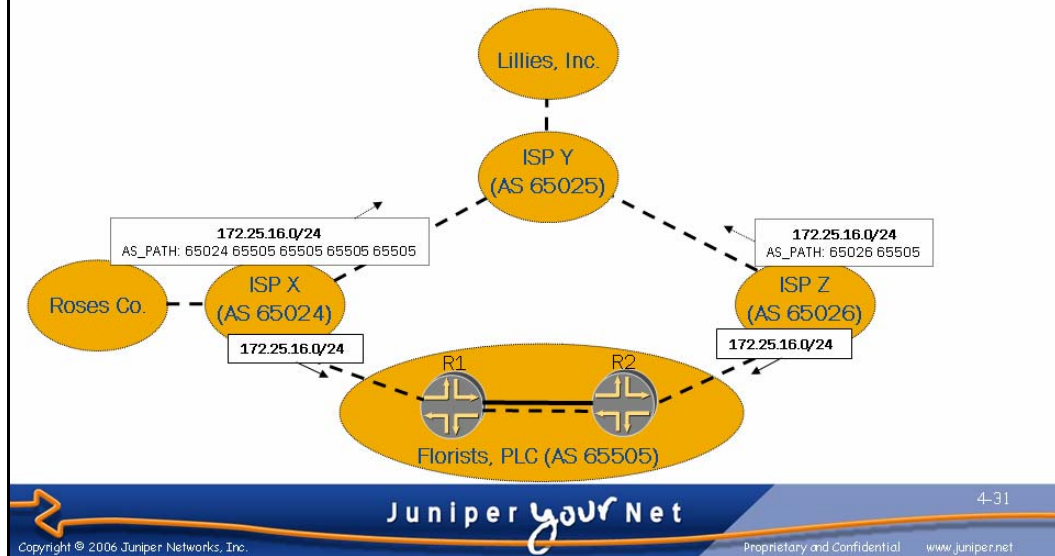


### AS Path Prepending Example—Session Start

Florists, PLC begins by trying to use AS path prepending alone to provide a primary/secondary inbound traffic flow. In this example, we see the results of this method. We see here that Florists, PLC has begun to advertise a prefix to its providers, prepending its own AS three times to the AS path in the announcement sent to ISP X.

## AS Path Prepending Example (2 of 4)

- ISPs X and Z both install routes for 172.25.16.0/24 and forward their routes to ISP Y

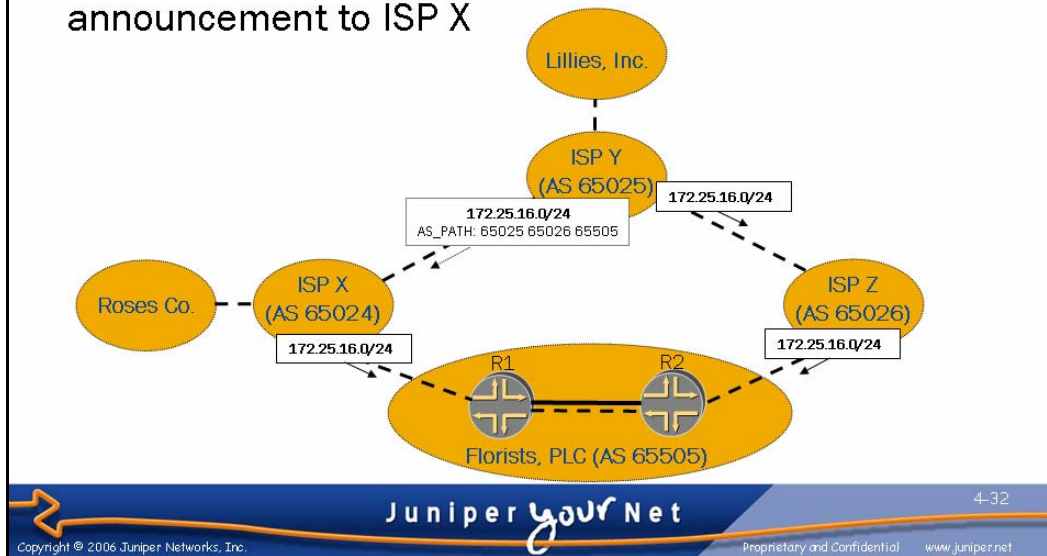


### AS Path Prepending Example—ISPs X and Z Forward the Announcement

On this slide, we see that ISPs X and Z both begin to use the route they learned from Florists, PLC because it is the only announcement they have heard for this prefix thus far. When they choose to use this path, they also send an update to ISP Y to tell it about this path.

## AS Path Prepending Example (3 of 4)

- ISP Y chooses the path through ISP Z (shorter AS path), installs the route, and sends the announcement to ISP X



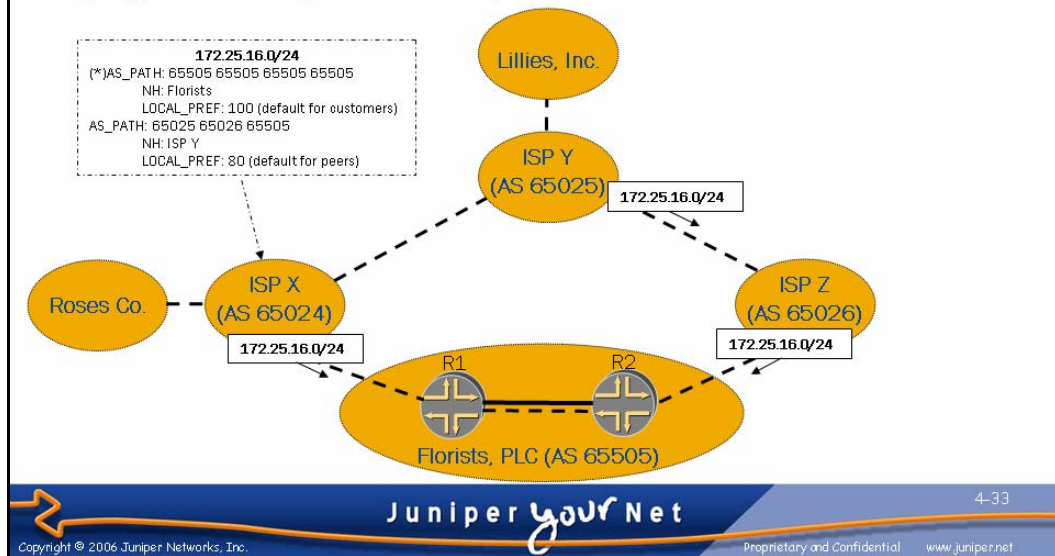
### AS Path Prepending Example—ISP Y Chooses a Path

ISP Y receives both announcements, but it chooses the one with the shorter AS path, which is the announcement from ISP Z. So, it installs a route for the Florists, PLC prefix pointing to ISP Z. Because it chose to use the path through ISP Z, rather than the one through ISP X, ISP Y sends an announcement to ISP X about the path to Florists, PLC through ISP Z. ISP Y does *not* send an announcement to ISP Z about the path through ISP X because ISP Y did not choose the path through ISP X as the best path.



## AS Path Prepending Example (4 of 4)

- ISP X continues using the path directly to Florists, PLC (higher local preference)



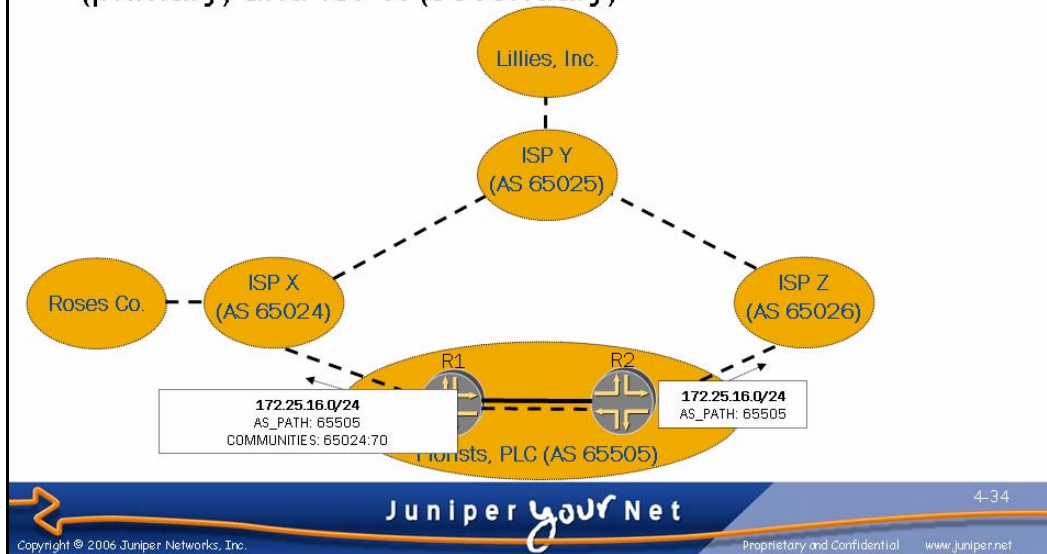
### AS Path Prepending Example—ISP X Chooses a Path

This slide shows the two entries in ISP X's routing table for the Florists, PLC prefix. Even though the path through ISP Y and ISP Z has a shorter AS path, ISP X continues to use the path directly to Florists, PLC because it has a higher local-preference value.

Thus, we see that merely performing AS path prepending is insufficient to achieve the desired results, because ISP X will prefer sending traffic over the secondary link.

## LOCAL\_PREF Example (1 of 20)

- Florists, PLC advertises its prefix to both ISP Z (primary) and ISP X (secondary)



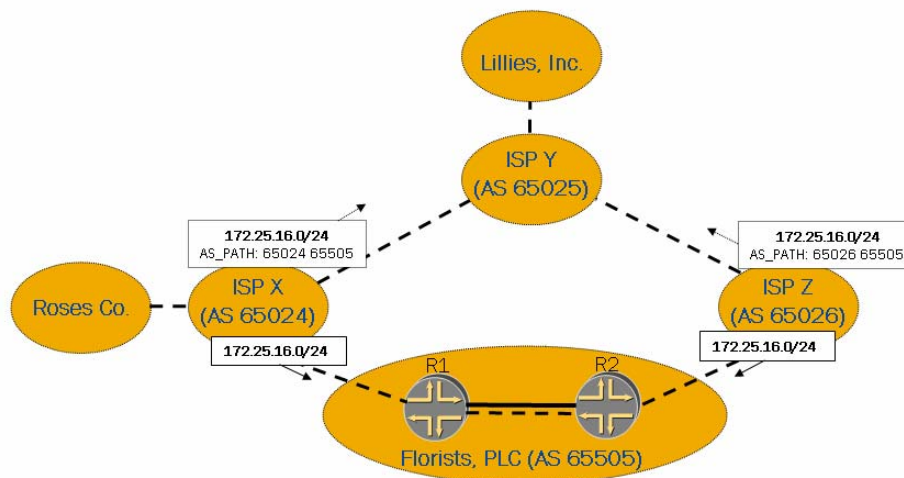
### Local Preference Example—Session Start

We saw the results of using AS path prepending alone to create a primary/secondary traffic flow. We saw that ISP X did not follow the primary path because the local preference value of the secondary path (directly to Florists, PLC) was higher than the local-preference value of the primary path (via ISP Y and ISP Z). When Florists, PLC observed this result, it seemed that the obvious solution to this problem was to use communities to have the provider change the local preference of the secondary announcement *instead* of using AS path prepending.

In this example, we will see the results of only using communities to change the provider's local preference to try to establish a primary/secondary inbound traffic flow. We see here that Florists, PLC has begun to advertise a prefix to its providers. It has added the community 65024:70 to the path attributes of the update it sends to ISP X. As the chart on slide 4-29 shows, this addition causes ISP X to change the local preference of this announcement to 70, which is lower than the local-preference value assigned to routes received from peer or upstream providers (such as ISP Y).

## LOCAL\_PREF Example (2 of 20)

- ISPs X and Z both install routes for 172.25.16.0/24 and forward their routes to ISP Y (ISP X deletes the community before forwarding the announcement)

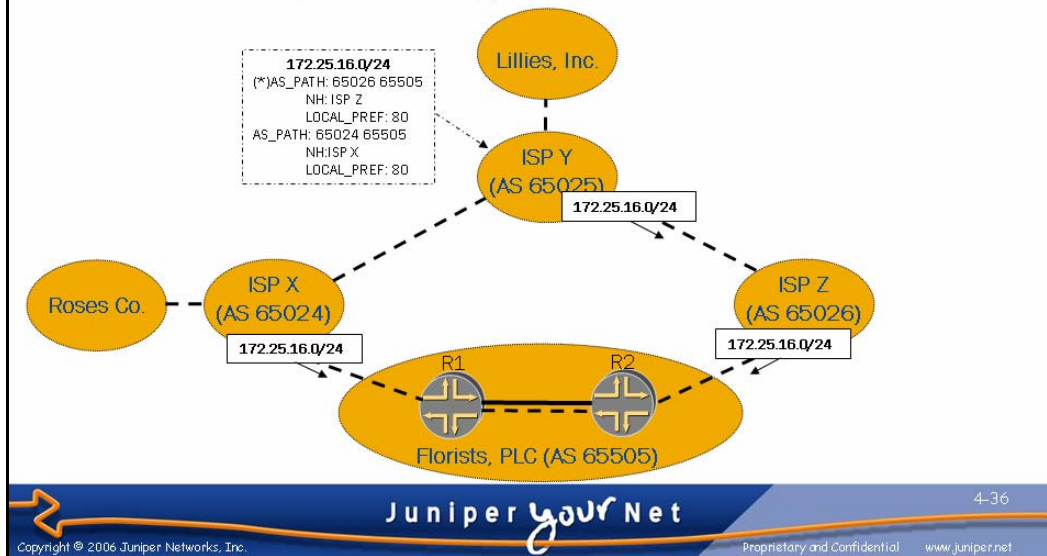


### Local Preference Example—ISPs X and Z Forward the Announcements

On this slide, we see that ISPs X and Z both begin to use the route they learned from Florists, PLC because it is the only announcement they have heard for this prefix thus far. When they choose to use this path, they also send an update to ISP Y to tell it about this path.

## LOCAL\_PREF Example (3 of 20)

- ISP Y chooses the path received first... let's assume that it is the path through ISP Z



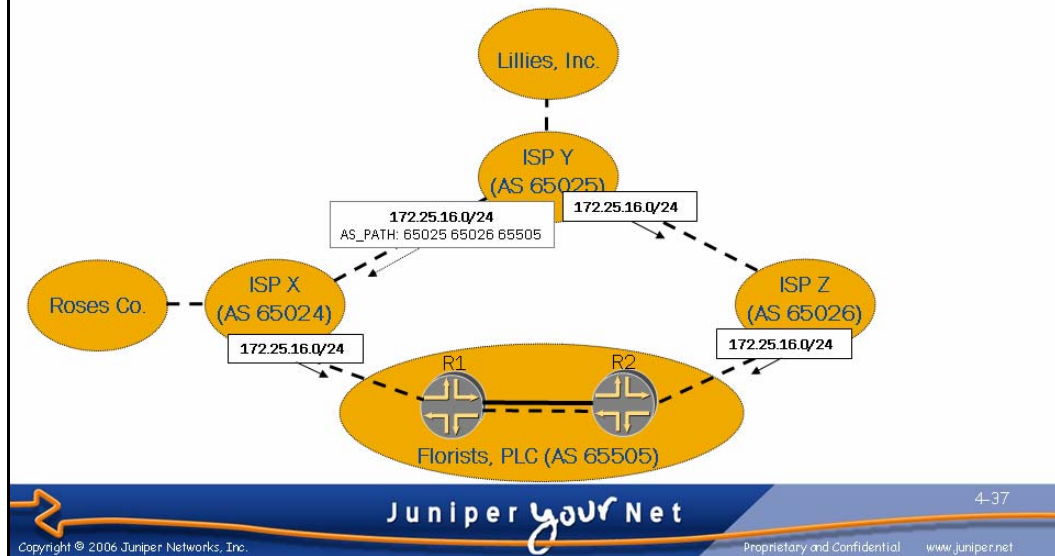
### Local Preference Example—ISP Y Chooses a Path

In this example, ISP Y receives announcements from both ISP X and ISP Z for the same prefix. For the sake of simplicity, we assume that ISP X and ISP Z are attached to the same router in ISP Z's network. The slide shows the two entries in the routing table for this path, in the order received. As the router follows the route selection algorithm described on slide 3-12, note that these two routes are tied through Step 9. Because these are both EBGp received paths, it prefers to keep the active route. The net effect of this is that the announcement received first is used.

To understand why the router always uses the route received first, consider the way the router processes these updates. ISP Y begins with no routes for the Florists, PLC prefix. It receives the update from ISP Z, determines this is the best path to the prefix (because it has no other paths), and makes this the active route in the routing table. Just milliseconds later, it receives the announcement from ISP X for the Florists, PLC prefix. It installs it in the routing table and then runs the selection algorithm to determine the best path. When it gets to Step 9, it prefers keeping the active path, which is the path through ISP Z.

## LOCAL\_PREF Example (4 of 20)

- ISP Y sends the announcement to ISP X

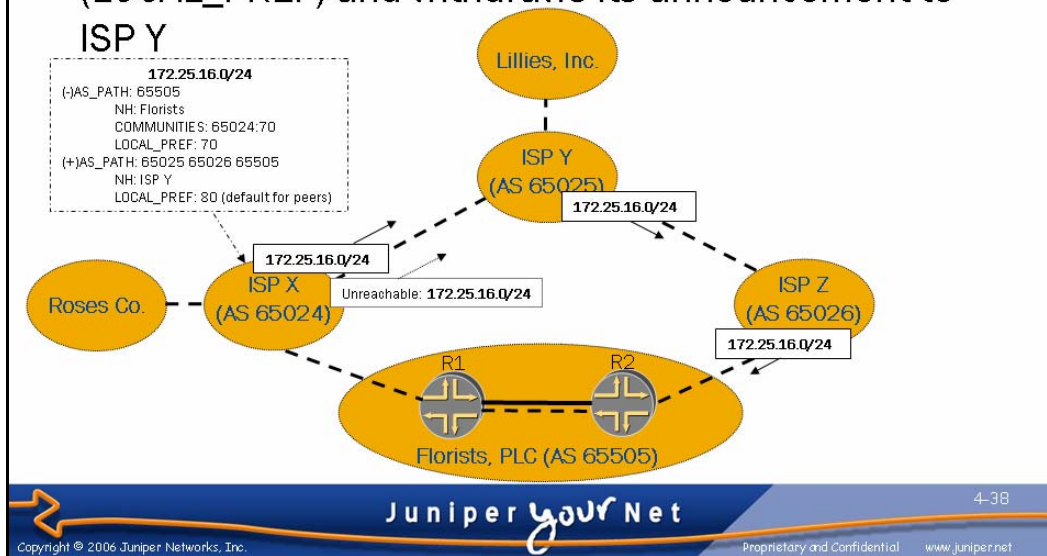


### Local Preference Example—ISP Y Updates ISP X

Because ISP Y chose the path through ISP Z as the best path to reach Florists, PLC, it sends an update to ISP X to announce the path to Florists, PLC through ISP Y and ISP Z.

## LOCAL\_PREF Example (5 of 20)

- ISP X begins to use the announcement through ISP Y (LOCAL\_PREF) and withdraws its announcement to ISP Y



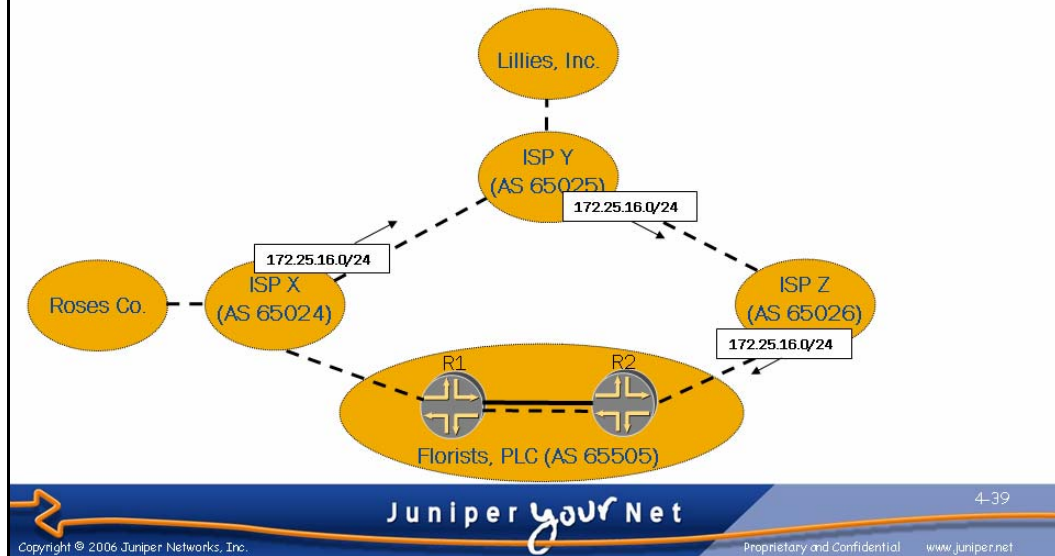
### Local Preference Example—ISP X Chooses a Path

Here, you see the routing table entries in ISP X for the Florists, PLC prefix. Notice from the plus sign (+) next to the path through ISP Y and ISP Z that ISP X chose this as the new active route for this prefix.

When it decides to use this path, it sends an update to ISP Y to explicitly withdraw the previous path it advertised to ISP Y. Because ISP Y was not using this path, it simply deletes the inactive route from the routing table and takes no further action.

## LOCAL\_PREF Example (6 of 20)

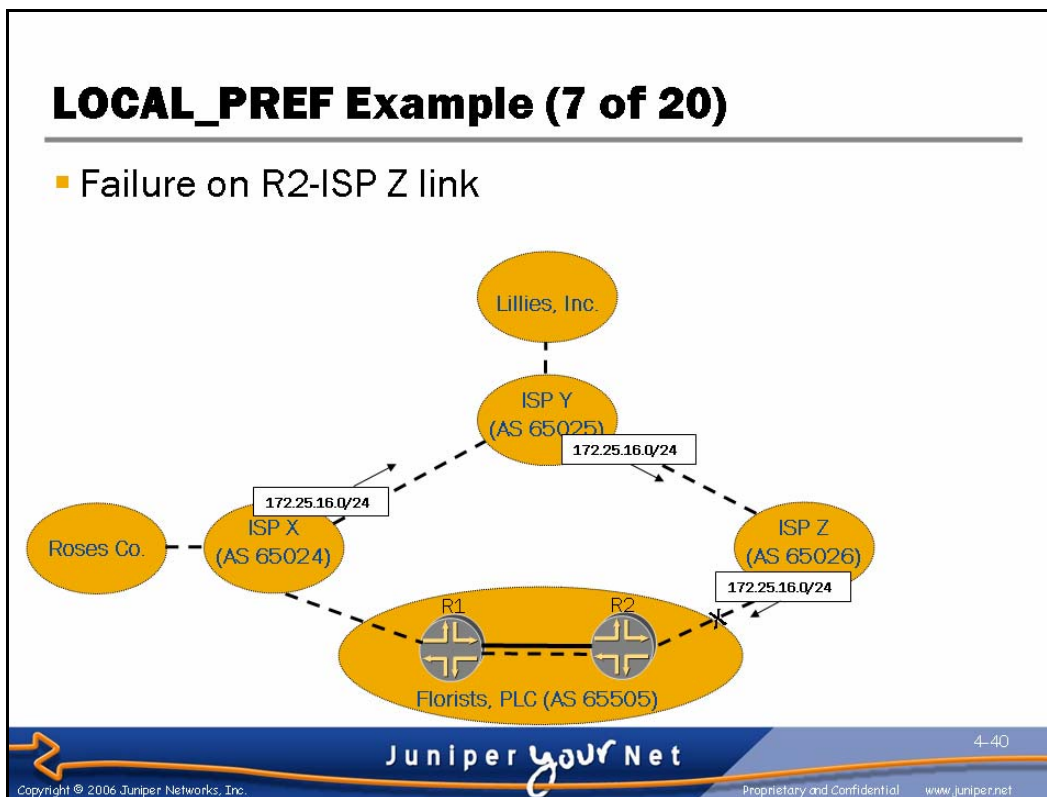
- Traffic now follows desired primary path



### Local Preference Example—Normal Operation

As you can see on the slide, this scenario has resulted in a *normal* operating mode where all traffic follows the primary path and no traffic follows the secondary path.





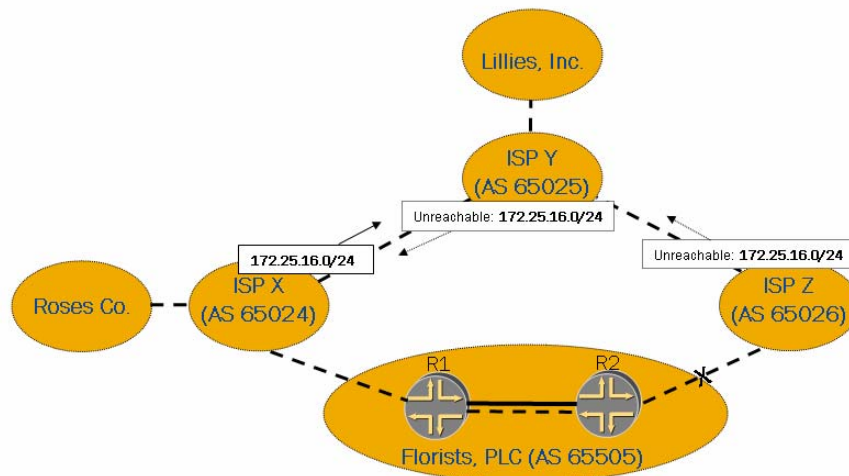
### Local Preference Example—Failure of the Primary Path

We now consider the result of a failure of the primary path. In this case, we assume that some sort of failure happened on the physical link between ISP Z and R2.



## LOCAL\_PREF Example (8 of 20)

- ISP Z withdraws the announcement from ISP Y
  - ISP Y in turn withdraws the announcement from ISP X

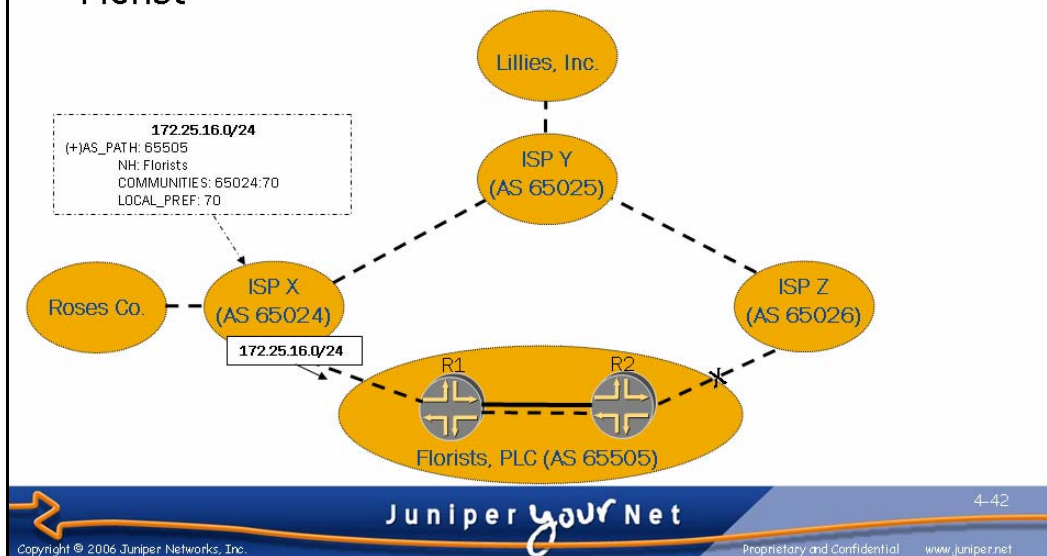


### Local Preference Example—ISP Z Withdraws the Route

When the link between ISP Z and R2 fails and the BGP session is dropped, ISP Z responds by deleting the routes that were received over this BGP session. Because the route received over this session was its active route to Florists, PLC, it also sends an update to ISP Y to explicitly withdraw the path to this prefix. When it receives the withdrawal, ISP Y responds by deleting this route from its routing table and sending an update to ISP X to explicitly withdraw the path to this prefix. This reaction leaves both ISP Y and ISP Z without a path to reach the Florists, PLC prefix.

## LOCAL\_PREF Example (9 of 20)

- ISP X begins using the announcement directly from Florist

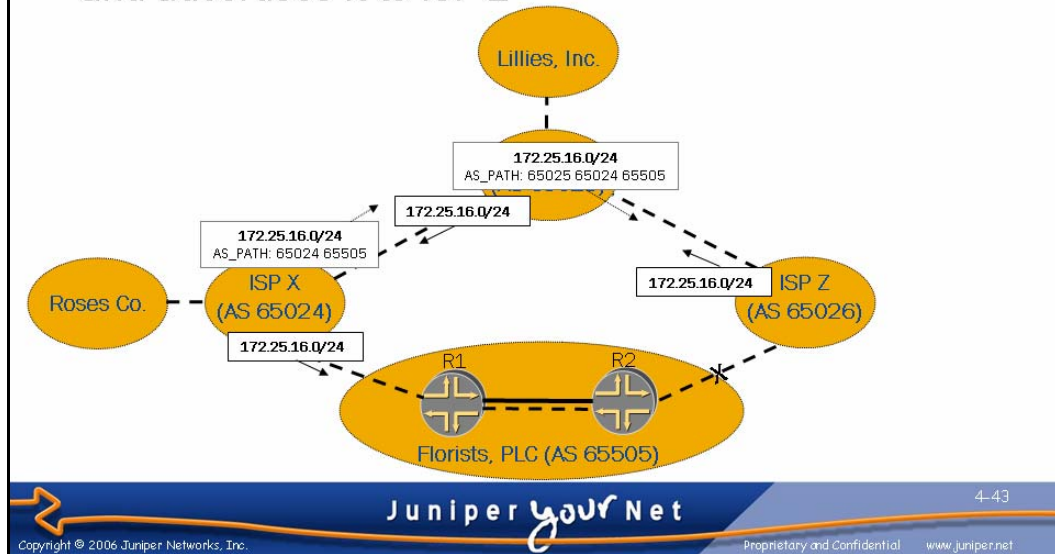


### Local Preference Example—ISP X Chooses the Secondary Path

Once ISP X receives the update from ISP Y withdrawing the path to Florists, PLC through ISP Y and ISP Z, it removes that route from its routing table. Because it still has a route to this prefix that it had received directly from Florists, PLC, it begins to use this route as the active route.

## LOCAL\_PREF Example (10 of 20)

- ISP X advertises the best path to ISP Y, who uses it and advertises it to ISP Z

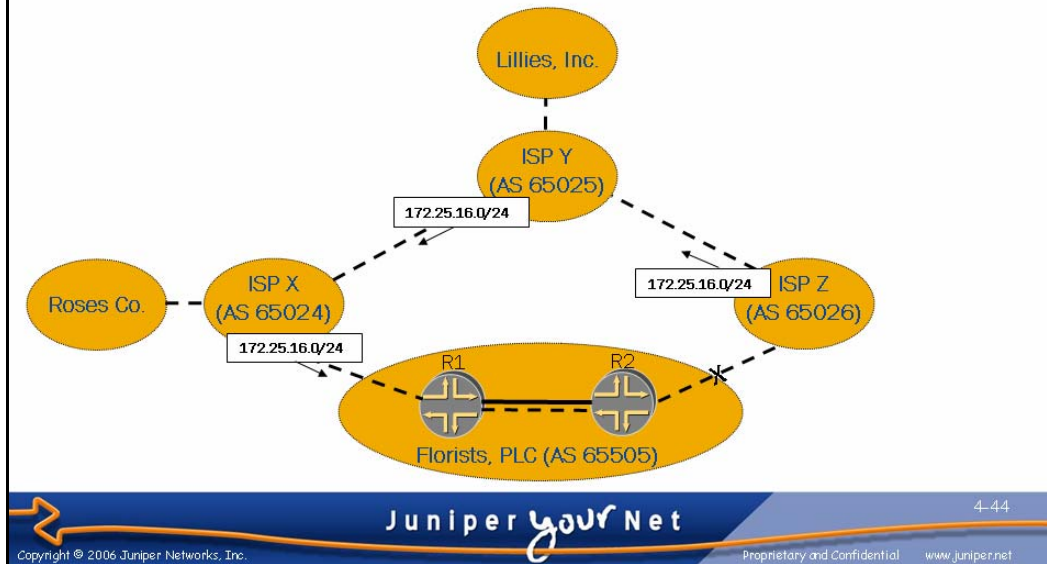


### Local Preference Example—ISP X Sends Updates

Once it installs the route directly to Florists, PLC as the active route, ISP X announces the route to ISP Y. Because ISP Y has no other routes to Florists, PLC, it installs the route as the active route and sends an update to ISP Z to announce the path. Because ISP Z also has no other routes to Florists, PLC, it also installs the route as the active route.

## LOCAL\_PREF Example (11 of 20)

- Traffic now follows the desired secondary path



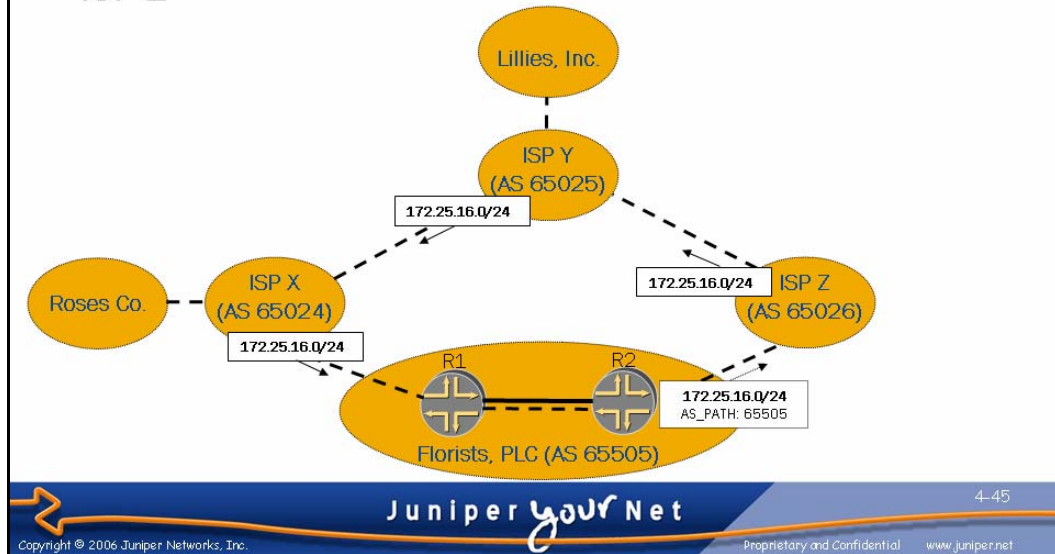
### Local Preference Example—Failure Operation

As you can see on the slide, this scenario resulted in a failure mode where all networks can continue to reach Florists, PLC and traffic is following the secondary path.

However, when considering primary/secondary routing policies, we must consider not only the normal and failure modes, but we must also consider the failback from a failure mode to normal operation. We consider this aspect of the process in the remainder of this example.

## LOCAL\_PREF Example (12 of 20)

- R2-ISP Z link is restored, R2 sends announcement to ISP Z

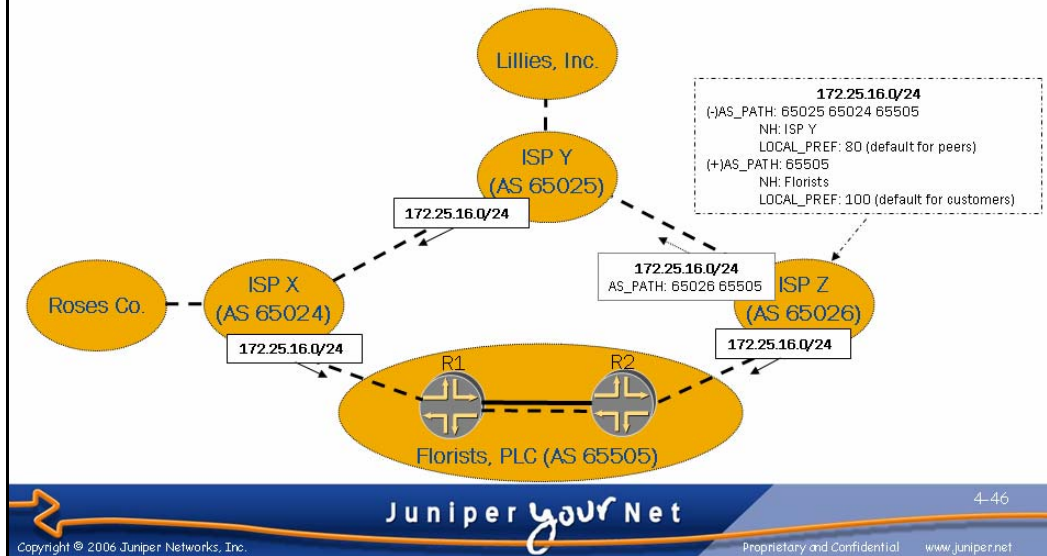


### Local Preference Example—Restoration

Once the link between R2 and ISP Z is restored, the BGP session is reestablished, and Florists, PLC again announces its prefix to ISP Z.

## LOCAL\_PREF Example (13 of 20)

- ISP Z uses announcement from Florists, PLC, and sends announcement to ISP Y

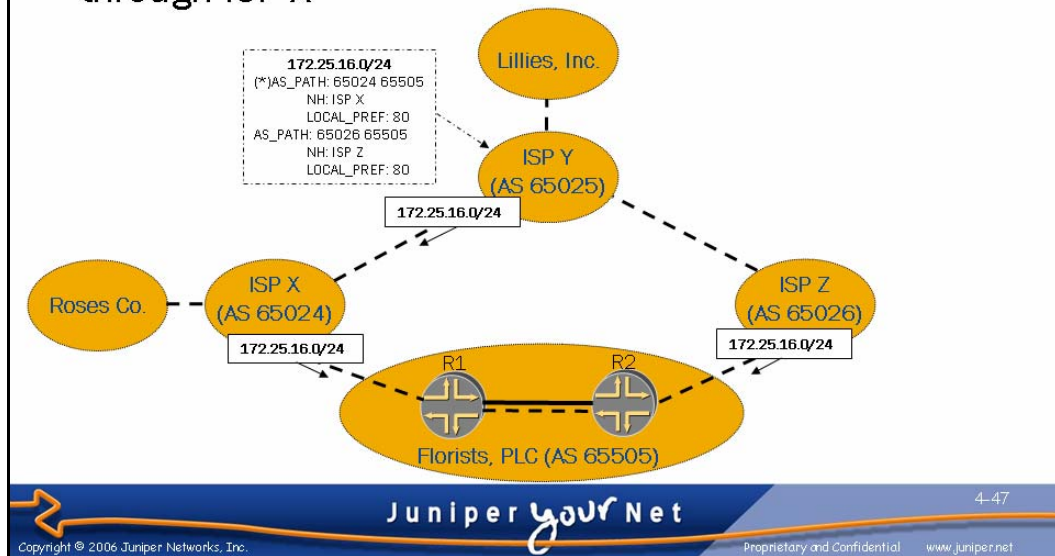


### Local Preference Example—ISP Z Chooses a Path

When ISP Z receives the announcement from Florists, PLC, it installs it in its routing table. Because this announcement has a higher local-preference value than the announcement from ISP Y, ISP Z chooses this route to be the active route. It then sends an update to ISP Y to announce this path.

## LOCAL\_PREF Example (14 of 20)

- ISP Y chooses the path received first... the path through ISP X



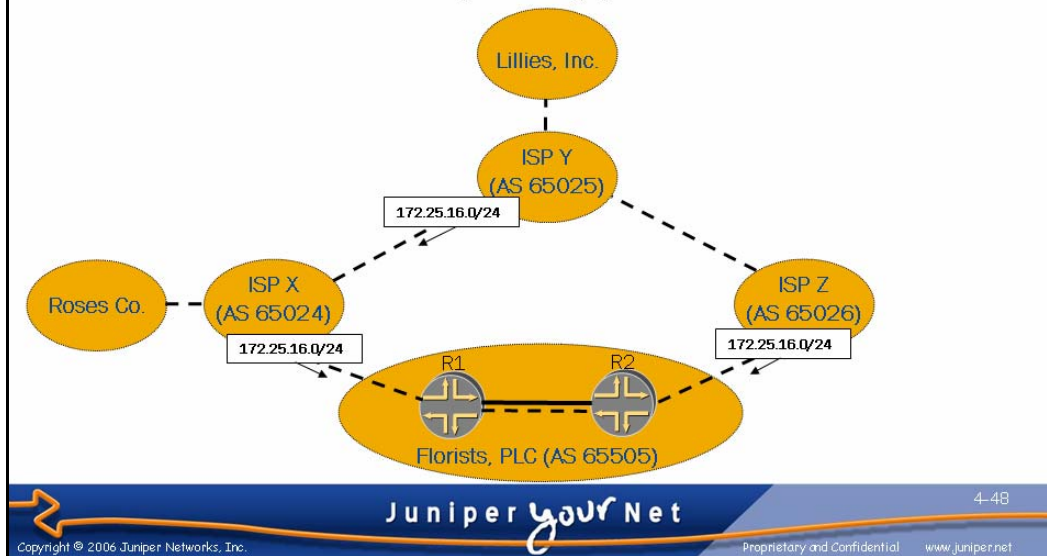
### Local Preference Example—ISP Y Chooses a Path

When ISP Y receives the announcement about the path through ISP Z, it installs it in its routing table. However, because it is tied with the path through ISP X, ISP Y prefers to continue using the announcement from ISP X. (Even if we assumed ISP X and ISP Z were attached to different routers within ISP Y, the router to which ISP X is attached would continue to prefer that path because it would be an EBGP-received path on that router. Therefore, at least part of ISP Y would continue to follow the path to Florists, PLC through ISP X.)



## LOCAL\_PREF Example (15 of 20)

- The failover to the secondary path worked, but not all traffic returned to the primary path after the failure



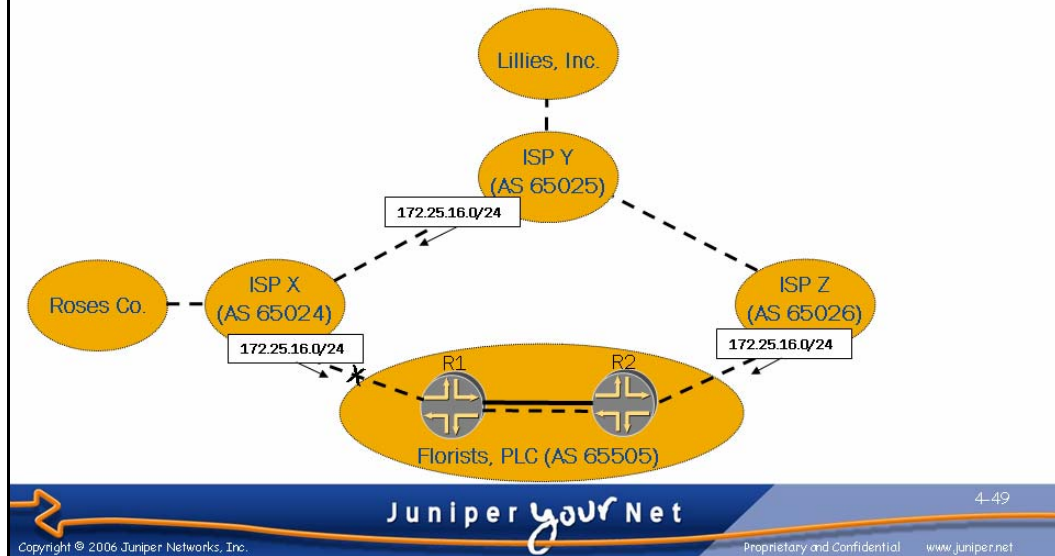
### Local Preference Example—Failure in Failback!

We see on the slide that the traffic has not fully returned to the primary path following the restoration of the primary path. Because ISP Y continues to prefer the path through ISP X, it never sends an update to ISP X to tell it about the path through ISP Z. Therefore, ISP X continues to use the secondary route it received. (As noted on the last page, even if ISP X and ISP Z attached to different routers within ISP Y, at least the router to which ISP X attaches would continue to prefer the route received from ISP X, so it would never send an update to ISP X to tell it about the primary path (through ISP Z). Thus, in that topology, even though part of ISP Y might prefer the primary path (through ISP Z), at least part of ISP Y and all of ISP X would prefer to follow the secondary path.)



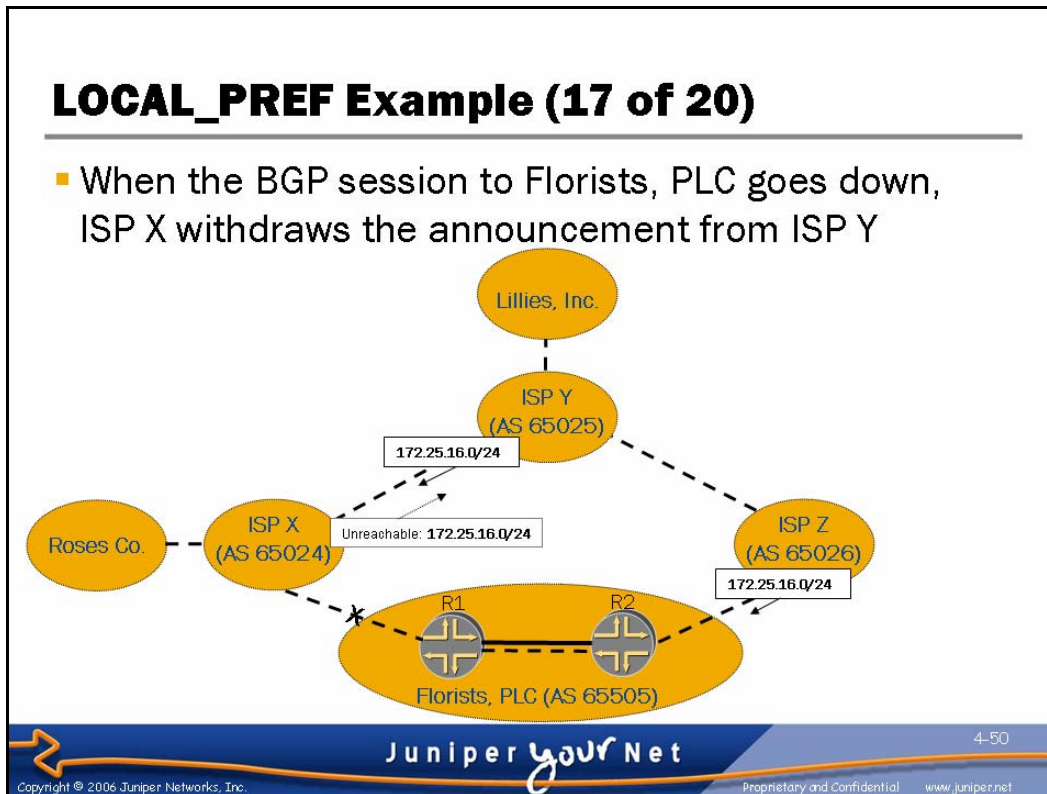
## LOCAL\_PREF Example (16 of 20)

- To *fix* this situation, the Florists, PLC network engineer clears the BGP session to ISP X



### Local Preference Example—Florists, PLC Resets the Secondary Session

In an effort to restore traffic to the primary path, a network engineer at Florists, PLC decides to reset the BGP session to ISP X.

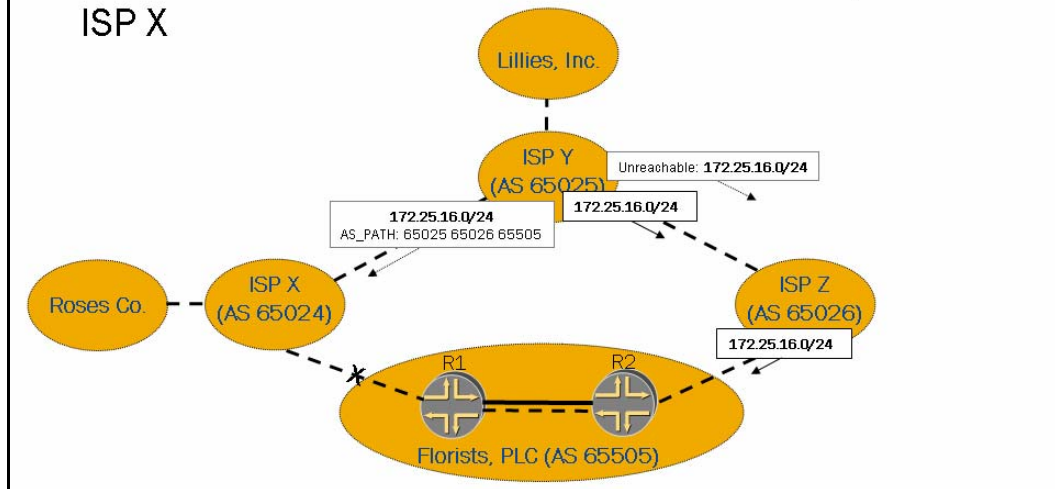


### Local Preference Example—ISP X Withdraws the Secondary Path

When the BGP session to Florists, PLC goes down, ISP X deletes the routes it received over this session from its routing table, including the direct path to the Florists, PLC prefix. Because this was the active (and, in fact, only) route to this prefix in its routing table, it sends an update to ISP Y to explicitly withdraw this path.

## LOCAL\_PREF Example (18 of 20)

- ISP Y withdraws the announcement (via ISP X) from ISP Z, begins using the announcement it still has from ISP Z, and sends the announcement for that path to ISP X

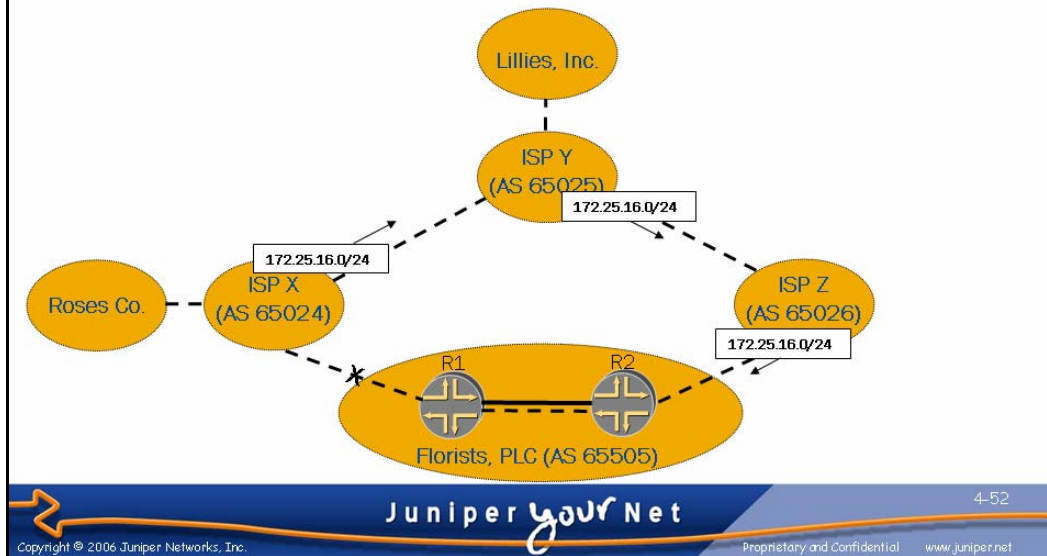


### Local Preference Example—ISP Y Chooses a Path

When ISP Y receives the update from ISP X withdrawing the path to Florists, PLC, it removes this route from the routing table. It then chooses the path through ISP Z as the new path to this prefix. It sends an update to ISP Z to explicitly withdraw the path to Florists, PLC through ISP X and sends an update to ISP X to announce the path to Florists, PLC through ISP Z.

## LOCAL\_PREF Example (19 of 20)

- Traffic is now again following the desired primary path



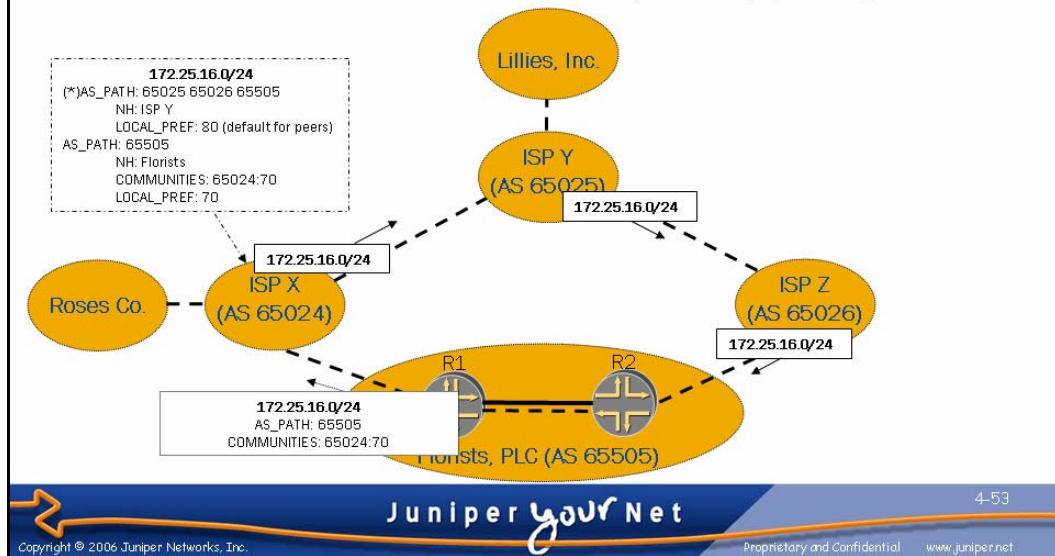
### Local Preference Example—ISP X and ISP Z Choose Paths

Because ISP Z was not using the path to Florists, PLC through ISP Y and ISP X, when it receives the update from ISP Y withdrawing this path, it simply removes the route from its routing table and continues using the path directly to Florists, PLC.

When ISP X receives the announcement from ISP Y about the path to Florists, PLC through ISP Y and ISP Z, it installs the route as the active route (because it has no other routes to this prefix).

## LOCAL\_PREF Example (20 of 20)

- When the R1-ISP X session is reestablished, ISP X continues to use the ISP Y path (LOCAL\_PREF)



### Local Preference Example—Normal Operation Restored

When the BGP session between R1 and ISP X is reestablished, R1 advertises the Florists, PLC prefix to ISP X with a community that causes ISP X to set the local-preference value to 70 for this path. When ISP X receives the announcement, it installs the route in its routing table, but it continues to use the path through ISP Y and ISP Z due to the higher local-preference value for that path.

At this point, normal operation is restored. However, it is not really acceptable (or necessary) for a network engineer to need to manually reset the backup path to restore normal traffic flow following a failure and restoration.

## Primary/Secondary Failback Solution

### ■ Solution:

- AS\_PATH prepending alone: Because of default local preference values, the secondary ISP still prefers the secondary path in normal operation
- LOCAL\_PREF modification alone: After failures, failback might not occur correctly because upstream ISPs might still prefer the secondary path
- AS\_PATH prepending + LOCAL\_PREF modification: In normal operation, the secondary ISP prefers the primary path (LOCAL\_PREF)
  - After failures, upstream ISPs failback to the primary path (AS\_PATH prepending)



### AS Path Prepending Only

When AS path prepending alone is used to implement a primary/secondary traffic flow with different primary and secondary ISPs, it is often the case that the secondary provider will assign a higher local preference to the secondary announcements (because they are received directly from a customer) and, therefore, will still prefer the secondary announcements.

### Local Preference Modification Only

When you try to implement a primary/secondary inbound traffic flow solely by causing one of your providers to modify the local preference of your announcements, it is often the case that an intermediate provider will continue to prefer the secondary path after a failure and restoration. (However, this behavior is dependent on the exact topology of the ISP interconnections, the relationship of the ISPs, and the particular routing policies of the ISPs.)

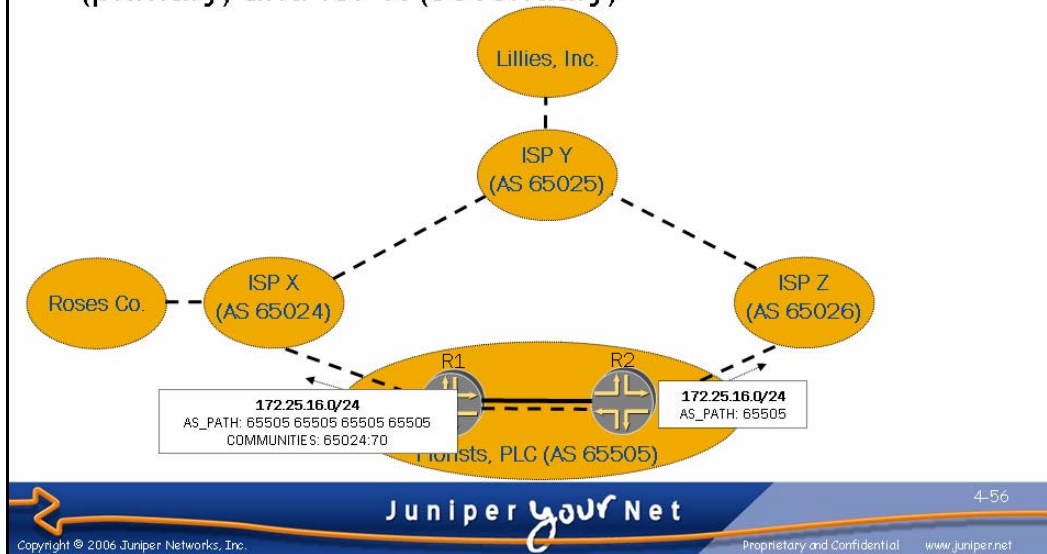
*Continued on next page.*

## AS Path Prepending and Local Preference Modification Together

When using both AS path prepending and causing a provider to modify the local-preference value of announcements to implement a primary/secondary traffic flow with different primary and secondary ISPs, both the failure and restoration should occur correctly. In normal operation, the lower local preference on the announcement via the secondary path causes the secondary ISP to prefer the primary path. After failures, the AS path prepending causes intermediate ISPs to prefer the primary path.

## Prepending + LOCAL\_PREF Example (1 of 16)

- Florists, PLC advertises its prefix to both ISP Z (primary) and ISP X (secondary)



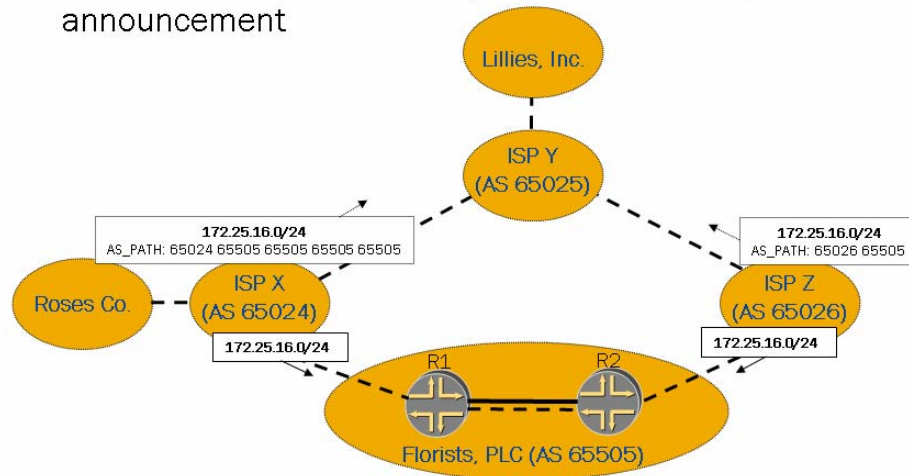
### Prepending and Local Preference Example—Session Start

We now look at the results of both prepending the AS to the AS path and also using communities to change the provider's local preference to try to establish a primary/secondary inbound traffic flow. We see here that Florists, PLC has begun to advertise a prefix to its providers. It prepended its AS three times and also added the community 65024:70 to the path attributes of the update it sends to ISP X. As the chart on slide 4-29 shows, this community causes ISP X to change the local-preference value of this announcement to 70, which is lower than the local-preference value assigned to routes received from peer or upstream providers (such as ISP Y).



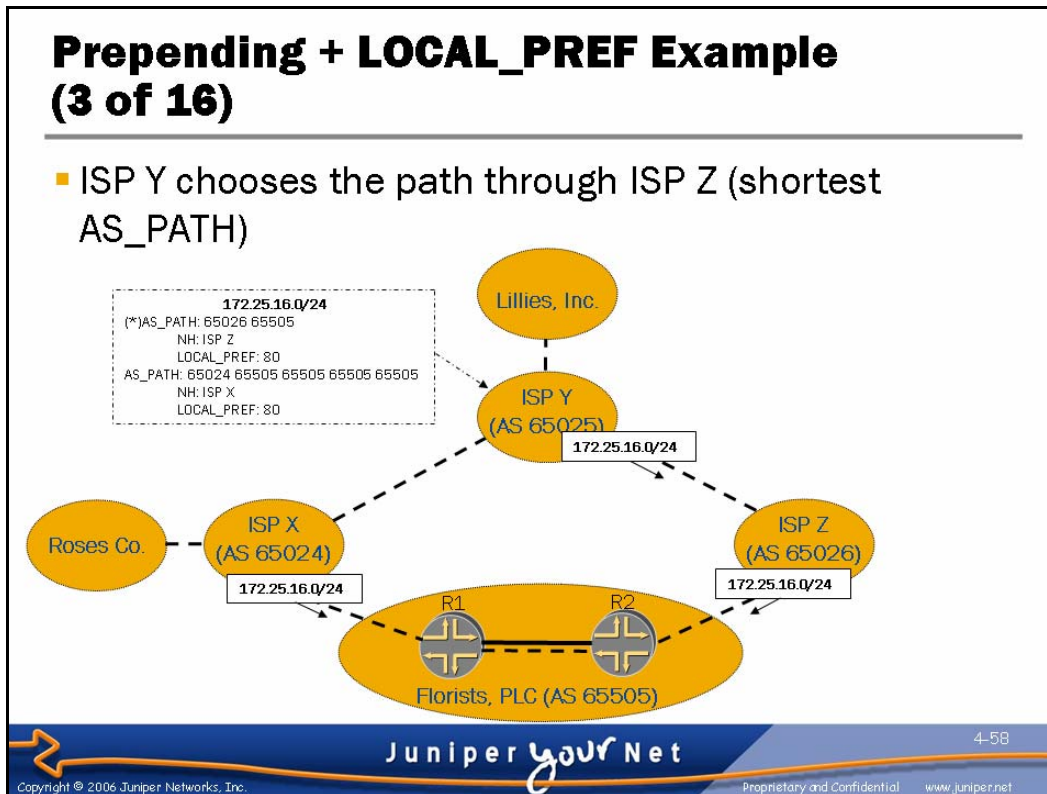
## Prepending + LOCAL\_PREF Example (2 of 16)

- ISPs X and Z both install routes for 172.25.16.0/24 and forward their routes to ISP Y
  - ISP X deletes the community before forwarding the announcement



### Prepending and Local Preference Example—ISP X and ISP Z Forward the Announcements

On this slide, we see that ISP X and ISP Z both begin to use the route they learned from Florists, PLC because it is the only announcement they have heard for this prefix thus far. When they choose to use this path, they also send an update to ISP Y to tell it about this path.

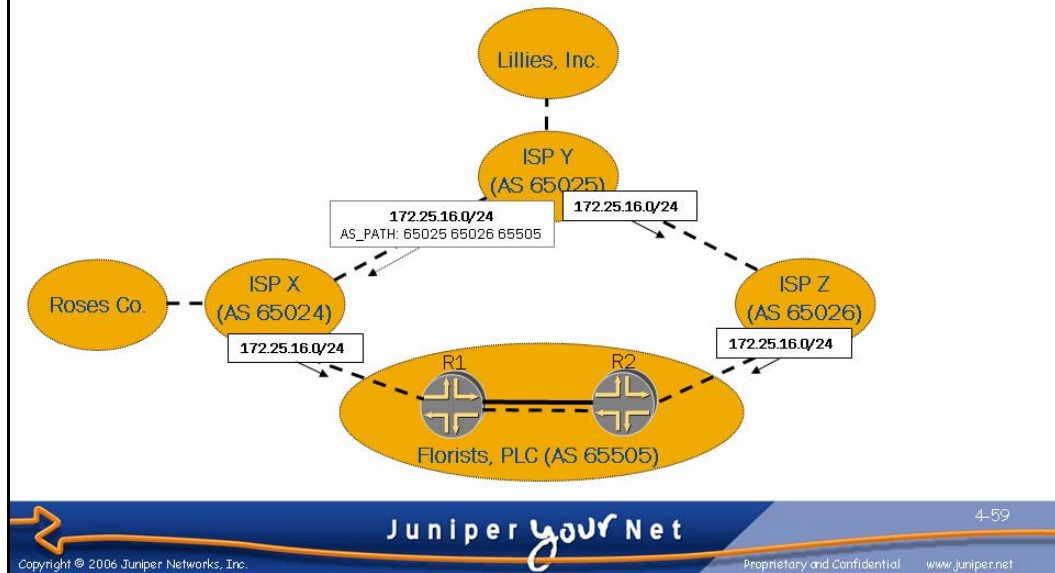


### Prepending and Local Preference Example—ISP Y Chooses a Path

ISP Y receives both announcements, but it chooses the one with the shorter AS path, which is the announcement from ISP Z. So, it installs a route for the Florists, PLC prefix pointing to ISP Z.

## Prepending + LOCAL\_PREF Example (4 of 16)

- ISP Y sends the announcement to ISP X

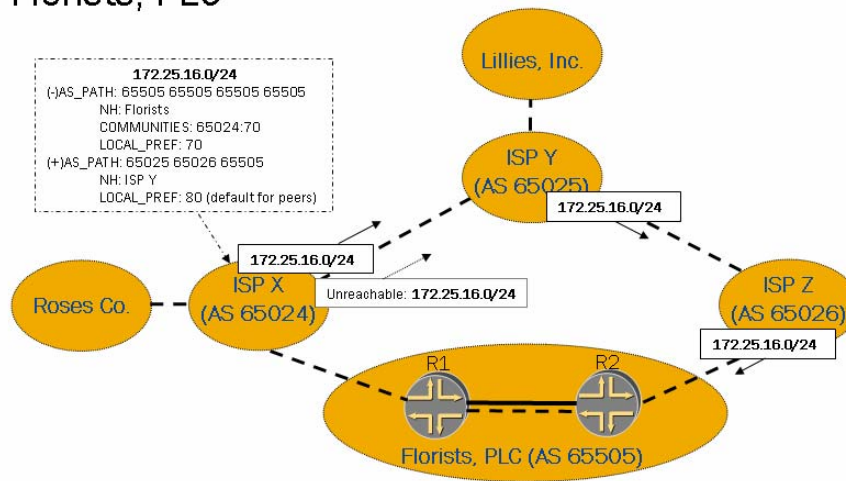


### Prepending and Local Preference Example—ISP Y Sends an Update to ISP X

Because ISP Y chose to use the path through ISP Z rather than the one through ISP X, ISP Y sends an announcement to ISP X about the path to Florists, PLC through ISP Z. ISP Y does *not* send an announcement to ISP Z about the path through ISP X because ISP Y did not choose the path through ISP X as the best path.

## Prepending + LOCAL\_PREF Example (5 of 16)

- ISP X begins to use the announcement through ISP Y (LOCAL\_PREF) and sends an update to ISP Y to withdraw its announcement about the path direct to Florists, PLC



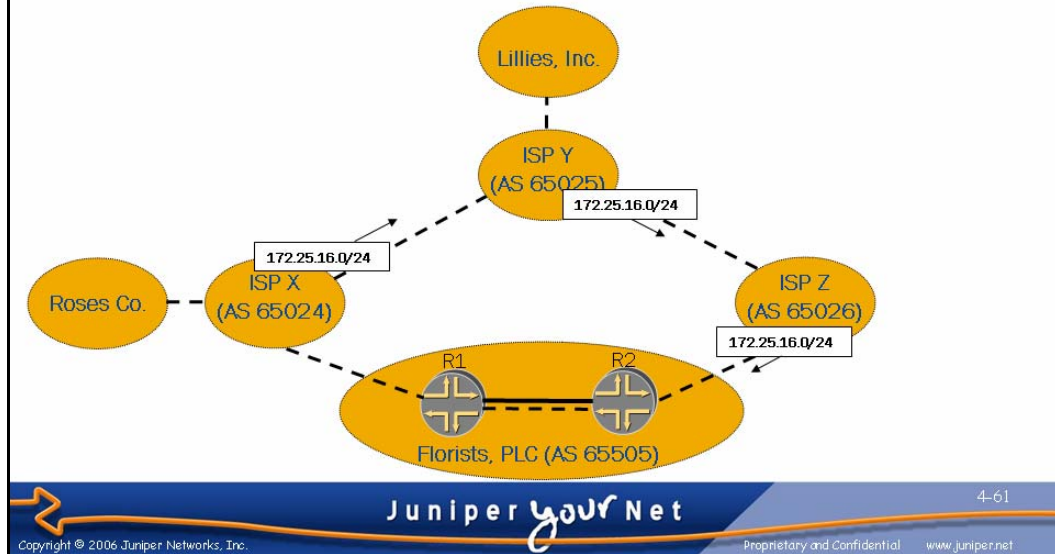
### Prepending and Local Preference Example—ISP X Chooses a Path

On this slide, you see the routing table entries in ISP X for the Florists, PLC prefix. Notice from the plus sign (+) next to the path through ISP Y and ISP Z that ISP X chose this as the new active route for this prefix.

When ISP X decides to use this path, it sends an update to ISP Y to explicitly withdraw the previous path it advertised to ISP Y. Because ISP Y was not using this path, it simply deletes the inactive route from the routing table and takes no further action.

## Prepending + LOCAL\_PREF Example (6 of 16)

- Traffic now follows desired primary path

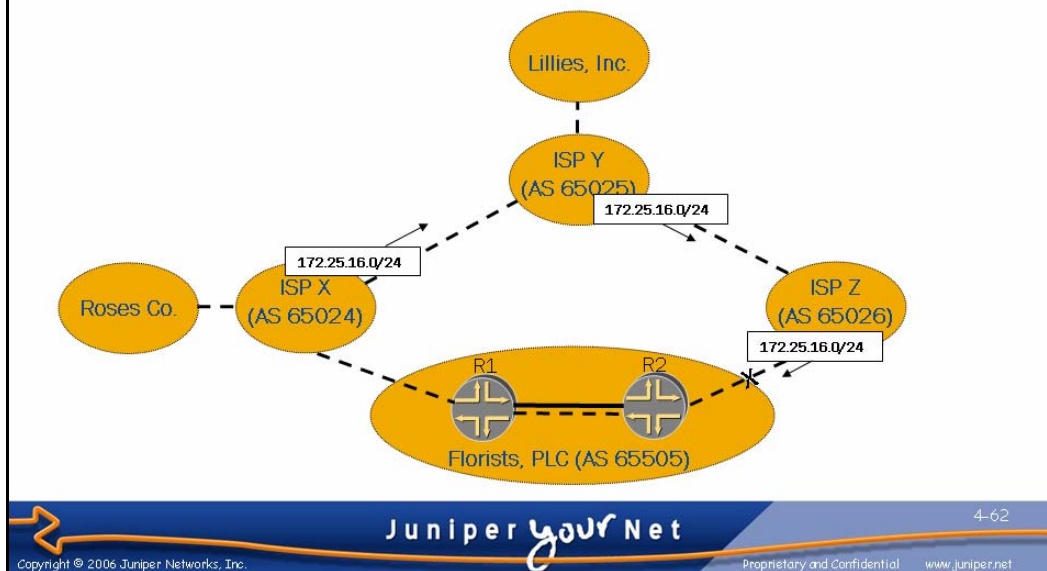


### Prepending and Local Preference Example—Normal Operation

As you can see on the slide, this scenario has resulted in a *normal* operating mode where all traffic follows the primary path and no traffic follows the secondary path.

## Prepending + LOCAL\_PREF Example (7 of 16)

- Failure on R2-ISP Z link

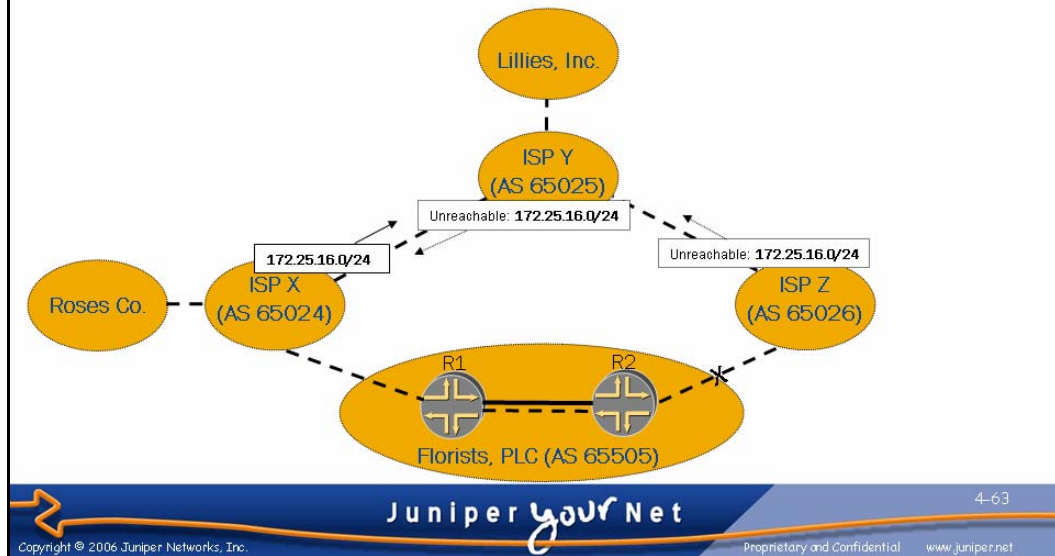


### Prepending and Local Preference Example—Failure of the Primary Path

We now consider the result of a failure of the primary path. In this case, we assume that some sort of failure happened on the physical link between ISP Z and R2.

## Prepending + LOCAL\_PREF Example (8 of 16)

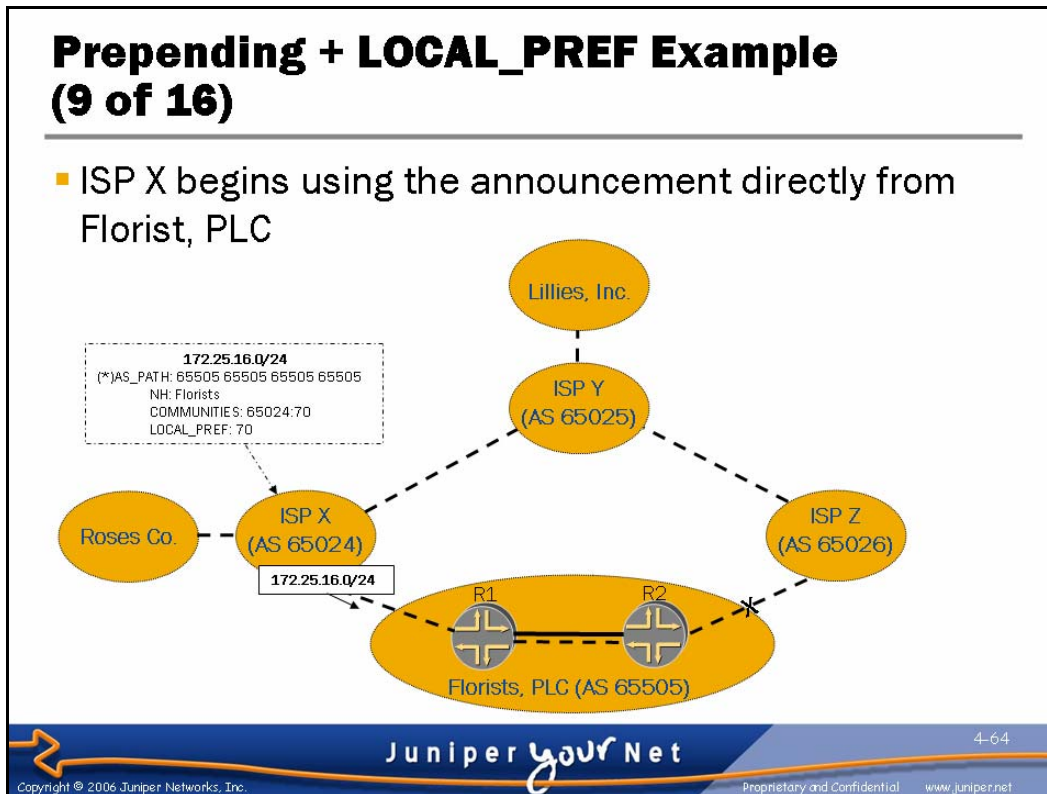
- ISP Z withdraws the announcement from ISP Y
  - ISP Y in turn withdraws the announcement from ISP X



### Prepending and Local Preference Example—ISP Z Withdraws the Route

When the link between ISP Z and R2 fails and the BGP session is dropped, ISP Z responds by deleting the routes that were received over this BGP session. Because the route received over this session was its active route to Florists, PLC, it also sends an update to ISP Y to explicitly withdraw the path to this prefix. When it receives the withdrawal, ISP Y responds by deleting this route from its routing table and by sending an update to ISP X to explicitly withdraw the path to this prefix. This reaction leaves both ISP Y and ISP Z without a path to reach the Florists, PLC prefix.





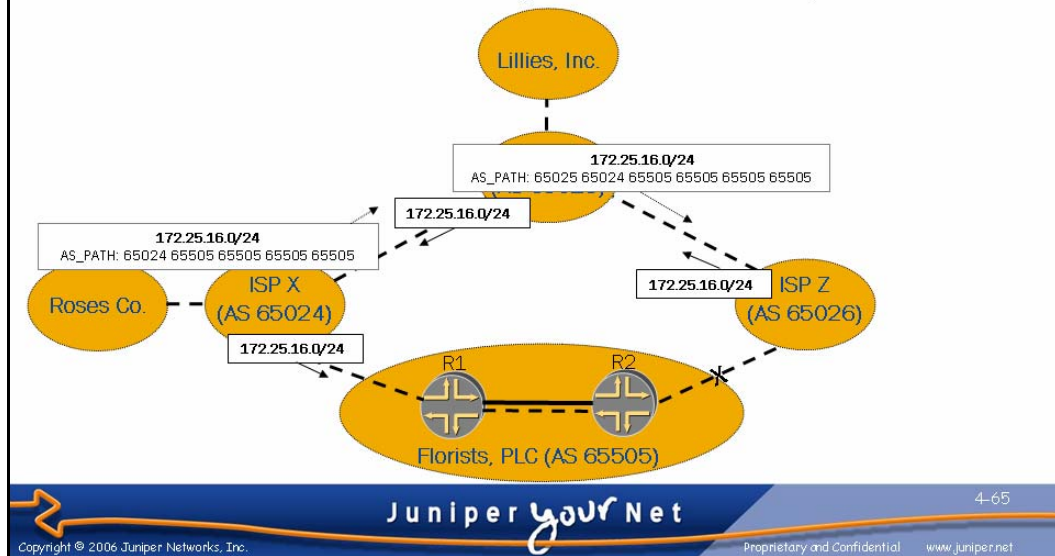
### Prepending and Local Preference Example—ISP X Chooses the Secondary Path

Once ISP X receives the update from ISP Y withdrawing the path to Florists, PLC through ISP Y and ISP Z, it removes that route from its routing table. Because it still has a route to this prefix that it received directly from Florists, PLC, it begins to use this as the active route.



## Prepending + LOCAL\_PREF Example (10 of 16)

- ISP X advertises the best path to ISP Y, who uses it and advertises it to ISP Z (who also uses it)

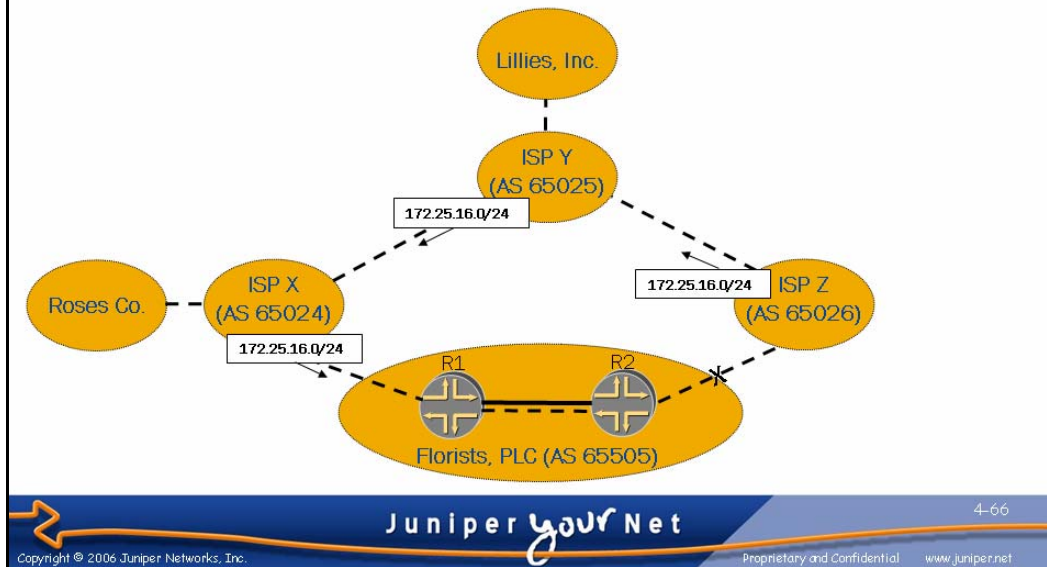


### Prepending and Local Preference Example—ISP X Sends Updates

Once it installs the route directly to Florists, PLC as the active route, it announces the route to ISP Y. Because ISP Y has no other routes to Florists, PLC, it installs the route as the active route and sends an update to ISP Z to announce the path. Because ISP Z also has no other routes to Florists, PLC, it also installs the route as the active route.

## Prepending + LOCAL\_PREF Example (11 of 16)

- Traffic now follows the desired secondary path



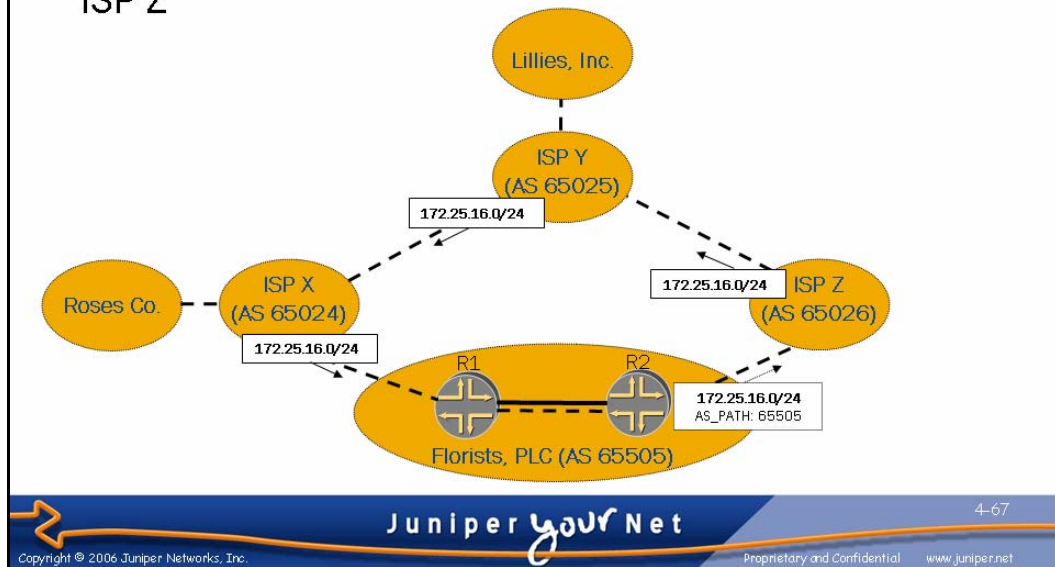
### Prepending and Local Preference Example—Failure Operation

As you can see on the slide, this scenario results in a failure mode where all networks can continue to reach Florists, PLC, and traffic follows the secondary path.

We now consider the failback from a failure mode to normal operation.

## Prepending + LOCAL\_PREF Example (12 of 16)

- R2-ISP Z link is restored, R2 sends announcement to ISP Z

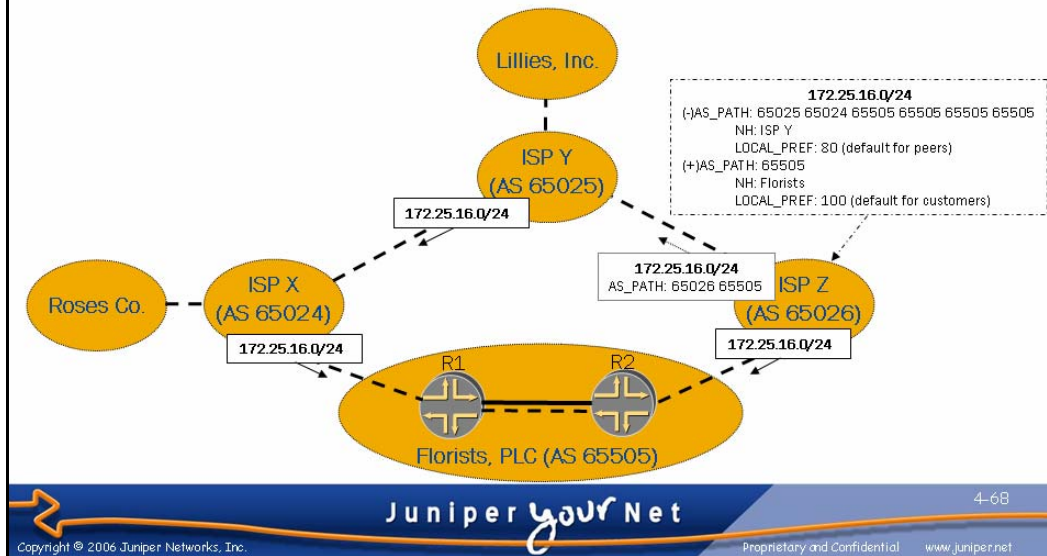


### Prepending and Local Preference Example—Restoration

Once the link between R2 and ISP Z is restored, the BGP session is reestablished, and Florists, PLC again announces its prefix to ISP Z.

## Prepending + LOCAL\_PREF Example (13 of 20)

- ISP Z uses the announcement from Florists, PLC and sends the announcement to ISP Y

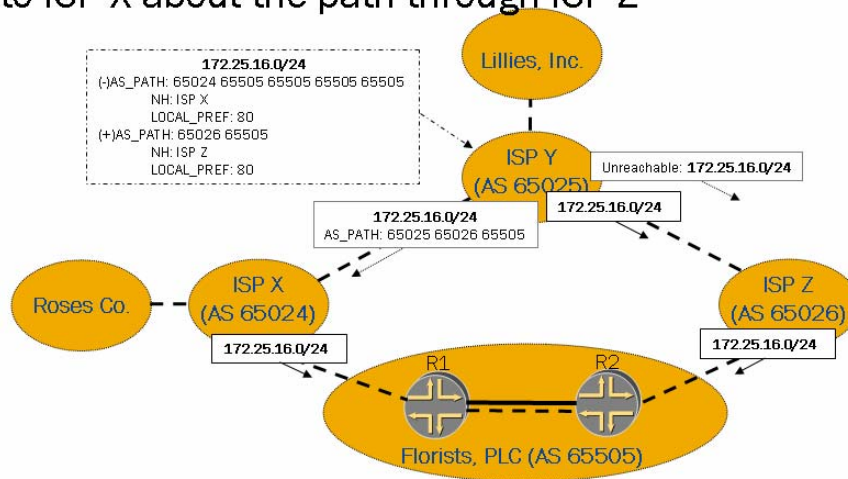


### Prepending and Local Preference Example—ISP Z Chooses a Path

When ISP Z receives the announcement from Florists, PLC, it installs it in its routing table. Because this announcement has a higher local preference than the announcement from ISP Y, ISP Z chooses this route to be the active route. It then sends an update to ISP Y to announce this path.

## Prepending + LOCAL\_PREF Example (14 of 16)

- ISP Y chooses the path through ISP Z (shortest AS\_PATH), so it sends an update to ISP Z to withdraw the path through ISP X and sends an announcement to ISP X about the path through ISP Z

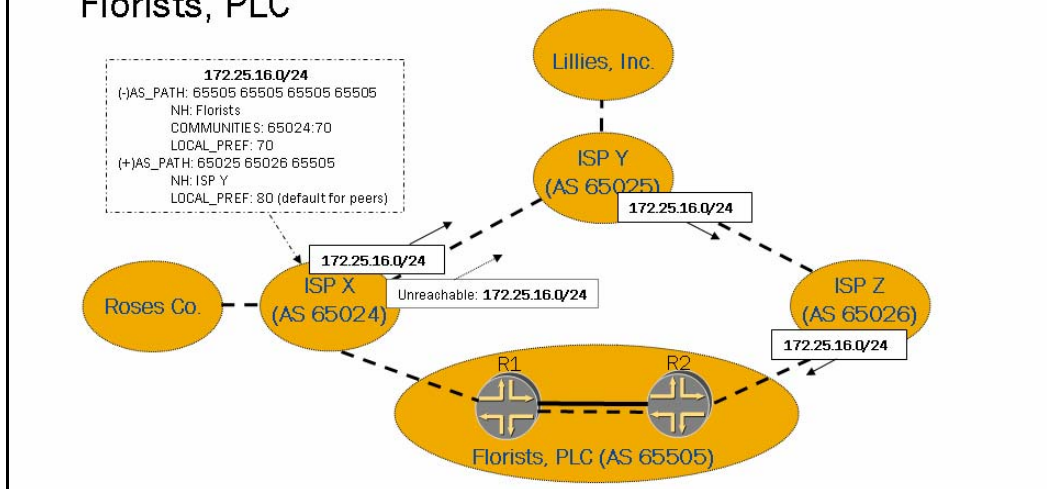


### Prepending and ISP Y Chooses a Path

When ISP Y receives the announcement about the path through ISP Z, it installs it in its routing table. Because the path through ISP Z has a shorter AS path length than the path through ISP X, ISP Y chooses the path through ISP X as the active route. Once it does this, it sends an update to ISP Z to explicitly withdraw the path to Florists, PLC through ISP X, and it also sends an update to ISP X to announce the path through ISP Z.

## Prepending + LOCAL\_PREF Example (15 of 16)

- ISP X begins to use the announcement through ISP Y (LOCAL\_PREF) and sends an update to ISP Y to withdraw its announcement about the path direct to Florists, PLC



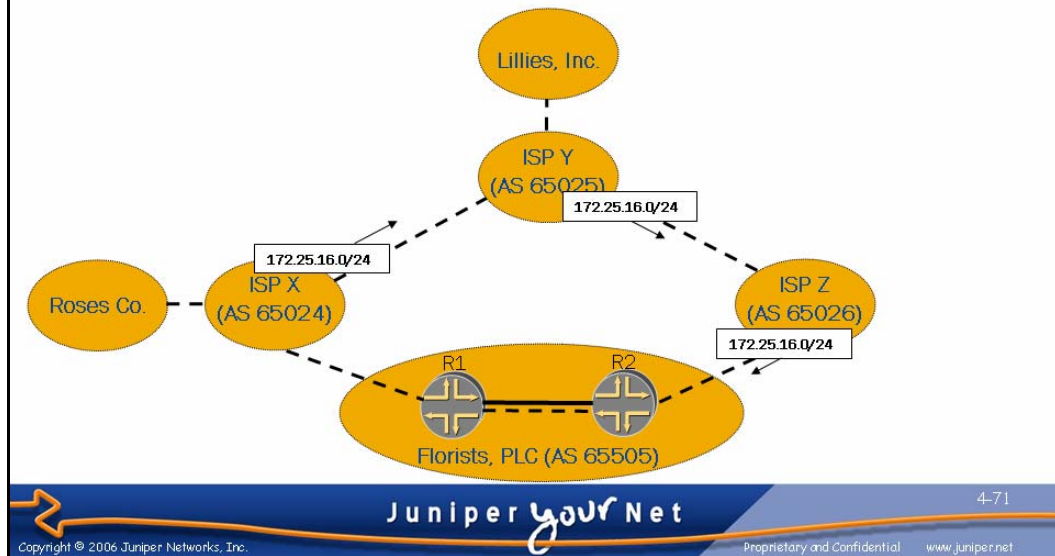
### Prepending and Local Preference Example—ISP X Chooses a Path

The slide shows the routing table entries in ISP X for the Florists, PLC prefix. Notice from the plus sign (+) next to the path through ISP Y and ISP Z that ISP X chose this route as the new active route for this prefix.

When ISP X decides to use this path, it sends an update to ISP Y to explicitly withdraw the previous path it advertised to ISP Y. Because ISP Y was not using this path, it simply deletes the inactive route from the routing table and takes no further action.

## Prepending + LOCAL\_PREF Example (16 of 16)

- Traffic is now again following the desired primary path



### Prepending and Local Preference Example—Normal Operation Restored

As you can see on the slide, *normal* operation is restored, with all traffic following the primary path and no traffic following the secondary path.

## Primary/Secondary Inbound: Configuration Example

### ■ Sample configuration excerpts:

```

protocols {
  bgp {
    group primary-isp {
      [...]
      export routes-to-ISP;
    }
    [...]
    group secondary-isp {
      [...]
      export [ set-backup routes-to-ISP ];
    }
    [...]
  }
}

policy-options {
  prefix-list announce-to-ISP {
    172.25.16.0/24;
  }
  policy-statement routes-to-ISP {
    from {
      prefix-list announce-to-ISP;
    }
    then accept;
  }
  policy-statement set-backup {
    then {
      community set ISP-X-localpref-70;
      as-path-prepend "65505 65505 65505";
    }
  }
  community ISP-X-localpref-70 members 65024:70;
}

```



### Sample Configuration: Primary/Secondary Inbound

The configuration on the slide shows the end state achieved by Flowers, PLC in this case study.



## Primary/Secondary Case Study Conclusion

- Either AS path prepending or communities to set local preference alone does not provide a strict primary/secondary inbound routing policy
- Must think through the effects of routing policies



### Use Both AS Path Prepending and Provider Local Preference

To ensure that traffic flows as you want it to flow in all situations, you must implement strict primary/secondary inbound routing policies using both AS path prepending and causing a provider to modify the local-preference value of announcements together. In normal operation, the lower local preference on the announcement via the secondary path causes the secondary ISP to prefer the primary path. After failures, the AS path prepending causes intermediate ISPs to prefer the primary path.

### Importance of Evaluating Routing Policies

You must thoroughly consider routing policies as you are implementing them. You must also thoroughly analyze the routing decisions that routers on the Internet (and, particularly, your ISPs and any intermediate ISPs between your ISPs) make in normal circumstances, in failure conditions, and in the recovery from failure conditions.

## Review Questions

1. Name several routing policies. What are the advantages and disadvantages of each policy?
2. Why should you use both AS path prepending and communities to modify provider local preference to establish a strict primary/secondary inbound routing policy?



### This Chapter Discussed:

- Three common routing policies used in the enterprise environment;
- The way attribute modifications affect routing decisions; and
- Implementing a routing policy for both inbound and outbound traffic using BGP.

## Lab 2: BGP Configuration

---

- Configure IBGP and EBGP sessions.
- Configure OSPF on a virtual router.
- Configure IGP-BGP interaction.
- Establish primary/secondary inbound and outbound routing policy.



### Lab 2: BGP Configuration

This slide lists the objectives for this lab.





# **Advanced Juniper Networks Routing in the Enterprise**

## **Chapter 5: Transitioning Between IGPs**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - Describe three IGP transition strategies
  - Transition between IGPs with minimal or no network disruption while maintaining network stability



### This Chapter Discusses:

- Three IGP transition strategies; and
- Transitioning between IGPs with minimal or no network disruption while maintaining network stability.

## Agenda: IGP Transition

---

- Transition Overview
- Overlay Transition
- Route Redistribution Transition
- Integrated Transition



### Transition Overview

The slide lists the topics we cover in this chapter. We discuss the highlighted topic first.

## IGP Transition Overview

- Goal is to move from one IGP to another IGP while:
  - Ending with a well designed network (including a good IGP design)
  - Maintaining network stability
  - Minimizing downtime
- Transition strategies are generic and can be applied to any beginning and ending IGPs
- Common elements:
  - Use route preferences to ensure that routes from desired protocol are used at each stage of transition
  - Export appropriate routes to each protocol



### IGP Transition Overview

There are many reasons why you might decide to transition your network from using one interior gateway protocol (IGP) to using a different IGP. When you find it necessary to move from one IGP to another, you have several goals. Your primary goal is to end with a well designed network. When you design a network well, the IGP design is an integral part of the network design. So, moving between IGPs often requires you to make some network design changes to create a well designed network by the end of the transition process. Your secondary goal is to maintain network stability while transitioning IGPs. Your tertiary goal is to minimize downtime during the transition. This chapter does not directly teach good IGP design; however, it does focus on how to maintain network stability and minimize network downtime during the transition.

### Generic Transition Strategies

While you might decide to transition your network from using one IGP to using a different IGP for many reasons, a common reason is to move from proprietary protocols (such as Cisco's EIGRP) to protocols based on open standards (such as OSPF). In this chapter, we discuss generic transition strategies. The examples in the chapter show how to transition a network of routers running the JUNOS software from RIP to OSPF. However, the transition strategies are generic, and you can use them for any beginning and ending IGPs, and you can apply the concepts to routers running any operating system.

*Continued on next page.*



## Generic Transition Strategies (contd.)

For a more specific discussion of transitioning from EIGRP to OSPF, see the Juniper Networks white paper included as Appendix B: *EIGRP to OSPF Migration Strategies*.

## Common Elements

The IGP transition strategies contain a few common elements. Every IGP transition strategy requires that you know which IGP provides the active routing information for the network at every point. You control this information in the JUNOS software using route preferences. Additionally, most transition strategies require you to export appropriate routing information between the two IGPs so that you can ensure that all routers have access to full routing information.

## Route Preference

- Ranks routes received from different sources
- Primary criterion for selecting the active route
- Ranges from 0 to 4,294,967,295, with lower value preferred

Route Preference Values

Routing Information Source	Default Preference
Direct	0
Local	0
Static	5
OSPF internal	10
RIP	100
Aggregate	130
OSPF AS external	150
BGP (both EBGp and IBGP)	170

### Preferred Routing Information Sources

The router uses route preference to differentiate routes received from various routing protocols or routing information sources. Route preference is equivalent to administrative distance on other vendors' equipment. To ensure the routers make consistent routing decisions, you should set route preferences consistently on every router in the network (including other vendors' routers).

### Primary Tiebreaker

JUNOS software uses route preference as the primary criterion for selecting the active route. Preference values cause routes from certain information sources to be ranked more preferably than the same route received from another information source. The table at the bottom of the slide shows the default preference values for a selected set of routing information sources.

*Continued on next page.*

## Lower Is Better

Routing preference values can range from 0 to 4,294,967,295. The router prefers lower preference values over higher preference values. This command output demonstrates that a direct route with a preference of 0 is preferred over an OSPF internal route with a preference of 10:

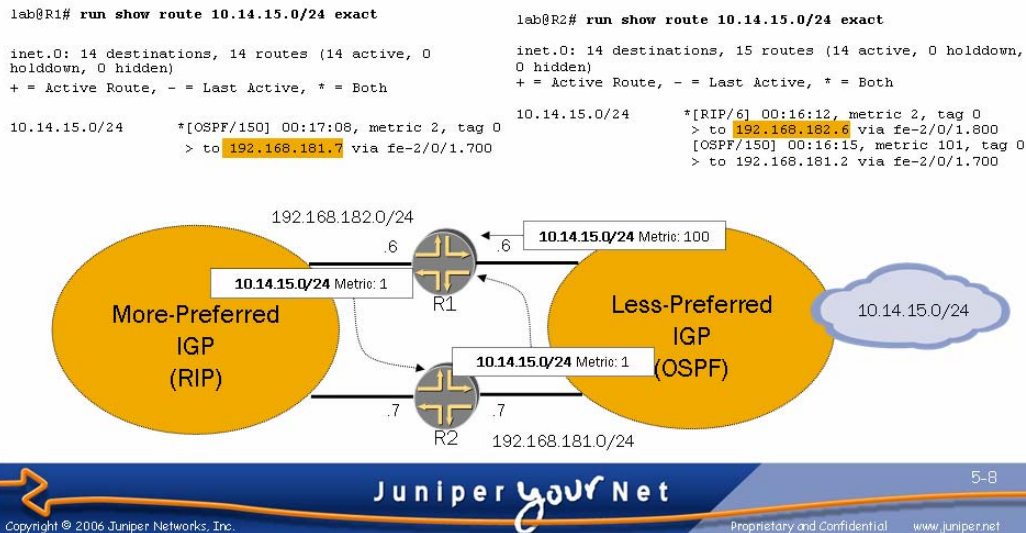
```
user@host> show route 10.251.254.130/31 exact

inet.0: 18 destinations, 19 routes (17 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

10.251.254.130/31  *[Direct/0] 1d 07:53:39
                   > via t1-4/0/0.0
                   [OSPF/10] 1d 07:53:32, metric 65
                   > via t1-4/0/0.0
```

## Exporting Routes

- Must avoid *double-exporting*, which can cause routing loops



### Avoiding Exporting Problems

You can cause problems when you indiscriminately export routes from one IGP to another. In particular, you can cause routing loops or route oscillation when you configure unconditional bidirectional exporting.

On the slide, a network administrator is transitioning from one IGP (RIP) to another IGP (OSPF). In an effort to provide redundancy during the transition, the network administrator configures two routers on the border between the old and new IGPs and configures both routers to export all RIP routes to OSPF and export all OSPF routes to RIP.

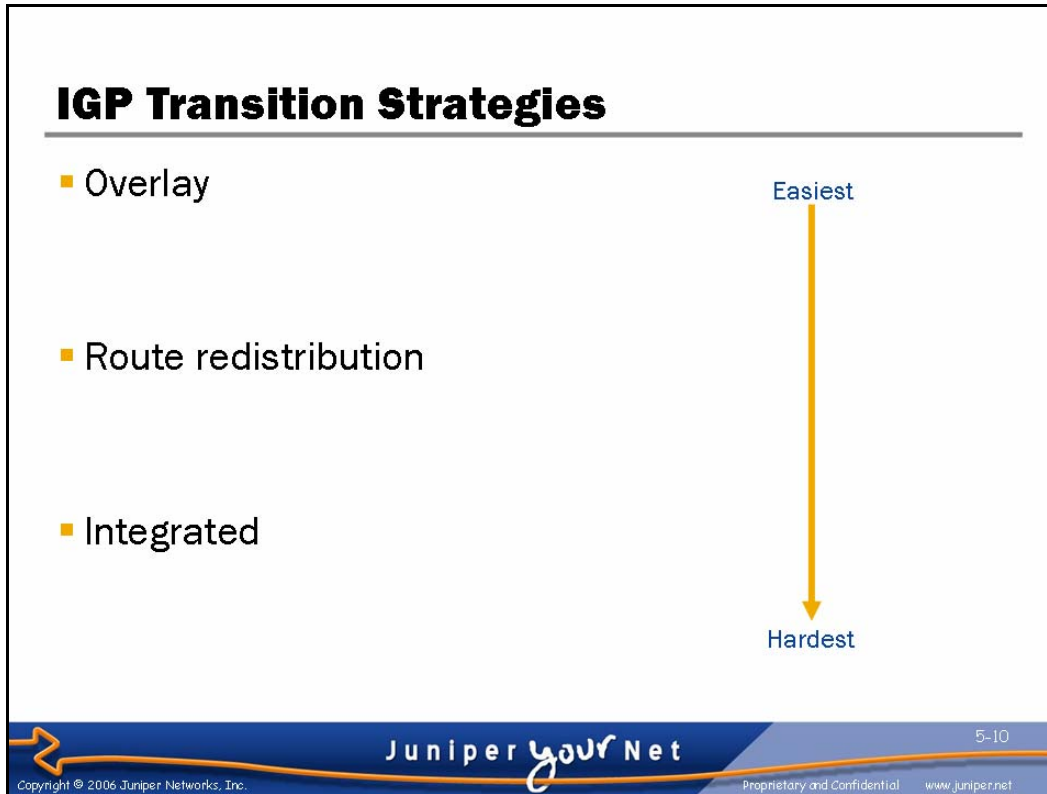
The slide shows the way the routers react when they receive a route via OSPF for the prefix 10.14.15.0/24. The OSPF route is an external Type 1 route with a metric of 100. When R1 and R2 receive the route, they install it in the routing table as the active route, export the route to RIP, and send a RIP advertisement. However, R1 accomplishes this process microseconds before R2. So, R2 receives the RIP route and installs it in the routing table. Because the routers in this network are configured to prefer RIP routes over OSPF external routes, R2 prefers the route it received from R1 via RIP. When R2 receives the route via RIP from R1, it installs it in the routing table as the active route, exports the route to OSPF (as an external Type 1 route with a metric of 1), and sends an OSPF LSA. When R1 receives the OSPF LSA, it examines the metrics and determines that the OSPF route via R2 is the best path to 10.14.15.0/24. It therefore installs that route as the active route in the routing table. It continues exporting this route to RIP, as it is configured to do. As you can see, this configuration has caused a routing loop to form!

*Continued on next page.*

### Avoiding Exporting Problems (contd.)

To avoid problems like this, you must configure bidirectional exporting very carefully. In particular, the safest way to configure bidirectional exporting is to configure very explicit export filters that only allow *expected* routes to be exported between each protocol. As you continue the transition, you must continually update these filters.

In the example on the slide, the routing loop (and routing information loop) would continue even if the router that originally advertised the 10.14.15.0/24 prefix in OSPF ceased making the announcement. The only way you can end the routing loop is to break the exporting loop. Configuring filters to only allow expected routes to be exported between each protocol would end the routing information loop.



### Overlay Transition

In the overlay transition, you configure all routers in the network to run the new IGP while they are still running the old IGP. Once you verify the proper operation of the new IGP, you configure all routers in the network to stop running the old IGP. This transition strategy is appropriate when all routers have the necessary resources (in particular, CPU and memory) to run both protocols simultaneously. Additionally, you must accomplish any network redesign necessary to accommodate the new IGP as part of the transition.

### Route Redistribution

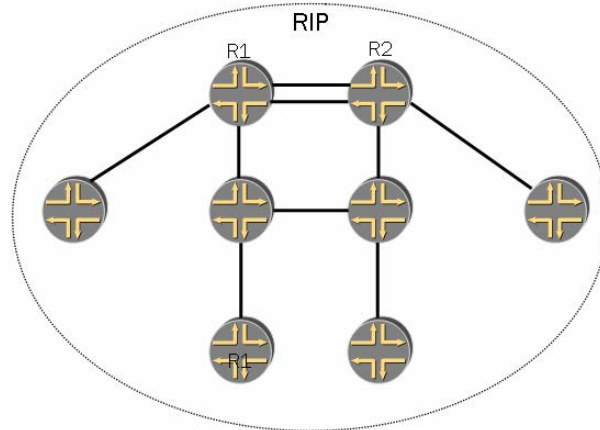
If all routers cannot run both protocols simultaneously or if you must make some topology changes as part of the IGP change, you can gradually migrate the network one portion at a time. For example, if you are migrating to OSPF, you might migrate one future OSPF area at a time. As you progress from one portion of the network to another, you must configure routers on the border of the two IGPs to conduct mutual route redistribution, exporting routes between the two IGPs.

*Continued on next page.*

## Integrated

If you must significantly redesign your existing topology to make it a well designed network with the new IGP, the integrated strategy might be the best approach. In this strategy, you create a new network core running the new IGP, connect it to the old network core, and export routes between the new and old IGPs. You then gradually migrate connections from the old core to the new core.

## Sample Network Topology: Start

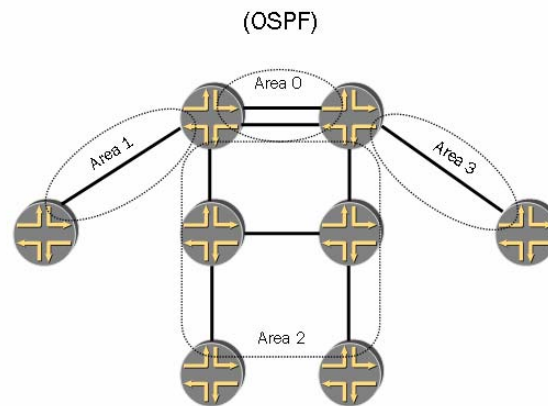


### Sample Topology: Starting State

This slide shows the starting topology this chapter uses for its examples. In this chapter, the examples start with a network using RIP as the IGP.



## Sample Network Topology: End



### Sample Network Topology: End State

The slide shows the ending topology of the examples in this chapter. The examples show transitioning the network to use OSPF as the IGP.

## Agenda: IGP Transition

---

- Transition Overview
- Overlay Transition
- Route Redistribution Transition
- Integrated Transition



### Overlay Transition

The slide highlights the topic we discuss next.

## Overlay Transition Steps

1. Configure all routers to assign the existing IGP a better route preference than the new IGP
2. Configure the new IGP on all routers
3. Verify that all routers have full routing information
4. Delete the old IGP



### Step 1: Configuring Route Preferences

You begin the overlay transition by configuring the router to assign the existing IGP a better (in other words, numerically lower) route preference than the new IGP. This configuration ensures that the router continues to use the existing IGP for routing information while you configure the new IGP.

### Step 2: Configuring the New IGP

You continue the overlay transition by configuring the new IGP on all routers. You should configure it with the desired end state (or at least as close to the desired end state as possible).

### Step 3: Verifying Routing Information

The next step in the overlay transition process is for you to verify that all routers are receiving full routing information via the new IGP. At the same time, you should verify that the router is making the routing decisions you expect and want it to make.

### Step 4: Deleting the Old IGP

The final step is for you to deactivate the old IGP on all routers, verify the proper operation of the new IGP, and delete the old IGP configuration. You should proceed from one end of the network to another to ensure that untransitioned routers are not in the path between two routers that are already transitioned.

## Overlay Transition Example: Step 1

- Configure all routers to assign RIP a better route preference than OSPF
  - Because OSPF internal routes have a preference of 10 and static routes have a preference of 5, you choose 7
  - Assuming all RIP peers were in the group *peer-routers*, use the following command on all routers:  
`set protocols rip group peer-routers preference 7`



### Overlay Transition Example: Step 1

This slide and the next few slides show an example of how you can transition the sample network from RIP to OSPF.

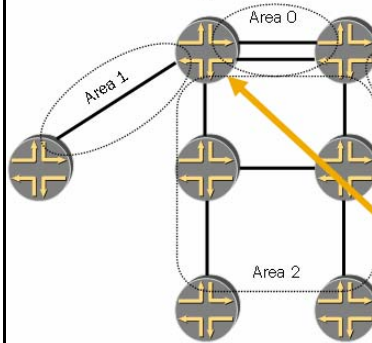
The first step in the overlay transition is for you to configure the router to assign a better route preference to the existing IGP than the new IGP. By configuring the old IGP to have a better route preference than the new IGP, you avoid extra cleanup and verification steps at the end of the transition. If you instead configure the routers to modify the new IGP's route preferences to be worse than the old IGP, you must remove the configuration that modified the new IGP's route preference at the end of the transition process. (And that removal could cause the router to make different route selection decisions in some cases. Therefore, you would really need to verify each router's configuration and routing decisions after making this change.)

In general, you want to configure the new routing protocol as it will exist at the end of the transition and make any temporary changes necessary for the transition to the existing IGP's configuration. For these reasons, it is best to begin by configuring all routers to assign the existing IGP a better route preference than the new IGP.

Looking at the chart on page 5-6, note that OSPF internal routes have a default preference of 10, so you must choose a preference for RIP routes that is numerically lower than 10. Also note that static routes have a default preference value of 5. Therefore, you want to choose a preference for RIP routes that is numerically higher than 5. In this case, you choose 7 and configure all routers using the command shown on the slide.

## Overlay Transition Example: Step 2

- Configure the new IGP on all routers
  - Configure OSPF as it will exist on the router



```

protocols {
  ospf {
    area 0.0.0.0 {
      interface fe-1/0/0.0;
      interface fe-2/0/0.0;
      interface lo0.0 {
        passive;
      }
    }
    area 0.0.0.1 {
      stub default-metric 1;
      area-range 10.14.8.0/22;
      interface se-3/0/0.0;
    }
    area 0.0.0.2 {
      area-range 10.14.12.0/22;
      interface se-3/0/1.0;
    }
  }
}

```



Juniper your Net

5-17

Copyright © 2006 Juniper Networks, Inc.

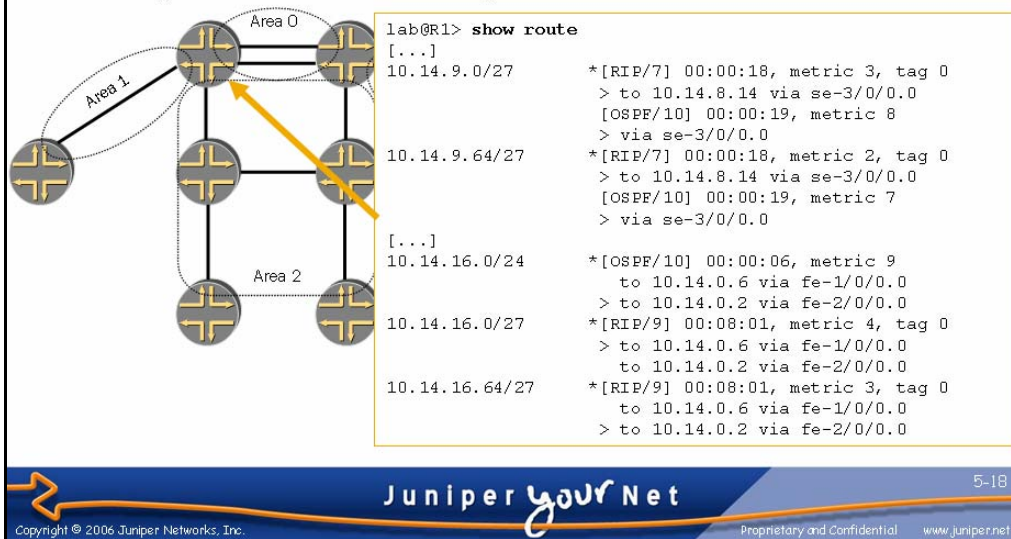
Proprietary and Confidential www.juniper.net

## Overlay Transition Example: Step 2

The next step is for you to configure OSPF on all the routers. Because you configured RIP routes to have a lower preference, the routers will continue using the RIP routes. Therefore, configuring OSPF does not change the router's routing decisions.

## Overlay Transition Example: Step 3

- Verify that all routers have full routing information
  - Pay attention to summary routes

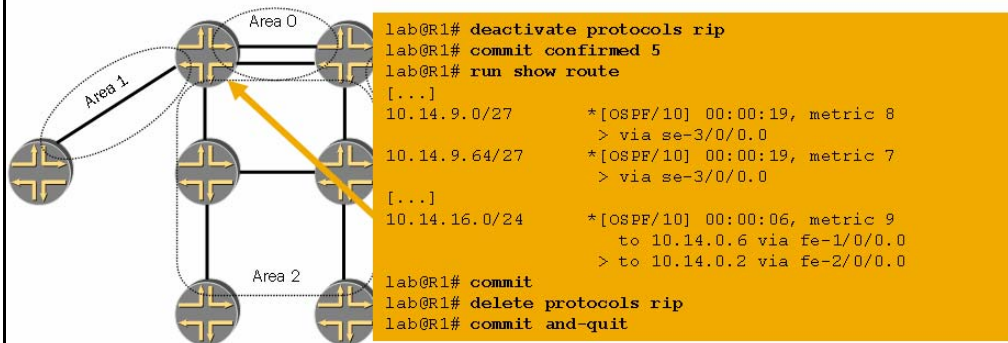


### Overlay Transition Example: Step 3

Now that you have configured both routing protocols to run side by side, you confirm that you are receiving the same routing information through both protocols. Note that the previous slide showed that you configured OSPF summary routes for the OSPF areas. Because of this configuration, you notice that the router is receiving summary routes through OSPF that it is not receiving through RIP (on the slide, R2 is summarizing 10.14.16.0/24 from Area 3). Likewise, the router is receiving more specific routes through RIP that it is not receiving through OSPF (on the slide, 10.14.16.0/27 and 10.14.16.64/27). Because each area border router (ABR) has full routing information for the area, you should ensure that each ABR is receiving the same routing information through both RIP and OSPF for the routes within each OSPF area.

## Overlay Transition Example: Step 4

- Delete the old IGP



### Overlay Transition Example: Step 4

Once you verify that you are receiving the same routing information through both the old and new IGPs, you are ready to start using the new IGP. To do this, gradually deactivate the old IGP throughout your network. On a router running the JUNOS software, you should follow the procedure on the slide. First, deactivate the protocol (allowing it to be quickly reactivated, should the need arise). Then, commit the configuration. If you are accessing the router in-band, you should use the **commit confirmed** command. Anytime you make major routing protocol changes using Telnet, it is always advisable to use **commit confirmed** to ensure that the router returns to a working state should a problem arise. After you confirm the router is still reachable and has full routing information, run the **commit** command again to confirm the configuration changes.

After you are confident that you do not need to reactivate the old IGP, delete the old IGP's configuration.

## Agenda: IGP Transition

---

- Transition Overview
- Overlay Transition
- Route Redistribution Transition
- Integrated Transition



### Route Redistribution Transition

The slide highlights the topic we discuss next.



## Route Redistribution Transition Steps

1. Configure all routers to assign the existing IGP a better route preference than the new IGP
2. Configure the new IGP on the ABRs
  - Establish redistribution with controls
3. Gradually convert routers to use the new IGP
  - Might require concurrent topology changes
4. Delete the old IGP on the ABRs



### Step 1: Configuring Route Preferences

You begin the route redistribution transition by configuring the router to assign the existing IGP a better (in other words, numerically lower) route preference than the new IGP. This configuration ensures that the router continues to use the existing IGP for routing information while you configure the new IGP.

### Step 2: Configuring the New IGP on the ABRs

You continue the transition by configuring the new IGP on the future ABRs. At the same time, you configure the two IGPs to export their routes to each other. You must configure appropriate policies to control the routes that the router exports.

### Step 3: Converting Routers Gradually

The next step in the transition process is to gradually convert the existing routers to use the new IGP. You deactivate the old IGP at the same time you activate the new one. At the same time, you make any necessary changes to the network topology.

### Step 4: Deleting the Old IGP on the ABRs

Once you configure the new routing protocol on all routers, the final step is for you to deactivate the old IGP on the ABRs, verify the proper operation of the network, and delete the old IGP configuration completely.

## Route Redistribution Example: Step 1

- Configure all routers to assign RIP a better route preference than OSPF
  - Because OSPF internal routes have a preference of 10 and static routes have a preference of 5, you choose 7
  - Assuming all RIP peers were in the group *peer-routers*, use the following command on all routers:  
**set protocols rip group *peer-routers* preference 7**



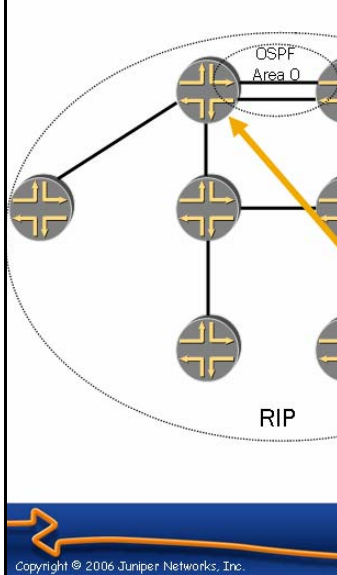
### Route Redistribution Example: Step 1

This slide and the next few slides show an example of how you can transition the sample network from RIP to OSPF using the route redistribution transition strategy.

Similar to the overlay transition strategy, the first step in the transition is for you to configure the router to assign a better route preference to the existing IGP than the new IGP. By configuring the old IGP to have a better route preference than the new IGP, you avoid extra cleanup and verification steps at the end of the transition. In this case, you choose to configure the router to assign RIP routes a preference of 7 and configure all routers using the command shown on the slide. (See page 5-16 for more details on the reasons for configuring the existing IGP with a high route preference and the reasons for choosing the route preference of 7 for RIP routes.)

## Route Redistribution Example: Step 2

- Configure the new IGP on the ABRs



```

lab@R1# show protocols
ospf {
  export rip-to-ospf;
  area 0.0.0.0 {
    interface fe-1/0/0.0;
    interface fe-2/0/0.0;
    interface lo0.0 {
      passive;
    }
  }
}
rip {
  group peer-routers {
    preference 7;
    export ospf-to-rip;
    neighbor fe-1/0/0.0;
    neighbor fe-2/0/0.0;
    neighbor se-3/0/0.0;
    neighbor se-3/0/1.0;
  }
}

lab@R1# show policy-options
prefix-list rip-routes {
  10.14.9.0/27;
  10.14.9.64/27;
  [...]
}
[...]

policy-statement ospf-to-rip {
  term unconverted-routes {
    from {
      protocol ospf;
      prefix-list rip-routes;
    }
    then reject;
  }
  term converted-routes {
    from protocol ospf;
    then {
      metric 1;
      accept;
    }
  }
}
policy-statement rip-to-ospf {
  term unconverted-routes {
    from {
      protocol rip;
      prefix-list rip-routes;
    }
    then {
      metric 10;
      external {
        type 1;
      }
      accept;
    }
  }
}

```

Copyright © 2006 Juniper Networks, Inc. 5-23

## Route Redistribution Example: Step 2

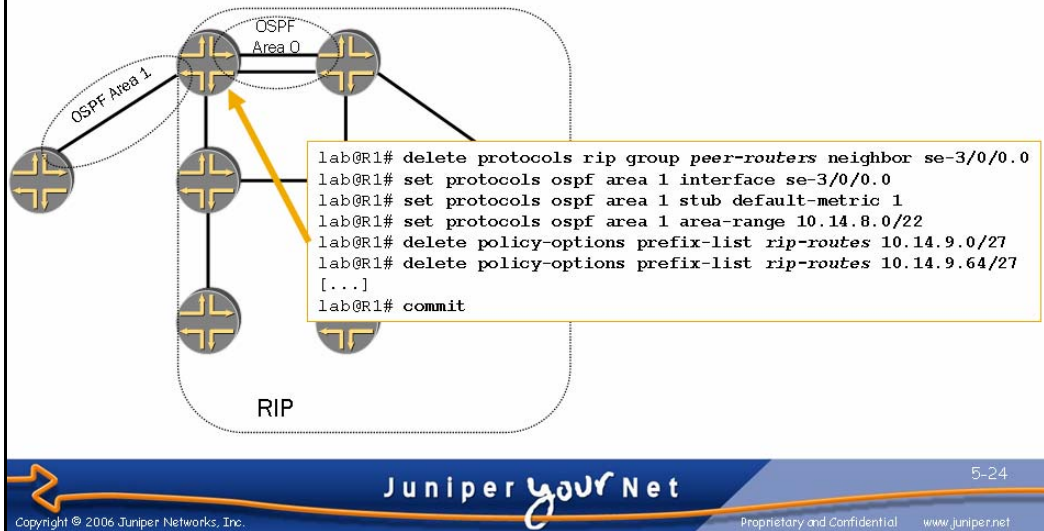
The next step is for you to configure OSPF on the ABRs. Because you configured RIP routes to have a lower preference, the routers continue using the RIP routes. Therefore, configuring OSPF does not change the routers' routing decisions.

Because the other routers only run either RIP or OSPF at any given time, you must configure the router to export RIP routes to OSPF, and *vice versa*. You must configure policies that control the routes that are exported between the two protocols to prevent routing information loops and route oscillation (see page 5-8).

On the slide is R1's configuration. You configure a prefix list called *rip-routes* that contains the list of routes currently available via RIP. (You should be able to compile such a list through your advanced preparation.) You configure a policy called *rip-to-ospf* that instructs the router to accept all routes it learns via RIP that are in the *rip-routes* prefix list, and you configure the router to use this policy as the OSPF export policy. You also configure a policy called *ospf-to-rip* that instructs the router to accept all routes it learns via OSPF that are *not* in the *rip-routes* prefix list, and you configure the router to use this policy as the RIP export policy. As you migrate routes from RIP to OSPF, you must delete those prefixes from the *rip-routes* prefix list on all ABRs to ensure that the routers export the proper routes from one protocol to another.

## Route Redistribution Example: Step 3

- Gradually convert routers to use the new IGP
  - Might require concurrent topology changes



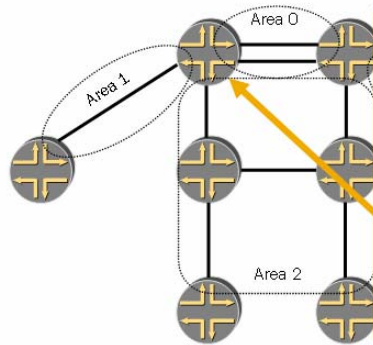
### Route Redistribution Example: Step 3

Once you configure OSPF on the ABRs, you now proceed to delete the old routing protocol and activate the new routing protocol on the remaining routers, proceeding one area at a time. Note that, unlike the overlay transition strategy, the route redistribution transition strategy does not require any routers other than ABRs to run two IGPs simultaneously. At the same time you switch routing protocols within an area, you should also make any associated topology changes.

The slide shows the changes required on R1. During this transition, you are migrating the routers that will be in OSPF Area 1 from RIP to OSPF. You delete the RIP configuration for the interface that will be in Area 1 (*se-3/0/0.0*). Then, you configure Area 1 and configure the appropriate interface to be in that area. You modify the *rip-routes* prefix list by removing the interfaces that you are transitioning from RIP to OSPF. Finally, you commit the configuration changes. After you make all changes, you verify on all routers that the appropriate routes are still available, both via OSPF (on the transitioned routers) and RIP (on the untransitioned routers).

## Route Redistribution Example: Step 4

- Delete the old IGP on the ABRs



```

lab@R1# deactivate protocols rip
lab@R1# commit confirmed 5
lab@R1# run show route
[...]
10.14.9.0/27          *[OSPF/10] 00:00:19, metric 8
                     > via se-3/0/0.0
10.14.9.64/27        *[OSPF/10] 00:00:19, metric 7
                     > via se-3/0/0.0
[...]
10.14.16.0/24         *[OSPF/10] 00:00:06, metric 9
                     to 10.14.0.6 via fe-1/0/0.0
                     > to 10.14.0.2 via fe-2/0/0.0

lab@R1# commit
lab@R1# delete protocols rip
lab@R1# commit and-quit

```

### Route Redistribution Example: Step 4

After you finish migrating all the routers to run OSPF, you complete the transition by deleting RIP from the ABRs.

## Agenda: IGP Transition

---

- Transition Overview
- Overlay Transition
- Route Redistribution Transition
- ➔ Integrated Transition



### Integrated Transition

The slide highlights the topic we discuss next.

## Integrated Transition Steps

1. Deploy a new network core running the new IGP
2. Connect the old and new network cores
  - Establish redistribution with controls
3. Gradually convert routers to the new network core
4. Decommission the old network core



### Step 1: Deploying a New Network Core

You begin the integrated transition by deploying a new network core. You configure the new network core to run the new IGP.

### Step 2: Connecting the Old and New Networks

You continue by connecting the old and new networks together. You either configure the new network core to run the old IGP or configure the old network core to run the new IGP. Usually, you configure the old network core to run the new IGP. By configuring the old network core to run the new IGP, you avoid adding temporary configuration to the old IGP. Additionally, when transitioning from a proprietary protocol to an open standards-based protocol, the new network core might not be able to run the proprietary protocol. You configure the routers running both protocols to export routes between the old and new protocols with proper controls.

### Step 3: Converting Routers Gradually

You next gradually migrate the network to the new core by migrating physical connections from the old network core to the new network core. At the same time, you delete the old IGP configuration and enable the new IGP configuration.

*Continued on next page.*

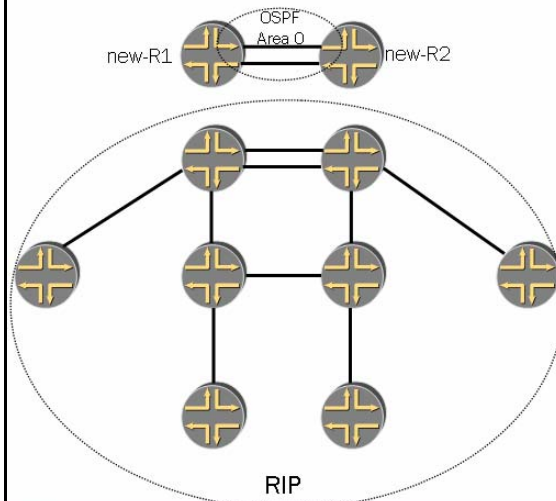
### **Step 4: Decommissioning the Old Network Core**

After you complete migrating the network to the new core, you decommission the old network core.



## Integrated Transition Example: Step 1

- Deploy a new network core running the new IGP



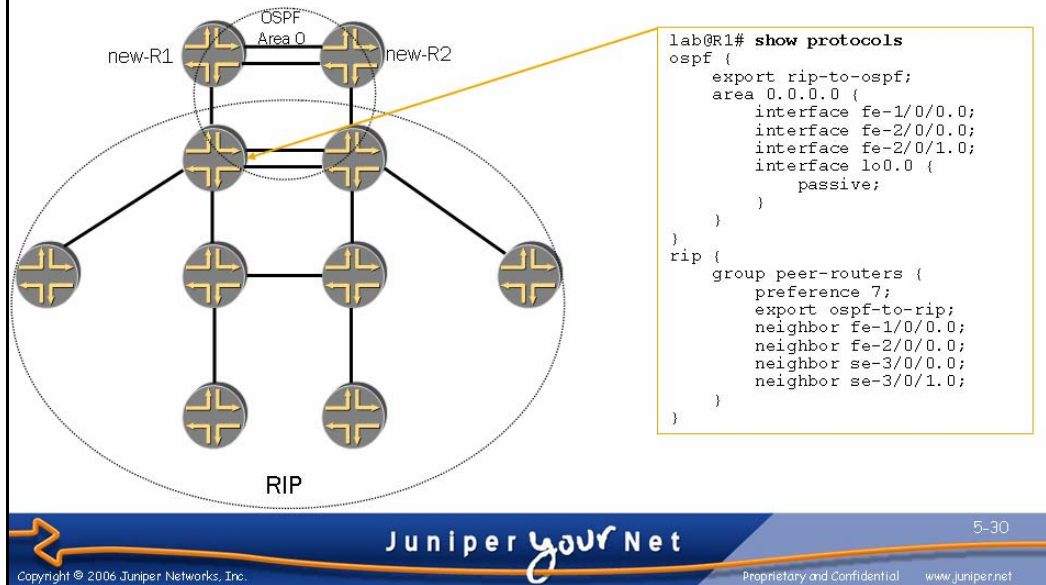
### Integrated Transition Example: Step 1

You begin the integrated transition by deploying a new network core. You configure these routers to use the new IGP.

In the example on the slide, you deploy the routers called new-R1 and new-R2. You also configure the routers to use OSPF and configure Area 0.

## Integrated Transition Example: Step 2

- Connect the old and new network cores



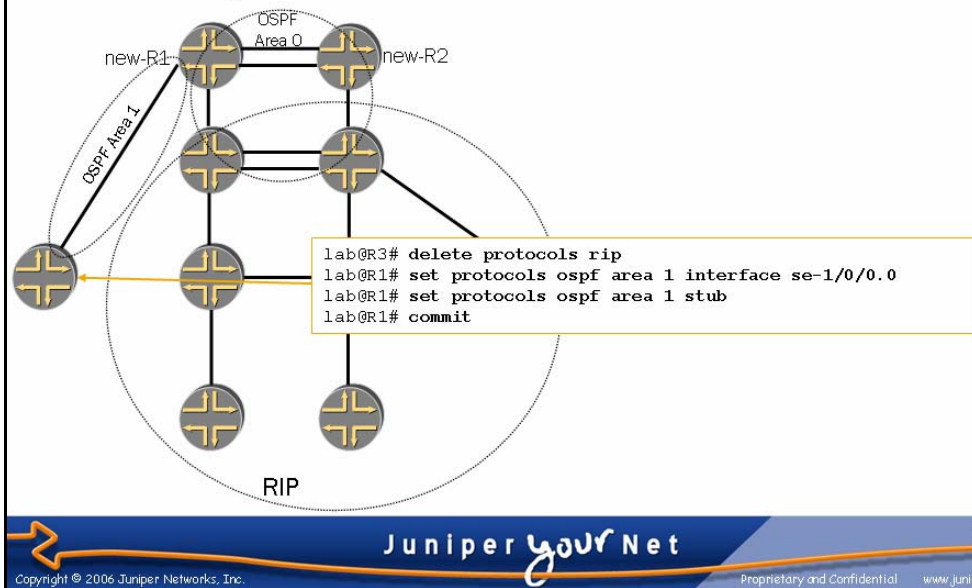
### Integrated Transition Example: Step 2

Once you deploy the new network core, you connect the old and new network cores. Also, you either configure the new network core to run the old IGP or configure the old network core to run the new IGP.

In the example on the slide, you connect new-R1 to R1 and connect new-R2 to R2. You configure the old network core to run OSPF. You also configure the old network core to export OSPF routes to RIP, and vice versa. In addition, you configure the router with appropriate policies to control the routes that it exports between protocols.

## Integrated Transition Example: Step 3

- Gradually convert routers to the new network core



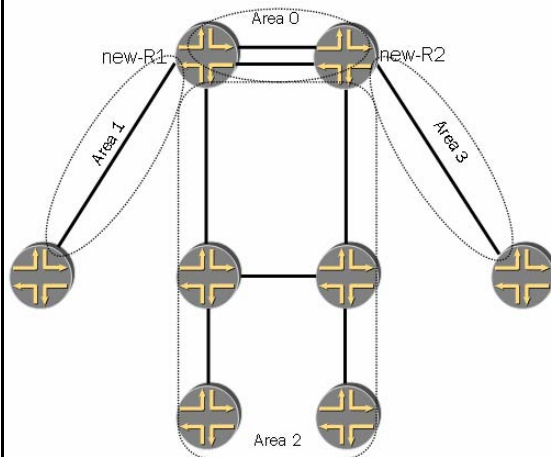
### Integrated Transition Example: Step 3

You continue the integrated transition by gradually migrating routers from the old network core to the new network core through a series of hot cuts.

In the example on the slide, you disconnect a router called R3 from R1 and reconnect it to new-R1. At the same time, you use the commands on the slide to delete the old routing protocol and enable the new routing protocol.

## Integrated Transition Example: Step 4

- Decommission the old network core



### Integrated Transition Example: Step 4

Once you migrate all the connections to the new core, you finish the integrated transition by decommissioning the old network core.

In the example on the slide, you remove R1 and R2 from the network.

## Review Questions

---

1. What are three strategies for transitioning between IGPs?
2. What are the benefits and drawbacks of each strategy?
3. What are the dangers of mutually exporting routes between IGPs?



### This Chapter Discussed:

- Three IGP transition strategies; and
- Transitioning between IGPs with minimal or no network disruption while maintaining network stability.

## Lab 3: IGP Transition

---

- Configure RIP.
- Prepare the network to transition to OSPF using the overlay model.
- Configure OSPF.
- Verify the proper operation of OSPF.
- Deactivate and delete the RIP configuration.



### Lab 3: IGP Transition

The slide highlights the objectives for this lab.



# **Advanced Juniper Networks Routing in the Enterprise**

## **Chapter 6: JUNOS Services Overview**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - List some services provided by the Adaptive Services PIC (or equivalents)
  - Configure MLPPP, MLFR, and CRTP
  - Describe service sets



### This Chapter Discusses:

- Some services provided by the Adaptive Services PIC (AS PIC) or equivalent PICs, modules, or processes;
- The configuration of Multilink PPP (MLPPP), Multilink Frame Relay (MLFR), and the compressed Real-Time Transport Protocol (CRTP); and
- Service sets.



## **Agenda: JUNOS Services Overview**

---

- JUNOS Services Overview
- Layer 2 Services Configuration
- Layer 3 Services Overview



### **JUNOS Services Overview**

The slide lists the topics we cover in this chapter. We discuss the highlighted topic first.

## **JUNOS Services Overview**

---

- **Layer 2 services:**
  - MLPPP
  - MLFR
  - CRTP
- **Layer 3 services:**
  - Stateful firewall
  - NAT
  - IPSec VPN
  - Intrusion detection



### **Layer 2 Services**

The AS PIC provides several Layer 2 services through its link services (`lsq-`) interface. You can configure MLPPP and MLFR to bind one or more interfaces together into a single logical interface. In addition, you can configure CRTP to compress the IP, UDP, and RTP headers for voice-over-IP (VoIP) traffic.

### **Layer 3 Services**

The AS PIC also provides several Layer 3 services through service sets and its adaptive services interface (`sp-`). The AS PIC can provide stateful firewall, Network Address Translation (NAT), IP Security (IPSec) virtual private network (VPN), and intrusion detection services (IDS).

## Related Cards

- Services provided by:
  - AS PIC
  - AS Module (M7i)
  - J-series software processes
  - Link Services PIC
  - Multilink Services PIC
  - Tunnel Services PIC



## Devices Providing JUNOS Services

Unless otherwise indicated, the AS PIC supports all the services described in this chapter. In addition to the AS PIC, several other devices can provide JUNOS services. The AS Module (ASM) is an optional component that you can order with the M7i router. The module provides the same services as the AS PIC, but because it is integrated, it does not occupy one of the four integrated PIC slots.

In addition, the J-series router has software processes that support the same services as the AS PIC. However, one noticeable difference is that the link services interface on the J-series router is designated `ls-`, rather than `lsq-`.

The Multilink Services PIC and Link Services PIC provide MLPPP and MLFR support. The Multilink Services PIC interfaces are designated `ml-`, while the Link Services PIC interfaces are designated `ls-`. The Multilink Services PIC supports no class-of-service (CoS) features, while the Link Services PIC supports some CoS features, and the AS PIC, ASM, and J-series router support full CoS features.

In addition to the features mentioned thus far, the AS PIC, ASM, and J-series router support some tunneling protocols (GRE and IP-in-IP, for example). The Tunnel Services PIC provides many of these same features. The Tunnel Services PIC, AS PIC, ASM, and J-series router use the same interface identifiers for these tunnel interfaces.

## AS PIC Service Package

- Must configure AS PIC for Layer 2 or Layer 3 service package under [edit chassis fpc slot pic pic adaptive-services]:  
**set service-package (layer-2 | layer-3)**



### AS PIC Service Package

To provide the most efficient processing possible, you must configure each AS PIC to provide Layer 2 or Layer 3 services, but not both. You can configure each PIC individually so that you can provide both Layer 2 and Layer 3 services in the same chassis using multiple AS PICs.

The J-series router and ASM do not have this restriction, but rather they support both Layer 2 and Layer 3 services at the same time.

## Agenda: JUNOS Services Overview

---

- JUNOS Services Overview
- Layer 2 Services Configuration
- Layer 3 Services Overview



### Layer 2 Services Configuration

The slide highlights the topic we discuss next.

## Layer 2 Services and VoIP

- Routers fragment packets on MLPPP and MLFR links
  - Reduces serialization delay
  - Can be configured with a single link
  - Queue 2 unfragmented
- CRTP reduces 40-byte IP/UDP/RTP header to 2 or 4 bytes



### Serialization Delay

Normally, a router can transmit only one packet at a time on any given link. When you configure a router to prioritize voice packets over other data packets, it transmits the voice packet as soon as it can; however, if it is currently in the middle of transmitting a different packet, it cannot begin transmitting the voice packet until it completes transmitting the other packet. This delay is called *serialization delay*. You can quantify the maximum serialization delay on a link by dividing the maximum transmission unit (MTU) by the transmission speed of the interface. For example, on a T1 with a 1,500-byte (12,000-bit) MTU, the serialization delay is:

$$12,000 \text{ bits} / 1,544,000 \text{ bits/sec.} = .008 \text{ seconds (approx.)}$$

Configuring MLPPP or MLFR causes the router to fragment packets into smaller chunks. When a high-priority voice packet arrives while the router is transmitting a different packet, the router can begin transmitting the voice packet as soon as it finishes transmitting the current fragment of the other packet (rather than waiting until it has transmitted the other packet in its entirety). This behavior is called *interleaving*. In addition, the J-series router does not fragment packets that are in Queue 2, but it transmits those packets unfragmented. Not fragmenting these packets prevents any fragmentation/reassembly delays for these packets, which is why an ideal configuration for J-series routers places all voice packets (and only voice packets) in the forwarding class associated with Queue 2.

*Continued on next page.*

## Compressed RTP

You can configure the router to use CRTP to compress the headers of RTP packets over a point-to-point link. By using CRTP, the routers on both ends of the link compress the IP, UDP, and RTP headers of RTP packets from 40 bytes to 2 or 4 bytes. CRTP is performed between two routers directly connected via a point-to-point link. Both routers must support CRTP to use it. In addition, the compression only occurs on the point-to-point link between the two routers.

## Multilink PPP

- Logically binds one or more physical links together
  - CoS at both bundle and constituent interface

```

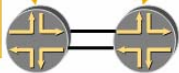
interfaces {
  se-0/0/0 {
    unit 0 {
      family mlppp {
        bundle lsq-1/2/0.1;
      }
    }
  }
  se-0/0/1 {
    unit 0 {
      family mlppp {
        bundle lsq-1/2/0.1;
      }
    }
  }
  lsq-1/2/0 {
    unit 1 {
      encapsulation multilink-ppp;
      family inet {
        address 192.168.1.1/30;
      }
    }
  }
}

```

```

interfaces {
  se-0/0/0 {
    unit 0 {
      family mlppp {
        bundle lsq-1/2/0.1;
      }
    }
  }
  se-0/0/1 {
    unit 0 {
      family mlppp {
        bundle lsq-1/2/0.1;
      }
    }
  }
  lsq-1/2/0 {
    unit 1 {
      encapsulation multilink-ppp;
      family inet {
        address 192.168.1.2/30;
      }
    }
  }
}

```



### Multilink PPP

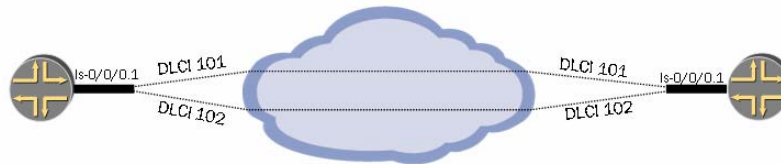
You can configure the router to bind one or more physical interfaces into a single multilink PPP bundle. To configure this, you create a logical interface for the bundle by configuring a unit with the `multilink-ppp` encapsulation on the AS PIC's `lsq-` interface, Link Services PIC's `ls-` interface, Multilink PIC's `ml-` interface, or J-series `ls-` interface. You then configure the constituent physical links for PPP encapsulation and assign them to the appropriate logical bundle interface under the `unit 0 family mlppp` hierarchy.

When you configure MLPPP, the router first sends the traffic to the PIC with the logical bundle interface for processing. This PIC performs any necessary fragmentation and determines how to distribute the traffic between the constituent links. It then sends the packets (or fragments) to the PFE to be sent to the appropriate output interfaces. The PFE sends these packets (or fragments) to the output interfaces for transmission. Because of this two-level processing, statistics are available for both the bundle interface and also the constituent links. Additionally, if you want to apply a CoS scheduler map for this bundle, you should configure a scheduler map for both the bundle interface and each constituent interfaces, and you should configure the same scheduler map at both levels.

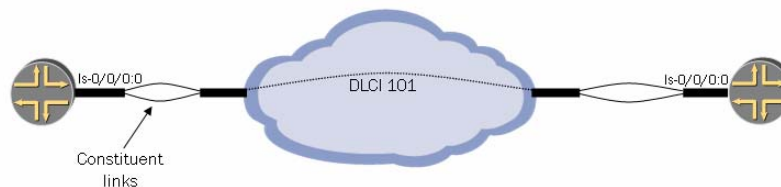


## Multilink Frame Relay

- FRF.15: end to end



- FRF.16: customer to provider



### End-to-End Multilink Frame Relay

One form of multilink Frame Relay (MLFR) is FRF.15, which allows you to configure a router to bind multiple data-link connection identifiers (DLCIs) together into a single logical interface. The DLCIs can be on the same or different physical interfaces. FRF.15 is comparable to running MLPPP over multiple DLCIs that connect two end points.

### Customer-to-Provider Multilink Frame Relay

Another form of MLFR is FRF.16, which allows you to configure a router to bind multiple physical connections to the provider into a single logical connection. This single logical connection then appears to be a *normal* Frame Relay connection, allowing you to configure multiple DLCIs over this connection. In essence, FRF.16 simply allows you to have a higher-bandwidth connection to the Frame Relay network.

## FRF.15 Configuration

- FRF.15 sample configuration:

```

interfaces {
  ls-0/0/0 {
    unit 1 {
      encapsulation multilink-frame-relay-end-to-end;
      family inet {
        address 192.168.100.1/30;
      }
    }
  }
  se-1/0/0 {
    unit 0 {
      dlci 101;
      encapsulation multilink-frame-relay-end-to-end;
      family mlfr-end-to-end {
        bundle ls-0/0/0.1;
      }
    }
  }
  se-1/0/1 {
    unit 0 {
      dlci 102;
      encapsulation multilink-frame-relay-end-to-end;
      family mlfr-end-to-end {
        bundle ls-0/0/0.1;
      }
    }
  }
}

```



### FRF.15 Sample Configuration

The sample configuration on this slide enables the FRF.15 topology shown on the previous slide. There are two physical circuits to the Frame Relay provider. Each circuit carries a single DLCI. You configure Unit 1 on the `ls-0/0/0` interface to be the logical interface that represents the FRF.15 bundle. You then configure each of the constituent logical interfaces (the units with the constituent DLCIs) to be part of the FRF.15 bundle.

## FRF.16 Configuration

### ■ FRF.16 sample configuration:

```
chassis {
  fpc 0 {
    pic 0 {
      mlfr-uni-nni-bundles 1;
    }
  }
}
interfaces {
  ls-0/0/0:0 {
    encapsulation multilink-frame-relay-uni-nni;
    unit 0 {
      dlci 101;
      family inet {
        address 192.168.100.1/30;
      }
    }
  }
  se-1/0/0 {
    encapsulation multilink-frame-relay-uni-nni;
    unit 0 {
      family mlfr-uni-nni {
        bundle ls-0/0/0:0;
      }
    }
  }
  se-1/0/1 {
    encapsulation multilink-frame-relay-uni-nni;
    unit 0 {
      family mlfr-uni-nni {
        bundle ls-0/0/0:0;
      }
    }
  }
}
```



Juniper *your* Net

6-13

Copyright © 2006 Juniper Networks, Inc.

Proprietary and Confidential www.juniper.net

### FRF.16 Sample Configuration

You begin the FRF.16 configuration by configuring the number of FRF.16 interfaces that the router should create. You do this by configuring the number of FRF.16 bundles under the [edit chassis fpc *fpc* pic *pic*] hierarchy with the **mlfr-uni-nni-bundles number** directive. The router creates enough channelized interfaces to support the specified number of FRF.16 bundles on the PIC you specify. For example, if you configure a J-series router to support five FRF.16 bundles, it creates interfaces `ls-0/0/0:0` through `ls-0/0/0:4`.

You then choose one of the channelized interfaces to be the logical interface for this FRF.16 bundle. You configure a single DLCI on this interface, although you could configure multiple DLCIs. Note that some hardware has limitations for the range of numbers that can be used to identify units. For example, the JUNOS software only allows you to configure units 0–47 on these interfaces on the J-series router platform. For that reason, you choose Unit 0 for this DLCI.

You then assign the constituent serial links to this logical bundle using the configuration shown on the slide.

## CRTP Configuration Guidelines

- Configure compression on the `lsq-` (or J-series `ls-`) interface
  - Add to an existing multilink unit
  - Add to a new unit and reference that from an existing interface (J-series platforms only)
- Configure compression based on UDP port range or queue



### CRTP Configuration

You configure CRTP for a unit of the AS PIC's `lsq-` interface or the J-series `ls-` interface. You can either add the compression to an existing multilink unit or you can configure compression on a new unit and reference that unit from an existing nonservices logical interface.

### Traffic Subject to Compression

You must configure the router to know what traffic it should try to compress. You can configure the router to choose traffic for CRTP compression based on a UDP port range or based on the queue in which the traffic is placed. If you configure both conditions, the router treats them as a logical OR and tries to compress traffic that matches either condition.

## CRTP Sample Configuration (1 of 2)

- CRTP on an existing MLPPP interface:

```
interfaces {
  lsq-1/2/0 {
    unit 1 {
      encapsulation multilink-ppp;
      compression {
        rtp {
          port minimum 2000 maximum 64009;
        }
      }
      family inet {
        address 192.168.1.1/30;
      }
    }
  }
}
```

### CRTP Sample Configuration: Part 1

On this slide, we show the configuration to add CRTP to one of the existing MLPPP interfaces created in the example on page 6-10. This configuration matches all traffic with a UDP source or destination port between 2000 and 64009, inclusive.

## CRTP Sample Configuration (2 of 2)

- CRTP on a nonmultilink interface:

```

interfaces {
  ls-0/0/0 {
    unit 9 {
      compression {
        rtp {
          port minimum 2000 maximum 64009;
        }
      }
      family inet {
        address 192.168.1.1/30;
      }
    }
  }
  se-1/0/0 {
    unit 0 {
      compression-device ls-0/0/0.9;
    }
  }
}

```



### CRTP Sample Configuration: Part 2

On the slide, we show the configuration to enable CRTP on an existing nonservices logical interface. You configure a new unit under the J-series router's `ls-` interface, and you configure that unit to perform RTP compression. You then use the **`compression-device services-interface`** configuration directive to enable CRTP on the existing nonservices logical interface. In this configuration, you must configure the Layer 3 configuration on the services interface.

## Agenda: JUNOS Services Overview

---

- JUNOS Services Overview
- Layer 2 Services Configuration
- Layer 3 Services Overview



### Layer 3 Services Overview

The slide highlights the topic we discuss next.

## Service Sets

- Service sets define the parameters for Layer 3 services:
  - Service rules or rule sets (IDS, NAT, stateful firewall, VPN)
  - Type of service set (interface, next hop)
  - Service interfaces



### Service Sets

A service set defines the way the router applies Layer 3 services to a packet. You configure the way packets are selected for processing by defining this service set as either an interface-style or next-hop-style service set and associating services interfaces with it. Once the router selects a packet to be processed through a particular service set, the router applies the Layer 3 service rules configured in that service set.

The IDS, NAT, and stateful firewall functions are performed on a per-connection (stateful) basis, while the IPSec VPN function is performed on a per-packet (stateless) basis. You can configure multiple kinds of rules within a service set except for IPSec VPN rules, which cannot be mixed with any other kinds of rules.



## Service Rules

- Service rules define the action to be taken on a particular data flow
  - Define direction (inbound, outbound, or both)
  - Contain terms
- Rule sets group rules together, similar to a policy chain



### Service Rules

You configure the router to apply Layer 3 services to connections through rules. You can configure rules for each kind of Layer 3 service, and you configure rules for each kind of service separately.

Similar to stateless firewall filters, service rules have terms that are processed sequentially. Once a connection matches a term, that action is taken and processing stops. Unlike stateless firewall filters, service rules are processed on a stateful basis. Once the connection matches a rule, the connection is added to the state table along with the actions defined in the rule. Future packets within the connection are only processed against the entry in the state table.

### Rule Sets

You can define a rule set for each Layer 3 service that is an ordered list of rules that should be applied to each connection. A rule set can only contain rules for one kind of service. The rule set behaves similarly to a policy chain.

You can achieve the same effect by simply listing multiple rules within a service set.

## Service Rule Terms

- Analogous to stateless firewall terms, except processed once per session in a stateful manner
  - Contain *from* and *then* clauses
- Match and action conditions vary for each of the Layer 3 services



### Service Rule Terms

Service rule terms are similar to stateless firewall terms. You configure them with names, the router processes them sequentially within a rule, and they contain *from* clauses to define match conditions and *then* clauses to define actions. However, unlike stateless firewalls, these terms are processed statefully. Therefore, the match conditions do not need to account for fragments or allow return traffic.

### Match and Action Conditions

Different match and action conditions are available for each Layer 3 service. For example, you can use IPSec rules to match on source and destination addresses and configure VPN options as actions, while you can also use applications as match criteria for stateful firewall rules and you can configure actions in stateful firewall rules that define whether a packet should be accepted.

## Next-Hop Style Versus Interface Style

- **Interface style:**
  - Configured at the interface level
  - Applied as traffic enters and leaves an interface
  - Tracks sessions per service set (rather than per interface)
- **Next-hop style:**
  - Uses two logical services interfaces: one *inside*, one *outside*
  - Applied as traffic is routed to the inside or outside interface
  - Forwards traffic from the inside interface to the outside interface, and *vice versa*



### Interface-Style Service Sets

You apply interface-style service sets to particular interfaces, and the services defined by the service sets are applied as traffic enters or leaves a particular interface. The AS PIC tracks sessions on a per-service-set basis. Because it tracks sessions on a per-service-set basis, you can cause the router to process asymmetric traffic flows correctly by applying the same service set to multiple interfaces.

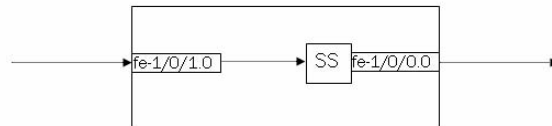
### Next-Hop-Style Service Sets

When you configure next-hop-style service sets, you associate them with specific inside and outside logical interfaces. These logical interfaces are units you configure on an AS PIC's `sp-` interface. You configure the router to route traffic to the inside or outside interface. When the AS PIC receives traffic on the inside interface associated with a service set, the AS PIC applies the configured Layer 3 services and then forwards the packet back to the router through the outside interface. Likewise, when the AS PIC receives traffic on the outside interface associated with a service set, it forwards the packet back to the router through the inside interface after applying the configured Layer 3 services.

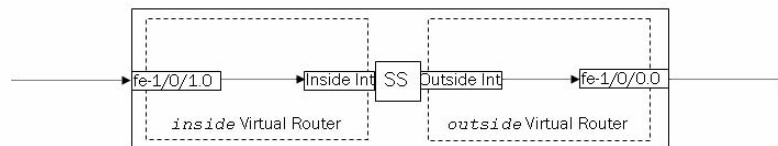
We recommend using the next-hop-style service set method of applying Layer 3 services.

## Next-Hop and Interface Styles Illustrated

- Interface style:



- Next-hop style:



### Service Set Processing Illustrated

The slide shows the way that the router processes packets through service sets (marked by “SS” on the diagram). In both examples, the router receives a packet on the `fe-1/0/1.0` interface and transmits it on the `fe-1/0/0.0` interface.

The interface-style service set is applied to the `fe-1/0/0.0` interface. As the router prepares to transmit the packet on this interface, the packet is sent to the AS PIC for the AS PIC to process the packet through the configured service set. Once the AS PIC transmits the packet back to the FPC, the router transmits the packet on `fe-1/0/0.0` without conducting another forwarding table lookup.

On the slide, the next-hop-style service set is implemented with virtual routers. It is not always necessary to use virtual routers with next-hop-style service sets (particularly with IPSec VPNs); however, it often is necessary (or at least highly preferable) to use virtual routers to implement next-hop-style service sets to avoid a routing loop within the router.

*Continued on next page.*

## Service Set Processing Illustrated (contd.)

In the example on the slide, the router receives a packet on the `fe-1/0/1.0` interface. The Packet Forwarding Engine (PFE) conducts a forwarding table lookup using the `inside.inet.0` table, which returns a next hop of the inside interface of the service set. The FPC transmits the packet to the AS PIC, which processes the packet through the Layer 3 service rules configured in the service set and transmits the packet back to the FPC, using the service set's outside interface as the input interface. Because the service set's outside interface is assigned to the *outside* virtual router, the PFE conducts a forwarding table lookup using the `outside.inet.0` table, which returns a next-hop interface of `fe-1/0/0.0`. The router then transmits the packet out this interface.

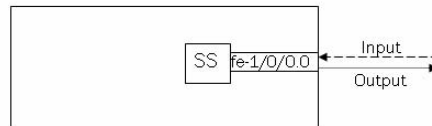
## Virtual Routers and Service Sets

As the example shows, next-hop-style service sets use two normal route lookups. The first route lookup directs traffic to an AS PIC *sp*- logical interface. After the AS PIC processes the packet through the Layer 3 services, the PFE conducts another route lookup to determine where to send the packet. It is necessary that the first route lookup return a service interface, while the second route lookup return a different next hop. However, if the AS PIC does not change the destination address during Layer 3 service processing and both route lookups are conducted using the same forwarding table, the router will continually send the packet to the same place. If it continually sends the packet to the AS PIC's *sp*- logical interface, you end up with a *services loop*, which is essentially a routing loop within the router itself. A services loop prevents traffic from being forwarded correctly, consumes bandwidth between the FPC and AS PIC, and consumes AS PIC resources.

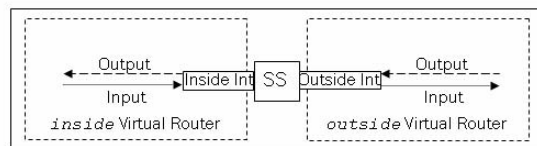
To prevent a services loop, you must ensure that the preservices and post-services route lookups return different next hops. One of the easiest and most obvious ways to ensure different next hops is to place at least one of the service set logical interfaces in a different virtual router. Using virtual routers ensures that the two route lookups return different next hops, and using virtual routers might also have other security advantages. We discuss virtual routers in more detail in Chapter 7, "Stateful Firewall and Network Address Translation".

## A Word About Directions

- Interface style: What you expect



- Next-hop style: Backwards



### Directions

When you configure service rules, you configure the router to apply each rule to traffic that is headed in a specific direction. You can configure the router to apply the rule to input traffic, output traffic, or both input and output traffic.

### Interface Style Versus Next-Hop Style

When the AS PIC processes an interface-style service set, it considers traffic received on the interface where the service set is applied to be input traffic. Likewise, it considers traffic that is about to be transmitted on the interface to be output traffic.

When the AS PIC processes a next-hop-style service set, it considers traffic direction from the perspective of the AS PIC's inside interface. Therefore, it considers traffic received on the outside interface to be output traffic and it considers traffic received on the inside interface to be input traffic. In essence, this process is backwards from what you would expect to be the case.

One of the practical implications of this difference is that you must be careful when trying to use service rules in both interface-style and next-hop-style service sets. In many cases, the direction will be incorrect, and you will find that you must create different rules for use with interface-style and next-hop-style service sets.

## Services Interfaces

### ■ Interface-style: Configure `family inet` on unit 0

```
[edit]
lab0R1# set interfaces sp-0/0/0 unit 0 family inet

[edit]
lab0R1# show interfaces sp-0/0/0
unit 0 {
    family inet;
}
```

### ■ Next-hop style: Create inside and outside units with `family inet`

```
[edit]
lab0R1# set interfaces sp-0/0/0 unit 1 service-domain outside family inet

[edit]
lab0R1# set interfaces sp-0/0/0 unit 2 service-domain inside family inet

[edit]
lab0R1# show interfaces sp-0/0/0
unit 0 {
    family inet;
}
unit 1 {
    family inet;
    service-domain outside;
}
unit 2 {
    family inet;
    service-domain inside;
}
```

## Services Interfaces

The AS PIC uses the `sp- Unit 0` logical interface for many purposes. You must never disable the Unit 0 logical interface. In most cases, the AS PIC sends traffic it generates through this logical interface.

When you configure service sets, you must associate them with units on the AS PIC's `sp-` interface. You can use Unit 0 for interface-style service sets that are not in a virtual router. In addition, you can create extra units for use with other service sets.

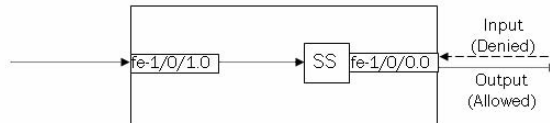
## Configuring Services Interfaces for Service Sets

To use a services interface to process service sets, you must configure the interface to support `family inet`. In addition, for next-hop-style service sets, you must configure the interface as an inside or outside interface.

## Sample Interface-Style Service Set

- Sample interface-style service set that allows all outbound traffic:

```
lab@R1# show services
stateful-firewall {
  rule allow-all-outbound {
    match-direction output;
    term allow-all {
      then {
        accept;
      }
    }
  }
}
service-set SFFW-OUTBOUND {
  stateful-firewall-rules allow-all-outbound;
  interface-service {
    service-interface sp-0/0/0;
  }
}
```



### Interface-Style Service Set Example

The slide shows a very simple interface-style service set example. You configure the service set with the name *SFFW-OUTBOUND*. You configure the service set to use one Layer 3 service rule: a stateful firewall rule called *allow-all-outbound*. You also configure the service set to use *sp-0/0/0* logical services interface to process the traffic. (You must assign this logical services interface to the same virtual router as the interfaces to which you apply this service set.)

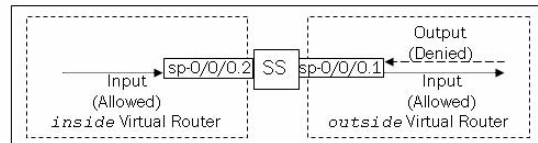
Because this is an interface-style service set, you configure the *allow-all-outbound* stateful firewall rule to accept all traffic in the output direction. By default, if you do not configure a service set to use any stateful firewall rules, the AS PIC allows all connections. However, if you configure a service set to use at least one stateful firewall rule, the AS PIC discards all connections that fail to match one of the rules you configure in the service set.



## Sample Next-Hop-Style Service Set

- Sample next-hop-style service set that allows all outbound traffic:

```
lab@R1# show services
stateful-firewall {
  rule allow-all-outbound {
    match-direction input;
    term allow-all {
      then {
        accept;
      }
    }
  }
  rule deny-all-inbound {
    match-direction output;
    term deny-all {
      then {
        discard;
      }
    }
  }
  rule-set allow-outbound-only {
    rule allow-all-outbound;
    rule deny-all-inbound;
  }
}
service-set NH-SFW-OUTBOUND {
  stateful-firewall-rule-sets allow-outbound-only;
  next-hop-service {
    inside-service-interface sp-0/0/0.2;
    outside-service-interface sp-0/0/0.1;
  }
}
```



### Next-Hop-Style Service Set Example

The slide shows a very simple next-hop-style service set. You configure the service set with the name *NH-SFW-OUTBOUND*. You configure the service set to use one Layer 3 service rule-set: a stateful firewall rule-set called *allow-outbound-only*. You also configure the service set to be a next-hop-style service set and use the *sp-0/0/0.1* logical services interface as the outside interface and the *sp-0/0/0.2* logical services interface as the inside interface. (You must assign these logical services interfaces to the appropriate virtual routers and configure appropriate routes to direct traffic to these interfaces.)

You configure a stateful firewall rule-set called *allow-outbound-only*, which you configure to include the *allow-all-outbound* and *deny-all-inbound* stateful firewall rules. Because this is a next-hop-style service set, you configure the *allow-all-outbound* stateful firewall rule to accept all traffic in the *input* direction. You also configure a stateful firewall rule called *deny-all-inbound* to explicitly block all traffic in the *output* direction. As indicated on page 6-24, you reverse the direction (input vs. output) when configuring rules for next-hop-style service sets rather than interface-style service sets.

## Applying Service Sets

### ■ Interface style: Configure on logical interface

- Must configure same service sets for input and output
- Can configure service filter
- Can configure one post-service filter on input
- Example:

```
lab@R1# show interfaces fe-0/0/0
unit 0 {
  family inet {
    service {
      input {
        service-set SFW-OUTBOUND;
      }
      output {
        service-set SFW-OUTBOUND;
      }
    }
    address 10.210.9.177/28;
  }
}
```

### ■ Next-hop style: Route traffic to inside or outside interface

- Example:

```
lab@R1# show routing-options
static {
  route 192.168.201.0/24 next-hop sp-0/0/0.1;
}
```



## Applying Interface-Style Service Sets

To apply interface style service sets to an interface, you configure them under the [edit interface *interface* unit *unit* family inet service] hierarchy. You must configure input and output service sets separately; however, you must configure the same service sets to be applied to both input and output traffic. By default, all unicast traffic is processed through the configured service set. You can configure service filters to select which traffic should be processed through a service set. We discuss service filters on page 6-30.

If you configure service filters, you can list multiple service sets. As the router processes a packet, it compares it to the service filters associated with each service set. The router processes the packet using the service set associated with the first service filter that selects the packet for processing. If no service filter selects the packet for processing, the packet is not processed through any service set or post-service filter, if one is configured.

You can configure one post-service filter to be applied in the input direction. You can configure the router to use any stateless firewall filter as a post-service filter. If you configure a post-service filter, the PFE processes packets that are processed by a non-VPN service set through the firewall filter. The PFE processes packets through the post-service filter after the AS PIC has processed the configured service set. Therefore, you must configure the post-service filter to use post-NAT addresses and ports.

*Continued on next page.*

## Applying Next-Hop-Style Service Sets

By comparison, applying next-hop-style service sets is relatively easy. You simply configure appropriate routes so the router forwards packets to the inside or outside interface of the service set, as appropriate. The slide shows an example of configuring the main router to send traffic destined for the 192.168.201.0/24 prefix to the outside interface of the *NH-SFW-OUTBOUND* service set (shown on page 6-27). Although it is not shown, you should also assign the *sp-0/0/0.2* (inside) logical interface to another virtual router that also contains routes to the hosts in the 192.168.201.0/24 prefix.

## Service Filters

- Select packets to be sent through a particular interface-style service set

- Configured under `[edit firewall family inet]`
- Applied to a service set on input, output, or both
- Never matches multicast traffic

• Example:

```
lab@R1# show firewall family inet
service-filter no-gre {
  term ignore-gre {
    from {
      protocol gre;
    }
    then skip;
  }
  term all-other-traffic {
    then service;
  }
}
[edit]
lab@R1# show interfaces fe-0/0/0 unit 0 family inet service
input {
  service-set SFW-OUTBOUND service-filter no-gre;
}
output {
  service-set SFW-OUTBOUND service-filter no-gre;
}
```



### Service Filters

You use service filters to tell the router which traffic should be processed through each interface-style service set on an interface. If you do not configure a service filter, the router uses a default service filter that matches all unicast traffic.

Service filters are stateless filters that you configure under the `[edit firewall family inet]` hierarchy. When configuring service filters, you can configure the same match options that are available for stateless firewall filters; however, the only terminating actions you can configure are the `skip` and `service` actions. If the router finds that the packet matches a term with the `skip` action, the router does not process the packet through the associated service set but rather begins to evaluate the packet against the service filter associated with the next service set configured on the interface. If the router finds that the packet matches a term with the `service` action, the router processes the packet through the associated service set (and only that service set) and does not evaluate any further service filters. If the router finds that the packet does not match a term with either the `skip` or `service` action, it takes the default action of `skip`.

You cannot configure service filters to match multicast traffic. Therefore, you cannot use interface-style service sets to process multicast traffic.

In the example on the slide, the router processes all packets inbound and outbound on the `fe-0/0/0.0` interface through the `no-gre` service filter. If the packet is a unicast packet with IP protocol 47 (GRE's protocol), the router does not process the packet through the service set. If the packet is a unicast packet with any other IP protocol, the router processes the packet through the `SFW-OUTBOUND` service set.

## Post-Service Filter

- A single stateless firewall filter applied to packets after the service set is applied
  - Only available on input
  - Only applied to packets that pass a non-VPN service set
  - Only applied to packets that are processed by a service set
  - Allows filter-based forwarding
  - Example:

```
lab@R1# show interfaces fe-0/0/0 unit 0 family inet service
input {
  service-set SFW-OUTBOUND service-filter no-gre;
  post-service-filter example-filter;
}
output {
  service-set SFW-OUTBOUND service-filter no-gre;
}
```



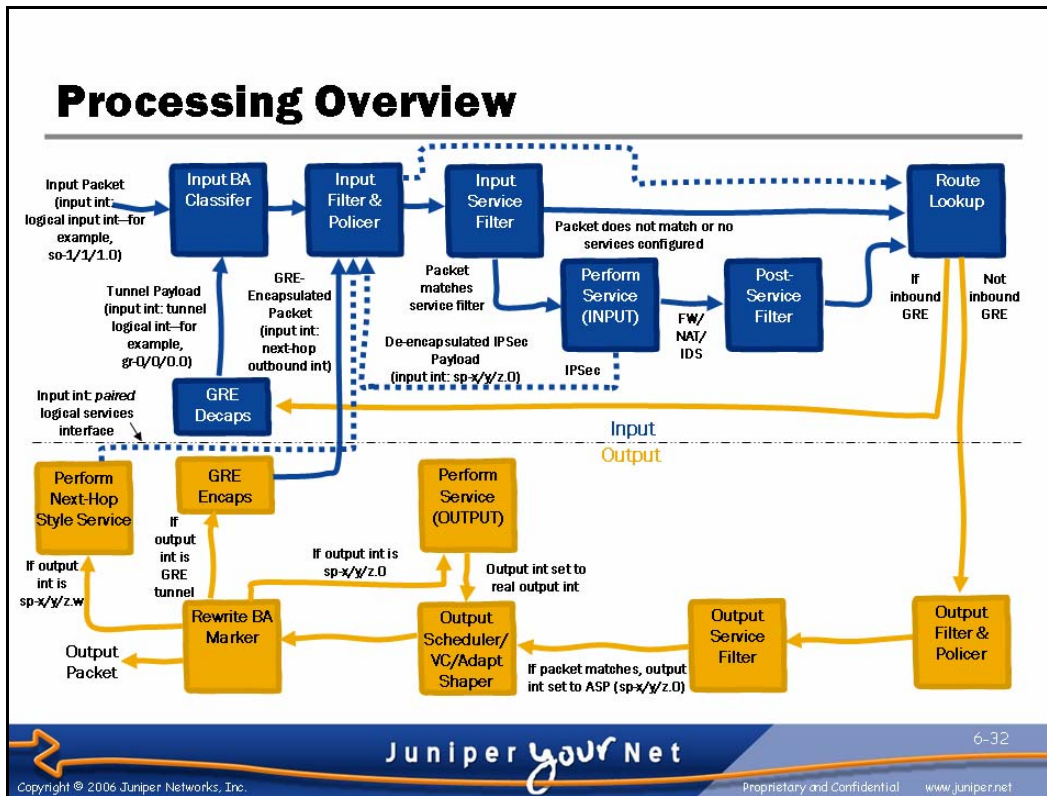
### Post-Service Filters

You can configure the router to process packets through a stateless firewall filter after the AS PIC has processed them through an interface-style service set. You can only configure the router to do this processing for packets in the input direction. The router only processes packets through the post-service filter if the AS PIC processed them through a non-IPSec VPN service set. Additionally, if a packet is skipped by all service filters (and, therefore, is not processed through a service set), the router does not evaluate it against the post-service filter.

Because the PFE evaluates packets against post-service filters after the AS PIC has processed the packets through the Layer 3 service sets, you must configure the post-service filter to match post-NAT addresses and ports.

A common use of post-service filters is to enable filter-based forwarding. If you want to use filter-based forwarding and interface-style service sets on the same interface, you must perform the filter-based forwarding in a post-service filter, rather than in a normal input stateless firewall filter.

In the example on the slide, the router evaluates input IP packets against the *no-gre* service filter. It forwards packets skipped by the *no-gre* service filter without sending the packet to the AS PIC for it to apply Layer 3 services and without evaluating the packet against the post-service filter. It sends packets that are marked for servicing by the *no-gre* service filter to the AS PIC for it to apply the Layer 3 services specified in the *SFW-OUTBOUND* service set. After the AS PIC processes the packets and transmits them back to the FPC, the PFE evaluates the packets against the *example-filter* stateless firewall filter.



### Packet Processing Overview

The slide shows a flowchart that represents the way JUNOS routers process packets. This diagram shows the interactions between many different functions that the router performs. The goal of the diagram is to help you understand the way that each configuration option applies to a given packet and to understand the way that different configuration options interact with each other.

You do not need to fully understand this slide at this point. We return to this diagram through the remaining chapters as we explain various interactions. Additionally, Appendix A covers several detailed examples.

## Review Questions

1. What Layer 2 services can you use to enhance the processing of VoIP traffic?
2. How do you configure an AS PIC to support Layer 2 or Layer 3 services?
3. What are the differences between interface-style and next-hop-style service sets?
4. What is a service filter?
5. What is a post-service filter?



### This Chapter Discussed:

- Some services provided by the AS PIC or equivalent PICs, modules, or processes;
- The configuration of MLPPP, MLFR, and CRTP; and
- Service sets.



## Lab 4: Layer 2 Services

---

- Configure MLPPP.
- Configure FRF.15.
- Configure FRF.16.
- Configure CRTP.



### Lab 4: Layer 2 Services

The slide lists the objectives for this lab.





# **Advanced Juniper Networks Routing in the Enterprise**

## **Chapter 7: Stateful Firewall and Network Address Translation**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - Configure a stateful firewall policy using the JUNOS CLI
  - Configure a NAT policy using the JUNOS CLI
  - Monitor stateful firewall and NAT operation using the JUNOS CLI



### This Chapter Discusses:

- Configuring a stateful firewall policy using the JUNOS CLI;
- Configuring a Network Address Translation (NAT) policy using the JUNOS CLI; and
- Monitoring stateful firewall and NAT operation using the JUNOS CLI.

## **Agenda: Stateful Firewall and NAT**

---

- Stateful Firewall and NAT Overview
- Applications
- Configuring Stateful Firewall Rules
- Configuring NAT Rules
- Implementing Stateful Firewall and NAT
- Monitoring Stateful Firewall and NAT



### **Stateful Firewall and NAT Overview**

The slide lists the topics we cover in this chapter. We discuss the highlighted topic first.

## Stateful Firewall and NAT Overview

- AS PIC makes decisions based on connections
  - One rule enables bidirectional communication
  - AS PIC tracks fragments
  - AS PIC tracks session state
  - JUNOS software supports applications through ALGs
- Stateful firewall and NAT can be configured independently of each other



### Stateful Firewall and Network Address Translation Overview

The Adaptive Services (AS) PIC supports stateful firewall and Network Address Translation (NAT). The AS module (ASM) and J-series router provide these same services. Because you configure stateful firewall and NAT on these devices in the same way, we refer to all three of these devices as *the AS PIC* in this chapter, except where otherwise noted.

With stateful firewall rules, you create rules that match connections (rather than packets) in a particular direction. Once the AS PIC sees a matching connection, it creates an entry in the session table and applies the rule to both directions of traffic in the connection. In addition, because the AS PIC tracks connections, the AS PIC automatically accounts for fragments and tracks session state.

For some applications, the AS PIC can also inspect the actual packet payload and make firewall and NAT decisions based on the contents of the payload. This support is commonly known as an application-level gateway (ALG).

### Stateful Firewall and NAT Interaction

You do not have to configure stateful firewall rules to configure NAT, and *vice versa*. If you configure a service set to perform both NAT and stateful firewall together, the AS PIC will automatically make appropriate NAT decisions based on ALGs you configure in the stateful firewall rules. However, if you configure a service set to perform only NAT, you must configure any necessary ALGs in the NAT rules.

## Stateful Firewall and NAT Actions

- Firewall:
  - Accept
  - Reject
  - Discard
  - Modifiers: syslog, allow specific IP options
- NAT:
  - NAT/PAT
  - Static/dynamic
  - Source/destination



### Stateful Firewall Actions

Within stateful firewall rules, you can configure the AS PIC to accept, reject, or discard connections. The accept and discard actions are equivalent to their stateless firewall filter counterparts. The reject action causes the AS PIC to send ICMP error messages for UDP connections and TCP RST messages for TCP connections.

If you do not configure a service set to use any stateful firewall rules, the AS PIC allows all connections. If you do configure a service set to use at least one stateful firewall rule, the AS PIC discards all connections that do not match the configured stateful firewall rules.

In addition to the terminating actions of accept, reject, and discard, you can also configure action modifiers of allow-ip-options and syslog. Unlike stateless firewall filters, you must configure a terminating action in each term of a stateful firewall rule.

### NAT

You can configure the AS PIC to perform static or dynamic NAT or Port Address Translation (PAT). The AS PIC can perform static NAT or PAT for both source and destination addresses and dynamic NAT or PAT for source addresses; however, you cannot configure the AS PIC to perform both source and destination translation for the same connection. You can also explicitly configure the AS PIC to perform no translation for given connections.

*Continued on next page.*

## **NAT (contd.)**

If you do not configure a service set to use any NAT rules, the AS PIC does not perform any translation. If you do configure a service set to use at least one NAT rule, the AS PIC does not perform any translation for connections that do not match the NAT rules you configure.

## **Agenda: Stateful Firewall and NAT**

---

- Stateful Firewall and NAT Overview
- Applications
- Configuring Stateful Firewall Rules
- Configuring NAT Rules
- Implementing Stateful Firewall and NAT
- Monitoring Stateful Firewall and NAT



### **Applications**

The slide highlights the topic we discuss next.

## Applications

- Define connection properties

```
lab@R1# set applications application example-application ?
Possible completions:
  application-protocol  Application protocol type
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
  destination-port      Match TCP/UDP destination port
  icmp-code             Match ICMP message code
  icmp-type            Match ICMP message type
  inactivity-timeout    Application-specific inactivity timeout (4..86400 seconds)
  learn-sip-register    Learn potential incoming SIP calls by inspecting the SIP register method
  protocol             Match IP protocol type
  rpc-program-number    Match range of RPC program numbers
  sip-call-hold-timeout SIP flow timeout when call is put on hold (0..36000 seconds)
  snmp-command         Match SNMP command
  source-port          Match TCP/UDP source port
  ttl-threshold        Traceroute TTL threshold
  uuid                Match universal unique identifier for DCE RPC objects
```

- Used as match conditions in IDS, stateful firewall, and NAT rules



### Application Definition

You use applications to define connection properties for a given protocol. When you define an application, you can define many different properties, which are listed on the slide.

### Application Uses

In stateless firewall filters, you match protocols by configuring match conditions such as `protocol`, `source-port`, and `destination-port` directly in a *from* clause. You cannot configure these same match conditions directly in stateful firewall, NAT, and IDS rules. Rather, you must configure applications or application sets as match conditions.



## Application-Level Gateways

- Automatically takes action based on Layers 4–7 information
  - Performs translation on addresses and ports in payload
  - Updates session table to allow extra connections



### Application-Level Gateways

Some protocols (for example, the Session Initiation Protocol [SIP] and FTP) include some combination of IP addresses and TCP or UDP ports in their payload. If a router is configured to perform NAT and translates only the Layer 3 and Layer 4 headers, the IP addresses, the TCP or UDP ports, or both included in the payload by these protocols will be wrong. This scenario can prevent these applications from functioning correctly.

Additionally, some protocols have control connections that begin other sessions. Because these sessions are created dynamically and often use random port numbers, the firewall rules will likely not allow these sessions.

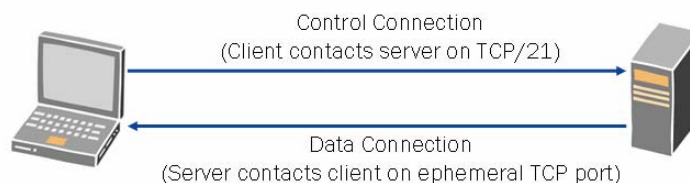
Application-level gateways (ALGs) allow the router to interact with protocols at Layer 4 and above. When you configure the router to use ALGs, it inspects the payload of connections, translating IP addresses and ports in the payload and updating the session table to allow sessions started by the control connection.

The mere fact that the stateful firewall or NAT rules accept a connection does not ensure that the AS PIC will use an ALG to process the connection. Rather, the connection must match a term that specifies the application as a match condition for the AS PIC to use the associated ALG to process the connection.

## ALG Example

### ■ Active FTP

- Client contacts server on TCP/21
- Client listens for data connection on ephemeral port
- Client sends server PORT command with IP address and TCP port
- Server opens data connection to IP/port in PORT command



### ALG Example: Active FTP

In active-mode FTP, the client contacts the server on TCP port 21. This connection serves as the control channel for the FTP session.

When it is time to exchange data, the client begins listening for a data channel connection on an ephemeral port (a random port chosen from the 1024–65535 range). Once the client begins listening on this ephemeral port, it sends the server a PORT command to tell it what IP address and TCP port to contact. The server then initiates a TCP connection to that IP address and port. Once the data transfer is complete, the data channel is closed. If another data transfer is necessary, the process is repeated.

In this example, if a router exists between the client and the server, and if you configure that router to protect the client with a stateful firewall, the router must allow the data channel connection to the client. Additionally, if you configure that router to translate the client's source address, the router must also translate the IP address and port in the PORT command. The FTP ALG included in the JUNOS software performs these functions.

## Predefined Applications

- JUNOS software comes with many predefined applications

- Configured in `junos-defaults` group
- Named `junos-*`

```
lab@R1# show groups junos-defaults applications
#
# File Transfer Protocol
#
application junos-ftp {
    application-protocol ftp;
    protocol tcp;
    destination-port 21;
}
#
# Trivial File Transfer Protocol
#
application junos-tftp {
    application-protocol tftp;
    protocol udp;
    destination-port 69;
}
```



### Predefined Applications

The JUNOS software contains many predefined applications. These applications are found in the configuration group `junos-defaults`. The configuration commands in the `junos-defaults` group do not appear in the configuration, but they are always active unless you explicitly override them. You can view the applications defined in the `junos-defaults` group with the configuration-mode CLI command **`show groups junos-defaults applications`**.

All the applications defined in the `junos-defaults` group have names that begin with `junos-`. If you choose to define custom applications, you can avoid name conflicts with these predefined applications by using names for your custom applications that do not begin with `junos-`.

## Predefined Application Sets

- JUNOS software comes with predefined application sets
  - Configured in `junos-defaults` group
  - Updated as ALGs are added
  - `junos-algs-outbound`: Can be used in a rule allowing a trusted network to initiate any connections to an untrusted network
  - `junos-management-inbound`: Should *not* be used—lists all ALGs that might be used for router management
  - `junos-routing-inbound`: Should *not* be used—lists all ALGs that might be used to support routing protocols



### Predefined Application Sets

In addition to predefined applications, the JUNOS software also has three predefined application sets. These application sets are found in the `junos-defaults` group and are updated whenever the list of predefined applications is updated.

The `junos-management-inbound` and `junos-routing-inbound` application sets are not intended to be used as match conditions in stateful firewall rules. Rather, their primary purpose is to provide a way for graphical user interfaces (GUIs) to retrieve a list of applications that are likely to be used to support network management for the `junos-management-inbound` application set or to allow routing protocol traffic for the `junos-routing-inbound` application set. Therefore, the primary function of these application sets is simply informational: to point you to applications that are likely to be used for these purposes.

The `junos-algs-outbound` application set is intended to provide an easy way to apply all predefined ALGs to a given rule. As we mentioned on page 7-9, the AS PIC does not use an ALG for a connection unless the connection matches a term that specifies the application as a match condition. Thus, if you configure a term that accepts all connections without specifying any applications as a match condition, the AS PIC accepts all connections but does not use any ALGs. This scenario causes many protocols (such as SIP and FTP) to function incorrectly. Therefore, if you are wanting to permit all connections (including all ALGs), you must configure two terms: one that accepts all connections that match the `junos-algs-outbound` application set and one that accepts all other connections.

## Defining Applications

- Configure under the [edit applications] hierarchy

```
[edit applications]
lab@R1# set application example-application ?
Possible completions:
  application-protocol  Application protocol type
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
  destination-port      Match TCP/UDP destination port
  icmp-code             Match ICMP message code
  icmp-type            Match ICMP message type
  inactivity-timeout    Application-specific inactivity timeout (4..86400 seconds)
  learn-sip-register    Learn potential incoming SIP calls by inspecting the SIP register method
  protocol             Match IP protocol type
  rpc-program-number    Match range of RPC program numbers
  sip-call-hold-timeout SIP flow timeout when call is put on hold (0..36000 seconds)
  snmp-command         Match SNMP command
  source-port          Match TCP/UDP source port
  ttl-threshold        Traceroute TTL threshold
  uuid                Match universal unique identifier for DCE RPC objects
```

- Use the application-protocol keyword for ALGs



### Defining Custom Applications

If you want to configure a custom application, you can do so under the [edit applications] hierarchy using the **set application application** command. The options to this command are shown on the slide and documented in the *JUNOS Services Interfaces Configuration Guide*.

### Configuring ALGs

You can use the application-protocol keyword to configure the router to use an ALG for a custom application. For example, if you want to configure the router to support an FTP server that accepts connections on a nonstandard port, you can include the application-protocol ftp; statement in the application definition. Including this statement causes the router to use the FTP ALG for connections matching this application.

## Sample Application Definition

### ■ Example: RDP (tcp/3389)

```
[edit applications]
lab0R1# set application rdp protocol tcp

[edit applications]
lab0R1# set application rdp destination-port 3389

[edit applications]
lab0R1# show
application rdp (
  protocol tcp;
  destination-port 3389;
)
```



### Sample Application Definition

The slide shows an example of defining a custom application. In the remote desktop protocol (RDP), clients initiate TCP connections to servers using destination port 3389. On the slide, you configure the custom application *rdp*, and tell the router that TCP connections that are initiated to destination port 3389 match this application.

## Defining Application Sets

- Configure under the `[edit applications]` hierarchy

- Example:

```
[edit applications]
lab@R1# set application-set rdp-ssh-telnet application rdp

[edit applications]
lab@R1# set application-set rdp-ssh-telnet application junos-ssh

[edit applications]
lab@R1# set application-set rdp-ssh-telnet application junos-telnet

[edit applications]
lab@R1# show
application rdp {
    protocol tcp;
    destination-port 3389;
}
application-set rdp-ssh-telnet {
    application rdp;
    application junos-ssh;
    application junos-telnet;
}
```



### Application Sets

You use application sets to group multiple applications together. You can use an application set as a match condition in a term in a stateful firewall, NAT, or IDS rule when you want the term to match any of the applications in the application set.

You associate applications with application sets using the configuration command **set application-set application-set application application** under the `[edit applications]` hierarchy.

## **Agenda: Stateful Firewall and NAT**

---

- Stateful Firewall and NAT Overview
- Applications
- Configuring Stateful Firewall Rules
- Configuring NAT Rules
- Implementing Stateful Firewall and NAT
- Monitoring Stateful Firewall and NAT



### **Configuring Stateful Firewall Rules**

The slide highlights the topic we discuss next.



## Stateful Firewall Rules

- Configure in the `[edit services stateful-firewall]` hierarchy
  - Very similar to stateless firewall filter configuration
  - Supports match conditions: source address, destination address, and application
  - Supports actions: accept, reject, and discard
  - Supports action modifiers: **syslog** and **allow-ip-options** commands
- Reference in service sets



### Configuring Stateful Firewall Rules

You configure stateful firewall rules in the `[edit services stateful-firewall]` hierarchy, using the **set rule rule** command. You must configure a match direction and terms. You can configure a *from* clause for each term that describes the match conditions for the term. If there is no *from* clause, all connections match the term (but the AS PIC does not use any ALGs). In addition, you must configure a *then* clause for each term. The *then* clause must include one of the actions and can also include one or more of the action modifiers.

### Using Stateful Firewall Rules

You reference stateful firewall rules in service sets by using the **set stateful-firewall-rules rule** configuration command in the `[edit services service-set service-set]` hierarchy.

You can also create an ordered list of stateful firewall rules by configuring a stateful firewall rule set. You add a stateful firewall rule to a rule set in the `[edit services stateful-firewall]` hierarchy using the configuration command **set rule-set rule-set rule rule**. You then reference the rule set in service sets by using the **set stateful-firewall-rule-sets rule-set** configuration command in the `[edit services service-set service-set]` hierarchy.

## Stateful Firewall Rule Example

- Example: allow all connections in the output direction

```
[edit services stateful-firewall]
lab@R1# show
rule allow-all-outbound {
  match-direction output;
  term allow-all-algs {
    from {
      application-sets junos-algs-outbound;
    }
    then {
      accept;
    }
  }
  term allow-all-others {
    then {
      accept;
    }
  }
}
```



### Stateful Firewall Rule Example

The example on the slide allows all connections in the output direction. It has two terms. The first term (*allow-all-algs*) matches and accepts connections using any applications predefined in the JUNOS software. This term causes the router to use ALGs for application protocols that require them. The second term (*allow-all-others*) matches and accepts all other connections.

## **Agenda: Stateful Firewall and NAT**

---

- Stateful Firewall and NAT Overview
- Applications
- Configuring Stateful Firewall Rules
- Configuring NAT Rules
- Implementing Stateful Firewall and NAT
- Monitoring Stateful Firewall and NAT



### **Configuring NAT Rules**

The slide highlights the topic we discuss next.

## NAT Rules

- Configure in the `[edit services nat]` hierarchy
  - Supports match conditions: source address, destination address, and application
  - Supports actions: no-translation, translated
  - Supports action modifier: `syslog`
- Reference in service sets



### Configuring NAT Rules

You configure NAT rules in the `[edit services nat]` hierarchy, using the **`set rule rule`** command. You must configure a match direction and terms. You can configure a *from* clause for each term that describes the match conditions for the term. If there is no *from* clause, all connections match the term. If you want the AS PIC to use any ALGs when performing translation for a connection, the connection must match a term in either the stateful firewall or NAT rules that lists the application as a match condition in the *from* clause.

You must configure a *then* clause for each term. The *then* clause must include one of the actions and can also include the `syslog` action modifier.

### Using Stateful Firewall Rules

You reference NAT rules in service sets by using the **`set nat-rules rule`** configuration command in the `[edit services service-set service-set]` hierarchy.

You can also create an ordered list of NAT rules by configuring a NAT rule set. You add a NAT rule to a rule set in the `[edit services nat]` hierarchy using the configuration command **`set rule-set rule-set rule rule`**. You then reference the rule set in service sets by using the **`set nat-rule-sets rule-set`** configuration command in the `[edit services service-set service-set]` hierarchy.

## NAT Translation Options

- Translation type:
  - Source static
  - Source dynamic
  - Destination static
- Address pool information:
  - Configure a prefix or specify a pool
  - Specify source or destination pool
  - Specify an alternate address pool to use if the source pool is exhausted



### Supported Translation Types

You can configure the AS PIC to perform both static and dynamic source address and port translation or static destination address translation. However, the AS PIC can only perform one of these three kinds of translations for any connection in a single service set. For example, you can configure the AS PIC to perform destination static NAT for some connections and dynamic source PAT for other connections, but you cannot configure it to perform both destination static NAT and source static NAT for the same connection in the same service set.

### Address Pool Configuration

When you configure NAT, you must specify the address pool that the AS PIC should use for translation. You can specify the pool in two ways: you can reference a configured NAT pool or you can specify a prefix directly in the *then* clause of the NAT rule. Using address pools for dynamic NAT has some definite advantages, which we discuss on page 7-24.

In addition to configuring a source or destination address pool, you can also specify an overflow address pool, which the AS PIC uses for dynamic source NAT if the primary source pool is exhausted.

## NAT Translation Types

- **Static:**
  - 1:1 translation
  - NAT pool must be same size as the addresses matched in the from clause
- **Dynamic (source NAT only):**
  - Address dynamically assigned to a source from pool
    - NAT: 1:1 translation
    - PAT: Many:1 translation



### Static Translation

When you configure static translation, you are configuring the AS PIC to perform one-to-one translation between specific addresses. For example, if you configure the AS PIC to perform source translation for all connections from the 192.168.100.0/24 network and use 172.31.178.0/24 as the source address pool, the AS PIC translates the source addresses like this: 192.168.100.1 becomes 172.31.178.1, 192.168.100.2 becomes 192.168.100.2, and so on.

So that the AS PIC knows which post-NAT address to use for each pre-NAT address, you must configure an address pool that is the same size as the addresses matched in the term's *from* clause.

### Dynamic Translation

When you configure dynamic source NAT, you are configuring the AS PIC to perform a one-to-one translation between actual source addresses and translated addresses chosen from a pool. When the AS PIC sees traffic from a new source address, it temporarily allocates that source address an IP address from the NAT pool. The AS PIC performs source address translation for all connections from that source address to the address assigned from the NAT pool, and no traffic from other source addresses will be translated to the same address from the NAT pool.

*Continued on next page.*

## Dynamic Translation

Once the AS PIC stops seeing traffic from that source address, it returns the assigned address to the NAT pool, making it available for assignment to other source addresses. Because your NAT pool must only be large enough to support the number of source addresses that simultaneously have active connections, you can usually support a large number of source addresses with a smaller NAT pool.

When you configure the AS PIC to perform dynamic source PAT, you are configuring the AS PIC to perform a many-to-one source address translation. In this mode, the AS PIC translates the source address of connections from many active source addresses to a smaller number of addresses (often, only one). To support this, the AS PIC tracks the source TCP and UDP ports that are in use on the addresses in the translation pool. It translates the source port of TCP and UDP connections to a port that is currently unused for the address assigned in the translation pool. This process provides a unique source address and port combination. When remote hosts respond to the translated address and port, the AS PIC uses this unique source address and port combination to identify the appropriate recipient of the information.

## NAT Pools

- NAT pool benefits:
  - Allows reuse of the same pool in multiple terms
  - Supports PAT
- NAT pool configuration:
  - Configure properties in `[edit services nat pool pool]`
  - Configure addresses using `address` or `address-range` options
  - Enable PAT using the `port` option (use `port automatic` when using the router's interface address)



### Benefits of NAT Pools

When you configure a NAT pool, you can reuse the same pool in multiple terms. This ability provides a benefit anytime the same address pool must be used in multiple terms. Reusing the same pool allows you to make all changes to the NAT pool in only one location. In addition, if you want to perform PAT, you *must* use a NAT pool. You can configure a NAT pool for static source or destination NAT. However, using NAT pools is especially beneficial when using dynamic source NAT or PAT.

When you use a pool for static destination NAT, the pool must contain exactly one IP address. When you use a pool for source NAT, the pool can contain larger prefixes.

### Configuration of NAT Pools

You configure the properties of a NAT pool in the `[edit services nat pool pool]` hierarchy. You can configure the addresses that are part of the NAT pool using the `address` or `address-range` options. When you use the `address` option, you specify a prefix. When you use the `address-range` option, you specify the first and last addresses included in the range.

If you want to configure the AS PIC to perform PAT, you must include the `port` option. When you specify the `port` option, you can either specify the keyword `automatic` or a range of ports that can be used as source ports for translation. If you choose to use the router's interface addresses as part of the PAT pool, you should use the `automatic` keyword.



## NAT Pool Example (Without PAT)

- Example: Configure a NAT pool that assigns addresses from the 172.24.17.0/24 prefix without PAT

```
[edit services nat]
lab@R1# set pool examplepool address 172.24.17.0/24

[edit services nat]
lab@R1# show
pool examplepool {
    address 172.24.17.0/24;
}
```



### NAT Pool Example (Without PAT)

In the example on the slide, you configure a NAT pool called *examplepool* that includes the addresses from the 172.24.17.0/24 prefix. You can use this pool in multiple NAT rule terms.

## NAT Pool Example (With PAT)

- Example: Configure a NAT pool that uses port translation to translate connections into 172.24.17.3 and 172.24.17.4

```
[edit services nat]
lab0R1# set pool examplepatpool address-range low 172.24.17.3 high 172.24.17.4

[edit services nat]
lab0R1# show
pool examplepatpool {
  address-range low 172.24.17.3 high 172.24.17.4;
  port automatic;
}
```



### NAT Pool Example (With PAT)

In the example on the slide, you configure a NAT pool called *examplepatpool*. This pool is meant to be used as a source address pool for a NAT rule term that configures dynamic source NAT. When this pool is used in that way, the router uses PAT to translate source addresses and ports into the addresses 172.24.17.3 and 172.24.17.4 and uses the automatically chosen ports.

## NAT Example: Static Destination NAT

- Example: Configure the router to perform NAT for connections destined for the public address 172.31.17.12, translating them to the private address 192.168.168.5

```
[edit services nat]
lab@R1# show
rule server-nat {
  match-direction input;
  term server {
    from {
      destination-address {
        172.31.17.12/32;
      }
    }
    then {
      translated {
        destination-prefix 192.168.168.5/32;
        translation-type destination static;
      }
    }
  }
}
```



### Static Destination NAT Example

In the example on the slide, you want to translate connections with a destination address of 172.31.17.12 to use the destination address 192.168.168.5.

You configure a rule called *server-nat* with a single term that matches connections destined for the address 172.31.17.12. In the *then* clause, you configure the AS PIC to translate matching connections to the destination address 192.168.168.5.

## NAT Example: Dynamic Source PAT

- Example: Use PAT to translate outgoing connections to 172.24.17.3 and 172.24.17.4

```
[edit services nat]
lab0R1# show
pool examplepatpool {
    address-range low 172.24.17.3 high 172.24.17.4;
    port automatic;
}
rule example-pat-rule {
    match-direction output;
    term PAT-all {
        then {
            translated {
                source-pool examplepatpool;
                translation-type source dynamic;
            }
        }
    }
}
```



### Dynamic Source PAT Example

In the example on the slide, you want to translate outgoing connections to use the source addresses 172.24.17.3 and 172.12.17.4 by using PAT.

You configure a rule called *example-pat-rule* with a single term that matches all connections in the output direction. You configure a then clause that tells the AS PIC to perform dynamic source translation using the *examplepatpool*, which you configured on page 7-26.

When you configure NAT rules in conjunction with stateful firewall rules, any ALGs matched in the stateful firewall rules are also used in NAT operation automatically. However, if you configure this NAT rule to be used without any stateful firewall rules, you want to create two terms: one that matches the *junos-algs-outbound* application set and another that matches all other connections.

## **Agenda: Stateful Firewall and NAT**

---

- Stateful Firewall and NAT Overview
- Applications
- Configuring Stateful Firewall Rules
- Configuring NAT Rules
- ➔ **Implementing Stateful Firewall and NAT**
- Monitoring Stateful Firewall and NAT



### **Implementing Stateful Firewall and NAT**

The slide highlights the topic we discuss next.

## Interface Style Versus Next-Hop Style

- Interface style:
  - Supports PAT to interface address
- Next-hop style:
  - Supports multiple security zones
  - Separates routing protocol information
  - Preferred solution



### Interface-Style Stateful Firewall and NAT

Configuring interface-style service sets for stateful firewall and NAT allows you to avoid some configuration complexity. However, in many cases, the configuration complexity that comes with next-hop-style service sets actually proves to be valuable, providing a great deal of flexibility and separation between security zones. Among the challenges you face with interface-style service sets is managing routing information such that only external addresses are announced externally and only internal addresses are announced internally. Additionally, interface-style service sets do not process multicast traffic, leaving multicast traffic untranslated and uninspected.

There is one case when you must use interface-style service sets: configuring PAT using an external interface address in the NAT pool.

### Next-Hop-Style Stateful Firewall and NAT

When you configure next-hop-style service sets to provide stateful firewall and NAT services, you use virtual routers to separate security zones. The router maintains routing information and routing protocol information separately for each virtual router, maintaining separation between the security zones. You configure the next-hop-style service set to act as the conduit between the virtual routers, so only traffic that you allow in the stateful firewall rules passes between the virtual routers.

We recommend the next-hop-style service sets solution for stateful firewall and NAT implementations.

## Interface Style Implementation

- Configure on external interface(s)

- Example:

```
lab@R1# show interfaces fe-0/0/0
unit 0 {
  family inet {
    service {
      input {
        service-set SFW-OUTBOUND;
      }
      output {
        service-set SFW-OUTBOUND;
      }
    }
    address 10.210.9.177/28;
  }
}
```

- Advertise external addresses using BGP, if necessary



### Service Set Configuration

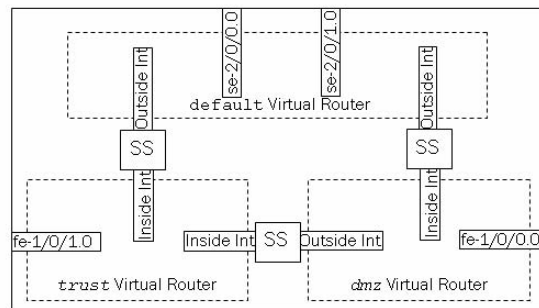
We discuss the configuration and application of interface-style service sets in Chapter 6, page 1, “JUNOS Services Overview”. The slide shows an example of applying a service set called *SFW-OUTBOUND* to the *fe-0/0/0.0* interface.

### Routing Protocol Implications

You must configure your routing protocols to advertise the correct addresses to the correct peers. In particular, if you configure the AS PIC to perform source NAT when sending traffic to the Internet but to not perform translation when sending traffic to peers, you must configure the router’s BGP sessions such that it advertises the NAT pool to your Internet service providers (ISPs) and the internal addresses to your partners.

## Next-Hop-Style Overview

- Virtual routers:
  - One routing instance per security zone
  - Service interfaces provide connections between routing instances
  - Routing protocols configured per routing instance



### Next-Hop-Style Configuration Overview

When you configure next-hop-style service sets to perform stateful firewall and NAT services, you must use virtual routers, which are implemented as a type of routing instance in the JUNOS software. For optimal security, you use a different virtual router for each security zone (except the untrusted zone, which should not be in a virtual router). For the virtual routers to communicate with each other, you connect them using a service set configured with appropriate rules.

You configure routing protocols and routing options (including static and aggregate routes) separately for each routing instance. Routing instances do not share routes or routing information unless you configure them to do so.



## Next-Hop-Style Implementation (1 of 3)

### ■ Configure service set

#### • Step 1: Configure service set

```
[edit]
lab@R1# show service service-set NH-TRUST-UNTRUST
stateful-firewall-rules allow-all-outbound;
nat-rules pat-all-outbound;
next-hop-service {
    inside-service-interface sp-0/0/0.2;
    outside-service-interface sp-0/0/0.1;
}
```

#### • Step 2: Prepare services interfaces

```
[edit interfaces sp-0/0/0]
lab@Tokyo# set unit 1 service-domain outside

[edit interfaces sp-0/0/0]
lab@Tokyo# set unit 2 service-domain inside

[edit interfaces sp-0/0/0]
lab@Tokyo# set unit 1 family inet

[edit interfaces sp-0/0/0]
lab@Tokyo# set unit 2 family inet
```



### Configuring the Service Set

We explained the configuration of next-hop-style service sets in Chapter 6, page 1, “JUNOS Services Overview”. The slide shows you configuring a next-hop style service set called *NH-TRUST-UNTRUST* and preparing the inside and outside services interfaces.

## Next-Hop-Style Implementation (2 of 3)

- Configure virtual routers:

- Do not put untrusted zone in a routing instance
- Configure one routing instance per zone

```
[edit]
lab@R1# set routing-instances trust instance-type virtual-router
```

- Assign interfaces to routing instance

```
[edit routing-instances trust]
lab@Tokyo# set interface fe-1/0/0.0
```

```
[edit routing-instances trust]
lab@Tokyo# set interface lo0.1
```

```
[edit routing-instances trust]
lab@Tokyo# set interface sp-0/0/0.2
```



### Configuring Virtual Routers

After you configure the service set, the next step is to configure the necessary virtual routers. You should configure one routing instance per security zone, except for the untrusted zone. You should not put the untrusted zone in a routing instance. You cannot configure the router to use an IPSec VPN endpoint that is in a virtual router. Therefore, if you configure the untrusted zone in a virtual router, you will not be able to configure IPSec VPN endpoints in the untrusted zone.

After you create a `virtual-router` type routing instance for each zone, you assign interfaces to be in the appropriate routing instances. Note that you assigned a new unit on the loopback interface to the routing instance. You can create one loopback unit for each routing instance. The router uses the firewall filters you configure on that unit for all local traffic in that routing instance. You also configure the inside services interface to be in the `trust` routing instance.

## Next-Hop-Style Implementation (3 of 3)

- Configure routing:

- AS PIC automatically adds static routes for NAT pools with a route preference of 1
- Add other static routes to both routing instances

```
[edit routing-instances trust routing-options]
lab@Tokyo# set static route default next-hop sp-0/0/0.2

[edit routing-options]
lab@Tokyo# set static route 172.31.127.10/32 next-hop sp-0/0/0.1
```

- Configure routing protocols in virtual router

```
[edit routing-instances trust protocols]
lab@Tokyo# set ospf area 0 interface fe-1/0/0.0
```



### Configure Routing

The AS PIC automatically adds static routes for NAT pools to the appropriate routing tables. These routes have a route preference of 1. You must configure any other static routes necessary to direct traffic to the appropriate services interfaces. Additionally, you must configure routing protocols appropriately for each routing instance.

Each routing instance has its own `protocols` and `routing-options` hierarchy under the `[edit routing-instances routing-instance]` hierarchy.

You can only configure routing protocols to run on interfaces that are in the same routing instance. Additionally, static routes can only use next hops (whether interface names or IP addresses) that are within the same routing instance.

## Sample Next-Hop-Style Configuration (1 of 2)

```

interfaces {
  sp-0/0/0 {
    unit 0 {
      family inet;
    }
    unit 1 {
      family inet;
      service-domain outside;
    }
    unit 2 {
      family inet;
      service-domain inside;
    }
  }
  fe-1/0/1 {
    unit 0 {
      family inet {
        address 192.168.100.1/24;
      }
    }
  }
  se-2/0/0 {
    unit 0 {
      family inet {
        address 172.31.145.2/30;
      }
    }
  }
  lo0 {
    unit 0 {
      family inet;
    }
    unit 1 {
      family inet {
        address 192.168.255.1/32;
      }
    }
  }
}

routing-options {
  aggregate {
    route 172.29.122.0/24;
  }
  autonomous-system 65109;
  protocols {
    bgp {
      group isp {
        export [ to-ISP reject-all ];
        neighbor 172.31.145.1 {
          description "isp-a";
          peer-as 65010;
        }
      }
    }
  }
  policy-options {
    prefix-list announce-to-ISP {
      172.29.122.0/24;
    }
    policy-statement reject-all {
      then reject;
    }
    policy-statement to-ISP {
      term match-routes {
        from {
          prefix-list announce-to-ISP;
        }
        then accept;
      }
    }
  }
}

```



### Sample Next-Hop-Style Configuration: Part 1

This slide and the next slide show a router configuration that uses next-hop-style service sets to protect interfaces in a virtual router called *trust* from the Internet. The service set performs both stateful firewall and NAT services. This is based on the diagram presented on page 7-32, but only the *trust* virtual router is configured.

The enterprise's public address pool is 172.29.122.0/24, which it announces to the ISP using BGP. The *trust* virtual router runs OSPF on its *fe-1/0/1.0* interface. The firewall rules allow all traffic from the *trust* virtual router to the untrusted zone and deny all traffic from the untrusted zone to the *trust* virtual router.

## Sample Next-Hop-Style Configuration (2 of 2)

```

routing-instances {
  trust {
    interface sp-0/0/0.2;
    interface fe-1/0/1.0;
    interface lo0.1;
    routing-options {
      static {
        route 0.0.0.0/0 next-hop sp-0/0/0.2;
      }
    }
    protocols {
      ospf {
        area 0.0.0.0 {
          interface fe-1/0/1.0;
        }
      }
    }
  }
}

services {
  stateful-firewall {
    rule allow-all-outbound {
      match-direction input;
      term allow-all-algs {
        from {
          application-sets junos-algs-outbound;
        }
        then {
          accept;
        }
      }
      term allow-all-others {
        then {
          accept;
        }
      }
    }
  }
}

nat {
  pool trust-pat-pool {
    address 172.29.122.1/32;
    port automatic;
  }
  rule pat-all-outbound {
    match-direction input;
    term PAT-all {
      then {
        translated {
          source-pool trust-pat-pool;
          translation-type source dynamic;
        }
      }
    }
  }
}

service-set NH-TRUST-UNTRUST {
  stateful-firewall-rules allow-all-outbound;
  nat-rules pat-all-outbound;
  next-hop-service {
    inside-service-interface sp-0/0/0.2;
    outside-service-interface sp-0/0/0.1;
  }
}

```



### Sample Next-Hop-Style Configuration: Part 2

This slide shows the remainder of the router configuration.

## **Agenda: Stateful Firewall and NAT**

---

- Stateful Firewall and NAT Overview
- Applications
- Configuring Stateful Firewall Rules
- Configuring NAT Rules
- Implementing Stateful Firewall and NAT
- Monitoring Stateful Firewall and NAT



### **Monitoring Stateful Firewall and NAT**

The slide highlights the topic we discuss next.

## Monitoring Routes (1 of 2)

### ■ Check routing tables:

- Main, showing automatic NAT static route

```
lab@R1> show route

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

[...]

172.29.122.1/32    *[Static/1] 3w0d 16:01:19
                  > via sp-0/0/0.1
```

- Routing instance

```
lab@R1> show route table trust.inet.0

trust.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 3w0d 19:51:19
                  > via sp-0/0/0.2
192.168.100.0/24   *[Direct/0] 3w0d 19:51:19
                  > via fe-1/0/1.0
192.168.100.1/32   *[Local/0] 3w0d 19:51:19
                  Local via fe-1/0/1.0
192.168.255.1/32   *[Direct/0] 3w0d 19:51:19
                  > via lo0.1
```



### Monitoring Routing Tables

When you encounter a problem with any service set implementation, first ensure that the appropriate routes appear in the routing table. You should ensure that the router has routes for both the forward and return paths of connections that are experiencing trouble.

The slide shows a static route that the AS PIC created for a source NAT pool. The slide also shows how to examine routing information for routing instances. When you create a `virtual-router` type routing instance, the router automatically creates a full set of routing tables for the routing instance. Just as the main router's main IPv4 routing table is designated `inet.0`, the `trust` routing instance's main IPv4 routing table is designated `trust.inet.0`.

You can use the command `show route a.b.c.d` to see the best route to the given address from each routing table on the router.

## Monitoring Routes (2 of 2)

- Monitor routing protocol operation:

- Use `instance` keyword

```
lab@R1> show ospf neighbor instance trust
```

Address	Interface	State	ID	Pri	Dead
192.168.100.3	fe-1/0/1.0	Full	192.168.100.3	128	35

```
lab@R1> show rip neighbor instance trust
```

```
[...]
```

```
lab@R1> show bgp neighbor instance trust
```

```
[...]
```

- traceroute, ping, telnet, ssh, and so on

- Use `routing-instance` keyword

```
lab@R1> traceroute 172.17.192.168 source 192.168.255.1 routing-instance trust
```

```
lab@R1> ping 172.17.192.168 source 192.168.255.1 routing-instance trust
```



### Monitoring Routing Protocol Operation

You monitor routing protocol operation using the same commands you would use if you were not using virtual routers. However, you might need to add the **`instance`** **`instance`** option to see information about routing protocol operation in routing instances.

### Generating Traffic

When you use tools that generate traffic locally (traceroute, ping, telnet, and so on), you can use the **`routing-instance`** **`instance`** option to have the router source the connections from a routing instance. If you specify this option, the router chooses an address from the specified routing instance and uses that instance's routing table to forward the traffic.



## Monitoring Sessions

- Monitor stateful firewall and NAT operation:

- Check session table

```
lab@R1> show services stateful-firewall flows
Interface: sp-0/0/0, Service set: NH-TRUST-UNTRUST
Flow
Flow      192.168.100.2:48694 -> 172.17.38.1:33438 Forward I      Frm count
NAT source 192.168.100.2:48694 -> 172.29.122.1:48694
UDP      172.17.38.1:33439 -> 172.29.122.1:48694 Forward O      0
NAT dest 172.29.122.1:48694 -> 192.168.100.2:48694
TCP      172.17.38.1:21 -> 172.29.122.1:4460 Watch O      2
NAT dest 172.29.122.1:4460 -> 192.168.100.2:4460
UDP      172.17.38.1:33440 -> 172.29.122.1:48694 Forward O      0
NAT dest 172.29.122.1:48694 -> 192.168.100.2:48694
UDP      192.168.100.2:48694 -> 172.17.38.1:33440 Forward I      1
NAT source 192.168.100.2:48694 -> 172.29.122.1:48694
TCP      192.168.100.2:4460 -> 172.17.38.1:21 Watch I      3
NAT source 192.168.100.2:4460 -> 172.29.122.1:4460
UDP      192.168.100.2:48694 -> 172.17.38.1:33439 Forward I      1
NAT source 192.168.100.2:48694 -> 172.29.122.1:48694
UDP      172.17.38.1:33438 -> 172.29.122.1:48694 Forward O      0
NAT dest 172.29.122.1:48694 -> 192.168.100.2:48694
```

- State: *Watch* indicates accepted and ALG active, *Forward* indicates accepted without ALG, *Drop* indicates discarded, *Reject* indicates rejected



### Monitoring Sessions

The command **show services stateful-firewall flows** shows the session table used for both stateful firewall and NAT. (Even if you only configured NAT rules, you still use this command to see the session table.)

The NAT line that appears under these flows shows which address field is being translated along with both the original and the translated address and port. Two entries exist in the session table for each connection, one entry for each direction of communication.

The *State* field shows whether the flow is being forward, discarded, or rejected. Additionally, the router displays the *Watch* state when the AS PIC uses an ALG to process the connection. If the router displays the *Forward* state, the AS PIC is not using an ALG to process the connection.

The *Dir* field indicates whether the flow is in the *input* or *output* direction. (Recall that these directions often seem backwards for next-hop service sets.)

The *Frm count* field indicates the number of packets that matched this flow in the session table.

## Monitoring Traffic Statistics

- Monitor traffic statistics:

```
lab@R1> show services stateful-firewall statistics
Interface  Service set  Accept  Discard  Reject  Errors
sp-0/0/0   NH-TRUST-UNTRUST  328      56        0        6

lab@R1> show services stateful-firewall statistics extensive
Interface: sp-0/0/0
Service set: sffw-outbound
New flows:
  Accepts: 0, Discards: 0, Rejects: 0
Existing flows:
  Accepts: 328, Discards: 56, Rejects: 0
Drops:
  IP option: 0, TCP SYN defense: 0
  NAT ports exhausted: 0
Errors:
  IP: 0, TCP: 0
  UDP: 6, ICMP: 0
  Non-IP packets: 0, ALG: 0
IP errors:
  IP packet length inconsistencies: 0
  Minimum IP header length check failures: 0
  Reassembled packet exceeds maximum IP length: 0
  Illegal source address: 0
  Illegal destination address: 0
  TTL zero errors: 0, Illegal IP protocol number (0 or 255): 0
  Land attack: 0
[...]
```



### Monitoring Traffic Statistics

You can monitor traffic-processing statistics with the **show services stateful-firewall statistics** command. Adding the **extensive** keyword will give you extra detail, which can be especially helpful in understanding the origin of errors listed in the summary statistics.

## Monitoring NAT Pools

- Monitor NAT pools:

```
lab@R1> show services nat pool
Interface: sp-0/0/0, Service set: NH-TRUST-UNTRUST
NAT pool      Type      Address      Port      Ports used
trust-pat-pool dynamic    172.29.122.1-172.29.122.1

lab@R1> show services nat pool detail
Interface: sp-0/0/0, Service set: NH-TRUST-UNTRUST
NAT pool: trust-pat-pool, Translation type: dynamic
Address range: 172.29.122.1-172.29.122.1
Out of address errors: 2
```



### Monitoring NAT Pools

You can list NAT pools with the **show services nat pool** command. This command lists all address pools, whether the pool is a prefix configured directly in a NAT rule term or a named pool. You can add the **detail** keyword to see the number of times the AS PIC has been unable to allocate an address from the pool due to address exhaustion. (For PAT operation, you can see the number of times the AS PIC has been unable to allocate a port due to port exhaustion using the **show service stateful-firewall statistics extensive** command.)

## Clearing Session Table and Statistics

- Clear flows:

- Not automatically done on rule change
- Prints number of sessions removed

```
lab@R1> clear services stateful-firewall flows
Interface  Service set  Conv removed
sp-0/0/0   NH-TRUST-UNTRUST  8
```

- Reset statistics:

```
lab@R1> clear services stateful-firewall statistics
```



### Clearing the Session Table

When you change the stateful firewall or NAT rules associated with a service set, existing flows remain in the session table and are unaffected by the change. There are times when you want to clear the session table to delete unwanted sessions after making rule changes. You can clear all sessions using the **clear services stateful-firewall flows** command with no options. To delete only some sessions, you can use options that specify match criteria. The router only deletes flows matching the specified criteria.

### Resetting Statistics

To clear the statistics found in the **show service stateful-firewall statistics** command output, use the **clear services stateful-firewall statistics** command.

## Review Questions

1. What are ALGs? How do they affect stateful firewall operation? How do they affect NAT operation?
2. When will the AS PIC use an ALG to process a connection?
3. What are NAT pools?
4. How do you configure the AS PIC to perform PAT?
5. Why should you use next-hop-style service sets for stateful firewall and NAT?



### This Chapter Discussed:

- Configuring a stateful firewall policy using the JUNOS CLI;
- Configuring a NAT policy using the JUNOS CLI; and
- Monitoring stateful firewall and NAT operation using the JUNOS CLI.

## Lab 5: Stateful Firewall and NAT

- Configure an interface-style service set to provide stateful firewall and NAT services.
- Configure a next-hop-style service set to provide stateful firewall and NAT services.



### Lab 5: Stateful Firewall and NAT

This slide lists the objectives for this lab.



# **Advanced Juniper Networks Routing in the Enterprise**

## **Chapter 8: IPSec VPNs**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - Configure an IPSec VPN using the JUNOS CLI
  - Configure an IPSec-over-GRE tunnel using the JUNOS CLI
  - Monitor VPN operations



### This Chapter Discusses:

- Configuring an IP Security (IPSec) virtual private network (VPN) using the JUNOS software CLI;
- Configuring an IPSec-over-generic routing encapsulation (GRE) tunnel using the JUNOS software CLI; and
- Monitoring VPN operations.



## Agenda: IPSec VPNs

---

- IPSec VPN Overview
- IPSec VPN Configuration
- Implementation Considerations
- Monitoring



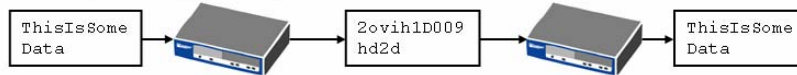
### IPSec VPN Overview

The slide lists the topics we cover in this chapter. We discuss the highlighted topic first.

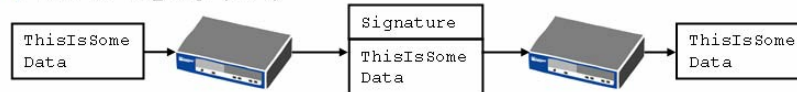
## IPSec Overview

- IPSec VPNs provide:

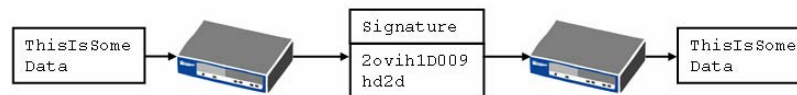
- Data privacy (ESP)



- Data integrity (AH)



- Or both (bundle)



### IPSec Overview

You can use IPSec to protect your data in two ways. First, you can protect your data from being viewed in transit by third parties by encrypting it. Second, you can protect your data from being modified in transit by third parties by authenticating the data. You can configure the JUNOS software to perform both these functions, either separately or together.

## IPSec VPN Establishment

- Dynamic mode has two parts:
  - IKE: Provides mechanism for securely establishing IPSec tunnels
  - IPSec: Provides the actual encrypted or authenticated (or both) tunnel
- IPSec SAs:
  - Always tunnel mode
  - Specific source and destination pair



### IPSec VPN Establishment

Many companies configure their devices to use dynamic IPSec security associations (SAs) when establishing their IPSec VPNs. When you configure the devices to use dynamic SAs, the devices use a two-step process to establish IPSec VPNs. First, they use the Internet Key Exchange (IKE) protocol to provide a secure channel to exchange session keys for the actual IPSec tunnels. These secure channels between endpoints are called IKE security associations (SAs). Usually only one IKE security association is active at any time between a unique pair of endpoints.

After the devices establish an IKE SA, they establish the actual IPSec tunnels that will carry the data between the endpoints. These tunnels are called *IPSec security associations (SAs)*. When the router uses dynamic SAs, it creates limited-lifetime SAs. When the SA expires, the device rekeys the tunnel with a new random key.

### IPSec Security Associations

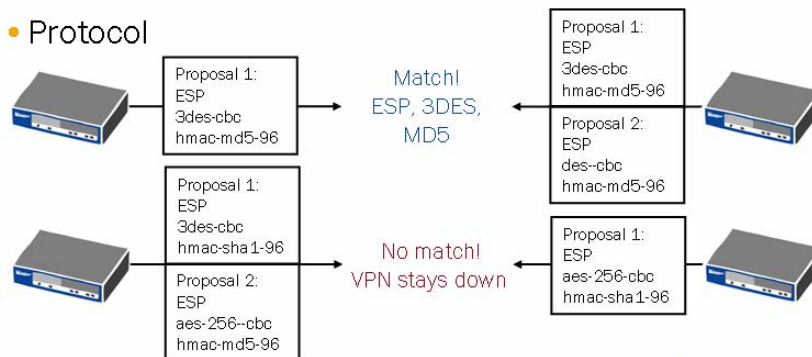
RFC 4301 (the IPSec standard) specifies two modes of operation: transport mode and tunnel mode. The JUNOS software always uses the tunnel mode, which encapsulates the original packets within an outer IPSec header. The original packet is unmodified during transit and is unchanged upon de-encapsulation, except that the time-to-live (TTL) field is decremented.

Each IPSec SA is unidirectional and specifies the way that traffic is encrypted from a specific source prefix to a specific destination prefix. Therefore, the JUNOS software establishes two SAs for each source/destination pair you configure—one SA for traffic in each direction.

## IPSec Negotiation

- Endpoints negotiate tunnels using *proposals* that define parameters

- Encryption algorithm
- Authentication algorithm
- Protocol



### Proposals

When two endpoints try to establish IKE or IPSec tunnels, they exchange proposals that define the parameters they are willing to use to secure the session. Proposals list a complete set of all negotiable parameters. Both devices must find at least one identical proposal for the session to establish. If they cannot find matching proposals, they do not establish an SA. (One of the exceptions to this requirement is SA lifetime. The two endpoints will negotiate to use the lower of the two SA lifetimes proposed.)

The examples on the slide show the negotiation of an IPSec tunnel. In the first example, both routers have a proposal that specifies the use of the Encapsulating Security Payload (ESP) protocol encrypted with the triple Data Encryption Standard with cipher block chaining (3DES-CBC) and authenticated with the HMAC-MD5 hash algorithm. Because they found a matching proposal, they will use this proposal for the IPSec tunnel.

In the second example, the two routers cannot find a matching proposal, so they do not establish an IPSec SA. Although the router on the left has proposals that specify the use of the 3DES-CBC encryption algorithm and the HMAC-SHA1 hash algorithm, it does not have a proposal that specifies that they should be used together. Therefore, because the two routers cannot find an exact match, they do not establish an IPSec SA.

Routers exchange proposals both for the IKE session itself and also for the IPSec tunnels. The IKE and IPSec proposals need not use the same security parameters. You can list multiple proposals. The router prefers to use the proposals in the order you list them in the configuration.

## Agenda: IPSec VPNs

---

- IPSec VPN Overview
- IPSec VPN Configuration
- Implementation Considerations
- Monitoring



### IPSec VPN Configuration

The slide highlights the topic we discuss next.

## IPSec VPN Configuration Overview

- Configure under `[edit services]`
- Many related configuration elements:
  - Service sets reference IPSec VPN rules
  - IPSec VPN rules reference IKE and IPSec policies
  - IKE and IPSec policies reference IKE and IPSec proposals



### IPSec VPN Configuration Overview

The JUNOS software contains two completely different ways to configure and monitor IPSec VPNs; however, you must use the correct method for your hardware. If you use the Encryption Services (ES) PICs, you configure the IPSec VPNs under the `[edit security ipsec]` hierarchy. This class does not cover the configuration of the ES PIC.

If you are using the AS PIC (including the ASM or J-series router), you configure IPSec VPNs under the `[edit services]` hierarchy. You configure service sets directly under the `[edit services]` hierarchy. (Note that service sets that reference IPSec VPN rules cannot contain any other types of service rules. A service set can either perform IPSec VPN functions or it can perform some combination of stateful firewall, NAT, and IDS.) You configure the remainder of the IPSec VPN parameters in the `[edit services ipsec-vpn]` hierarchy.

### Configuration Relationships

The IPSec VPN configuration contains many related elements. You must understand these relationships so that you can appropriately understand and configure IPSec VPNs. A service set can reference one or more IPSec VPN rules. An IPSec VPN rule references an IKE policy and an IPSec policy. IKE and IPSec policies reference one or more IPSec proposals. While this configuration structure might appear fairly complex to use, this configuration structure provides a great deal of flexibility.

## IPSec VPN Configuration Relationships

```
[edit services]
lab@R1# show
service-set vpn-to-Tokyo {
  next-hop-service {
    inside-service-interface sp-0/0/0.2;
    outside-service-interface sp-0/0/0.1;
  }
  ipsec-vpn-options {
    local-gateway 172.17.37.4;
  }
  ipsec-vpn-rules HongKong-to-Tokyo;
}
ipsec-vpn {
  rule HongKong-to-Tokyo {
    term LANs {
      from {
        source-address {
          192.168.200.0/24;
        }
        destination-address {
          192.168.201.0/24;
        }
      }
      then {
        remote-gateway 172.17.38.4;
        dynamic {
          ike-policy main_mode_ike_policy;
          ipsec-policy dynamic_ipsec_policy;
        }
      }
    }
  }
  match-direction input;
}

ipsec {
  proposal esp_shal_3des_ipsec_proposal {
    protocol esp;
    authentication-algorithm hmac-shal-96;
    encryption-algorithm 3des-cbc;
  }
  policy dynamic_ipsec_policy {
    perfect-forward-secrecy {
      keys group2;
    }
    proposals esp_shal_3des_ipsec_proposal;
  }
}
ike {
  proposal psk_shal_3des_ike_proposal {
    authentication-method pre-shared-keys;
    authentication-algorithm shal;
    encryption-algorithm 3des-cbc;
  }
  policy main_mode_ike_policy {
    mode main;
    proposals psk_shal_3des_ike_proposal;
    pre-shared-key ascii-text
      "$9$kgT3At0Rc10B1MLNY2UjH"; ## SECRET-DATA
  }
  traceoptions {
    flag ike;
  }
  establish-tunnels immediately;
}
```



Juniper your Net

8-9

Copyright © 2006 Juniper Networks, Inc.

Proprietary and Confidential www.juniper.net

### IPSec VPN Configuration Relationships

The slide shows a sample IPSec VPN configuration and highlights the interdependencies that exist between the various components.

## Service Set Configuration

- Next-hop style vs. interface style:
  - Interface style:
    - Does not support multicast traffic
    - Secures unicast traffic headed *through* an interface
  - Next-hop style:
    - Supports multicast traffic
    - Supports routing protocols directly over an IPSec tunnel
    - Secures traffic routed to inside interface
- All IPSec rules in a service set use the same local endpoint



### Next-Hop-Style Versus Interface-Style Service Sets

For IPSec VPNs, you should typically use next-hop-style service sets. When you use next-hop-style service sets, you can use the inside services interface in much the same way you would use a GRE interface. You can configure the router to run routing protocols over the tunnel, and you can configure static routes that specify the inside interface as the next hop.

If you choose to use an interface-style service set, traffic is encrypted as it leaves an interface on the router. Traffic that you want to secure must be routed out an interface towards the destination. As the router processes the traffic for transmission, it sends the traffic to the AS PIC to be secured. Because the routing table makes it appear as though the traffic is following the normal (unsecured) path to the destination, it might be less obvious that the router is actually encrypting the traffic. Additionally, with an interface-style service set, you cannot secure multicast traffic or run a routing protocol over the tunnel. The one time you should use an interface-style service set is when you configure an IPSec-over-GRE tunnel and use the same endpoints for both the GRE and IPSec tunnels (see page 8-22).

You can see the difference in the way the two types of service sets are processed in the chart on page 6-32 and the detailed examples in Appendix A, "Life of a Packet".

### One Local Endpoint per Service Set

You can configure tunnels to multiple remote endpoints in a single service set. However, all the tunnels will share the same local address.



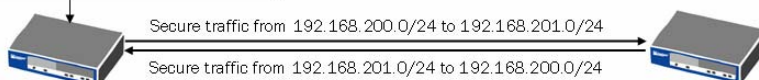
## IPSec VPN Rules (1 of 2)

- Specify what traffic to secure
  - Must specify exactly one source prefix and one destination prefix in the *from* clause of each term
  - Specify address pairs in outbound direction—SA for return traffic is automatically created
  - Must match remote peer's configuration!

```

Next-hop-style
match-direction input;
term LANS {
  from {
    source-address {
      192.168.200.0/24;
    }
    destination-address {
      192.168.201.0/24;
    }
  }
}

```



### Selecting Traffic to Secure

As we mentioned, each IPsec SA is unidirectional and specifies how traffic from a specific source prefix to a specific destination prefix will be secured (see page 8-5). When you configure IPsec VPN rules, each term must contain a *from* clause that specifies exactly one source prefix and one destination prefix. (If you fail to specify a source prefix, a destination prefix, or both, the router will use the default value of 0.0.0.0/0 for the prefix.) The router uses these addresses and the match direction you configure to determine the local and remote addresses between which it should secure traffic.

You should configure the router to match the traffic that you want secured in the outbound direction. (For next-hop service sets, this is the *input* direction. For interface-style service sets, this is the *output* direction.) The router automatically creates bidirectional SAs, so merely specifying that traffic should be secured in the outbound direction causes the router to automatically secure traffic in both directions.

In the example on the slide, we configured the router to secure traffic that has source addresses in the 192.168.200.0/24 prefix and destination addresses in the 192.168.201.0/24 prefix. Because this rule is used in a next-hop-style service set and the match direction is *input*, the router knows the source address is the local address and the destination address is the remote address. The router automatically creates bidirectional SAs.

For two IPsec peers to establish an IPsec SA, they must agree on the source and destination addresses they will secure. Therefore, the addresses we specify in our *from* statement must exactly match the addresses we configured the remote side to secure.

## IPSec VPN Rules (2 of 2)

- Specify how to secure traffic in *then* clauses
  - Remote gateway
  - Manual or dynamic SA (and parameters)
  - Tunnel MTU
  - Syslog
  - Antireplay (on by default)



### Specifying the Security

After you select traffic that the router is to secure using the *from* clause, you must tell the router how to secure it in the *then* clause. You use the *then* clause to configure the remote endpoint for the IPSec VPN tunnel. You also use the *then* clause to configure the router to establish a manual or dynamic SA and to specify the related parameters.

By default, the IPSec tunnel MTU is 1500 bytes and the router uses antireplay protection for the IPSec SA. You can change these defaults in the *then* clause. You can also configure the router to log information about the packet using the syslog functions.

## IPSec VPN Service Set and Rule Example

- Secure data between local prefix 192.168.100.0/16 and remote prefix 192.168.150.0/16:

```

services {
  service-set example-VPN {
    interface-service {
      service-interface sp-0/0/0;
    }
    ipsec-vpn-options {
      local-gateway 172.17.201.17;
    }
    ipsec-vpn-rules example-VPN-rule;
  }
  ipsec-vpn {
    rule example-VPN-rule {
      term LANs {
        from {
          source-address {
            192.168.100.0/16;
          }
          destination-address {
            192.168.150.0/16;
          }
        }
        then {
          remote-gateway 172.19.166.4;
          dynamic {
            ike-policy main mode_ike_policy;
            ipsec-policy dynamic_ipsec_policy;
          }
        }
      }
    }
  }
  match-direction output;
}
[...]
```



### IPSec VPN Service Set and Rule Example

In the example on the slide, you configure a service set to secure data between local prefix 192.168.100.0/16 and remote prefix 192.168.150.0/16.

## Dynamic SAs

- Dynamic SAs are the most common form of VPN
  - Specify IKE policy
  - Specify IPsec policy
- Two policies, two purposes:
  - IKE policy defines the parameters for the secure channel that the endpoints use to establish IPsec SAs
  - IPsec policy defines the security parameters for the IPsec SAs
  - Both use proposals to define acceptable combinations of security parameters



### Dynamic Security Associations

Dynamic SAs have a two-step process for establishing VPNs. First, IKE establishes a secure channel over which to exchange keys. Then, the routers use this secure channel to establish limited-lifetime IPsec SAs that use dynamically generated keys that can be regenerated when the IPsec SA expires. Because the IKE process handles the IPsec SA establishment, it is easy to add additional IPsec SAs. Additionally, the dynamically generated (and regenerated) keys are more secure than statically configured keys would be for each SA.

When you use dynamic SAs, you must configure both an IKE policy and an IPsec policy.

### IKE and IPsec Policies

IKE policies define the parameters used to create the IKE SA. The IPsec policy defines the security parameters used for the IPsec SA that actually secures data. Both policies use proposals to define acceptable combinations of security parameters. The router attempts to use the proposals in the order you list them in the configuration. It attempts to use the most preferred proposal that the peer also supports.

## Common IKE and IPSec Options

### ■ IKE options:

- Mode: Main mode (default) provides identity protection during session start—aggressive mode does not
- Diffie-Hellman Group 1 (768-bit) vs. Group 2 (1024-bit)

### ■ IPSec options:

- Protocol: ESP (encryption and authentication), AH (strong authentication), bundle (ESP encryption + AH authentication)
- PFS: New keys are not derived from old keys but are rekeyed using Diffie-Hellman Group 1 or Group 2



### IKE Options

IKE has two modes of operation: *main* and *aggressive*. Both provide a secure session start. In main mode, the initial exchange takes longer, but the peers can hide their identity. In aggressive mode, the initial exchange is shorter, there is less negotiation, and the peers cannot hide their identity. The initiator determines the mode used.

The Diffie-Hellman algorithm allows two devices to securely establish a shared session key without ever transmitting the key in cleartext. You can configure the router to use Diffie-Hellman Group 1 or Group 2 when performing Diffie-Hellman exchanges. Because Group 2 supports a larger range of prime numbers, it provides more security, but it also requires more processing power when performing the Diffie-Hellman exchange.

### IPSec Options

You can configure the router to secure traffic using IPSec three different ways. You can configure the router to secure traffic with ESP, which supports both encryption and authentication. You can configure the router to secure traffic with an authentication header (AH), which authenticates the entire packet (including headers). You can also configure the router to secure traffic using ESP for encryption and AH for authentication. This last mode is known as the *bundle* mode.

If you want the router to completely regenerate new keys for every IPSec SA when the SA expires, without deriving them from old keys, you should configure the Perfect Forward Secrecy (PFS) protocol. When you configure PFS, you can configure the router to use Diffie-Hellman Group 1 or Group 2 to generate the new keys.

## IKE Policy Example

### ■ Example:

- Establish a main-mode IKE session and secure the IKE communication with a preshared key and the AES-256 encryption algorithm
- Use the SHA1 hashing algorithm for authentication and Diffie-Hellman Group 2

```
[edit services ipsec-vpn]
lab0R1# show
[...]
ike {
  proposal psk_sha1_aes256_ike_proposal {
    authentication-method pre-shared-keys;
    dh-group group2;
    authentication-algorithm sha1;
    encryption-algorithm aes-256-cbc;
  }
  policy main_mode_ike_policy {
    mode main;
    proposals psk_sha1_aes256_ike_proposal;
    pre-shared-key ascii-text
"$9$kgT3AtORc10B1MLNY2UjH"; ## SECRET-DATA
  }
}
[...]
```



### IKE Policy Example

The example on the slide shows an IKE policy and a proposal that it references.

## IPSec Policy Example

### ■ Example:

- Secure the data with ESP using the 3DES encryption algorithm
- Use the MD5 hashing algorithm for authentication and PFS with Diffie-Hellman Group 2

```
[edit services ipsec-vpn]
lab@R1# show
[...]
ipsec {
  proposal esp_md5_3des_ipsec_proposal {
    protocol esp;
    authentication-algorithm hmac-md5-96;
    encryption-algorithm 3des-cbc;
  }
  policy dynamic ipsec_policy {
    perfect-forward-secrecy {
      keys group2;
    }
    proposals esp_md5_3des_ipsec_proposal;
  }
}
[...]
```



### IPSec Policy Example

The example on the slide shows an IPSec policy and a proposal that it references.

## **Agenda: IPSec VPNs**

---

- IPSec VPN Overview
- IPSec VPN Configuration
- ➔ **Implementation Considerations**
- Monitoring



### **Implementation Considerations**

The slide highlights the topic we discuss next.



## Minimum Configuration Example

- The minimum configuration necessary to establish a next-hop-style VPN:

```
[edit]
lab@R1# show interfaces sp-0/0/0
unit 0 {
    family inet;
}
unit 1 {
    family inet;
    service-domain outside;
}
unit 2 {
    family inet;
    service-domain inside;
}
[edit]
lab@R1# show services
service-set NH-basic-vpn {
    next-hop-service {
        inside-service-interface sp-0/0/0.2;
        outside-service-interface sp-0/0/0.1;
    }
    ipsec-vpn-options {
        local-gateway 10.14.8.21;
    }
    ipsec-vpn-rules simple-rule;
}

ipsec-vpn {
    rule simple-rule {
        term secure-everything {
            then {
                remote-gateway 10.14.8.22;
                dynamic {
                    ike-policy basic-ike-policy;
                }
            }
        }
    }
    match-direction input;
}
ike {
    policy basic-ike-policy {
        pre-shared-key ascii-text
        "$9$rwV187ws42Dkgo"; ## SECRET-DATA
    }
}
```



### Minimum IPSec Configuration Example

The slide shows the minimum configuration necessary to establish a next-hop-style VPN between two endpoints. Using this configuration causes the router to secure all traffic routed to the inside interface of the VPN.

## Minimum Configuration Parameters

- The minimum configuration results in these default parameters:
  - IKE: Main mode, 3DES-CBC, SHA1, preshared key, 3600-second lifetime
  - IPSec: ESP, 3DES-CBC encryption, HMAC-SHA1-96 authentication, 28800-second lifetime, antireplay, no PFS



### Minimum Configuration Parameters

If you configure the router with the minimum configuration shown on the previous slide, the router uses the default parameters shown on this slide to secure the traffic.

## Incoming IPsec Processing

- Forwarding table entry for destination address/source address/protocol tuple directs traffic to the AS PIC
  - For both interface-style and next-hop-style service sets, return traffic is sent to the AS PIC regardless of incoming interface

```
lab@R1> show route forwarding-table
Routing table: inet
Internet:
Destination      Type RtRef Next hop          Type Index NhRef Netif
[...]
172.17.37.4.172.17.38.4.50/72
user             0
service          337    3
172.17.37.4.172.17.38.4.51/72
user             0
service          337    3
[...]
```



### Incoming IPsec Processing

When an IPsec peer sends IPsec traffic to the router, it must know to direct this traffic to the AS PIC for some combination of de-encapsulation, decryption and authentication. With interface-style service sets, it might seem obvious the way this happens. As the traffic enters the interface, a service filter directs the PFE to send the traffic to the AS PIC to be processed by the appropriate service set. However, how this process works is probably slightly less obvious with next-hop-style service sets.

When you configure a service set with IPsec rules, the router installs an entry in the forwarding table to ensure that the PFE knows to send all IPsec traffic to the AS PIC for processing. This forwarding table entry matches on the destination address/source address/protocol tuple. Because these three fields are a total of 72 bits and the PFE should only follow this forwarding table entry if it finds an exact match in all three fields, the forwarding table entry has a /72 bit length.

The router installs these entries in the forwarding table for both next-hop-style and interface-style service sets. Therefore, if you use an interface-style service set and any traffic arrives on an interface where you did not apply the service set, the PFE still sends the traffic to the AS PIC when it performs the route lookup.

## IPSec-over-GRE Tunnels

- An IPSec-over-GRE tunnel is a GRE tunnel encrypted by IPSec
  - Commonly used for interoperability with older Cisco IOS images
  - Should use interface-style service set if the IPSec and GRE endpoints are the same
- Might require special configuration to permit GRE traffic
  - Must permit the *outbound* GRE packets through the *inbound* stateless firewall filters
  - Must use service filters to skip Layer 3 service processing of the *outbound* GRE packets through the *inbound* services



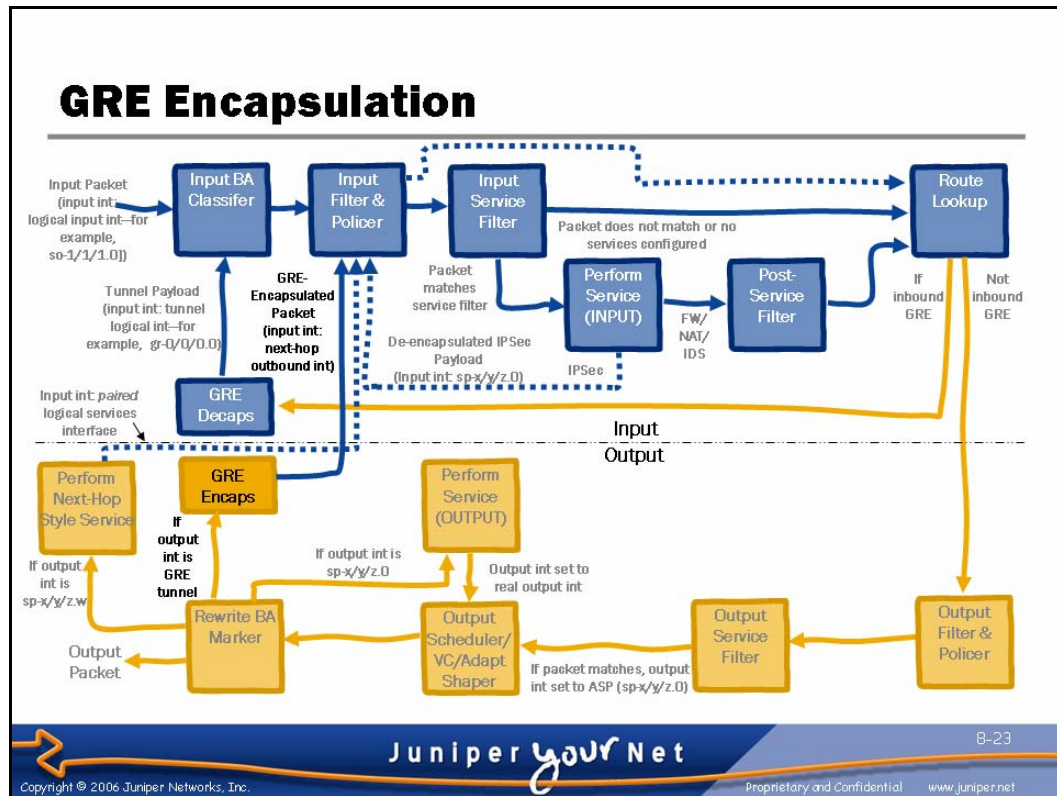
### IPSec-over-GRE Tunnels

Contrary to what the name might imply, IPSec-over-GRE tunnels are really GRE tunnels that are secured within IPSec. Therefore, to configure an IPSec-over-GRE tunnel, you first establish an IPSec SA and then establish a GRE tunnel over this SA. Additionally, if the IPSec and GRE endpoints are the same, you should use an interface-style service set. If you try to establish an IPSec-over-GRE tunnel with a next-hop-style service set and the same endpoints on both sides, you will be unable to use simple static routes to route the GRE traffic over the IPSec tunnel and then also correctly route the IPSec traffic.

You can use a next-hop-style service set if you use filter-based forwarding to route the GRE traffic over the IPSec tunnel. To implement this scenario, you must take advantage of a peculiarity in the way the JUNOS software routes GRE traffic.

### GRE Recirculation

When you encapsulate packets within a tunnel using either a Tunnel PIC or an AS PIC, the original packet is sent to the PIC for processing. After the PIC encapsulates the packet, it sends the packet back to the PFE so the router can route the encapsulated packet appropriately. When it sends the packet back to the PFE, it tells the PFE that the packet's inbound interface is the *outbound* interface the packet will follow to reach its destination. The PFE then performs inbound processing as if this packet had arrived on that interface. You might need to make certain configuration changes to accommodate this traffic.



### GRE Encapsulation Process

The packet processing chart on the slide shows the way GRE encapsulation is processed through a T-series, M-series, or J-series router. A route lookup determines that the next-hop interface for a packet is a GRE tunnel. The PFE directs the FPC to transmit the packet to the Tunnel PIC or AS PIC. The PIC encapsulates the packet and sends the encapsulated packet back to the FPC, telling the PFE that the input interface is the next-hop output interface. The PFE then performs normal input processing, beginning with the logical interface policer.

You must account for this recirculation when configuring the *input* processing parameters for the next-hop *outbound* interface for GRE packets. In particular, you must account for the extra traffic in interface policers, you must allow the *outbound* packets through the *input* filters on the *outbound* interface, and you must ensure that the *outbound* packets are not subjected to Layer 3 *input* services on the *outbound* interface. It is a common configuration mistake to forget to write and apply service filters that skip the *outbound* packets from being serviced by the *input* services on the *outbound* interface.

Incidentally, understanding the way this packet flow occurs can also provide you some additional flexibility. For example, you can use the knowledge of packet recirculation to support IPSec-over-GRE tunnels using next-hop-style service sets (see page 8-26 for a sample configuration).

We show a full explanation of IPSec-over-GRE packet processing in Appendix A.

## IPSec-over-GRE Tunnel Example (1 of 3)

- Establish an IPSec-over-GRE tunnel from 172.17.37.4 to 172.17.38.4:

```
[edit]
lab@R1# show interfaces gr-0/0/0
unit 0 {
  tunnel {
    source 172.17.37.4;
    destination 172.17.38.4;
  }
  family inet {
    address 192.168.25.129/30;
  }
}

[edit]
lab@HongKong# show interfaces fe-2/0/1 unit 0
family inet {
  service {
    input {
      service-set gre-vpn service-filter match-vpn-no-gre;
      service-set sffw-outbound service-filter no-gre;
    }
    output {
      service-set gre-vpn service-filter match-vpn;
      service-set sffw-outbound;
    }
  }
  address 172.17.37.4/29;
}

[edit]
lab@R1# show firewall family inet
service-filter match-vpn {
  term vpn {
    from {
      source-address {
        172.17.37.4/32;
      }
      destination-address {
        172.17.38.4/32;
      }
    }
    then service;
  }
  term default {
    then skip;
  }
}
service-filter no-gre {
  term ignore-gre {
    from {
      protocol gre;
    }
    then skip;
  }
  term default {
    then service;
  }
}
[...]
```



### IPSec-over-GRE Tunnel Example: Part 1

In this sample configuration, we establish an IPSec-over-GRE tunnel to a Cisco IOS device. The local GRE and IPSec endpoint is 172.17.37.4 and the remote endpoint is 172.17.38.4. Several things worth noting about this configuration include the following:

- Because the GRE and IPSec endpoints are the same, we have used an interface-style service set. (We show an example of a next-hop style service set on page 8-26.)
- On older Cisco IOS images, you must use an access list to express precisely which traffic will be secured. These access-list entries must exactly match the source and destination prefixes you list in the from terms of your rules. Because you only want to secure GRE traffic between the two endpoints, you specify these addresses as the source and destination addresses with /32 masks in both the Cisco and Juniper Networks configurations.
- You configure and apply service filters for all the input services that prevent GRE traffic from being processed by the input services on the output interface. (Also note that you must only configure the input service filter for the VPN to match the actual IPSec traffic itself, rather than the traffic encapsulated within the IPSec tunnel.)
- Once this tunnel is established, you can run dynamic routing protocols over the GRE tunnel or route traffic to the remote peer using static routes.

## IPSec-over-GRE Tunnel Example (2 of 3)

```
[edit]
lab@R1# show firewall family inet
[...]
service-filter match-vpn-no-gre {
  term ignore-gre {
    from {
      protocol gre;
    }
    then skip;
  }
  term vpn {
    from {
      source-address {
        172.17.38.4/32;
      }
      destination-address {
        172.17.37.4/32;
      }
    }
    then service;
  }
  term default {
    then skip;
  }
}

[edit]
lab@R1# show services
service-set gre-vpn {
  interface-service {
    service-interface sp-0/0/0;
  }
  ipsec-vpn-options {
    local-gateway 172.17.37.4;
  }
  ipsec-vpn-rules vpn-to-cisco;
}
[...]
```

```
ipsec-vpn {
  rule vpn-to-cisco {
    term gre-tunnel {
      from {
        source-address {
          172.17.37.4/32;
        }
        destination-address {
          172.17.38.4/32;
        }
      }
      then {
        remote-gateway 172.17.38.4;
        dynamic {
          ike-policy main_mode_ike_policy;
          ipsec-policy dynamic_ipsec_policy;
        }
      }
    }
  }
  match-direction output;
}
ipsec {
  proposal cisco_compat {
    protocol esp;
    authentication-algorithm hmac-md5-96;
    encryption-algorithm des-cbc;
  }
  policy dynamic_ipsec_policy {
    perfect-forward-secrecy {
      keys group1;
    }
    proposals cisco_compat;
  }
}
[...]
```



### IPSec-over-GRE Tunnel Example: Part 2

The slide continues the example.



## IPSec-over-GRE Tunnel Example (3 of 3)

```
[edit]
lab@R1# show services
[...]
ike {
  proposal cisco-compat {
    authentication-method pre-shared-keys;
    authentication-algorithm md5;
    dh-group group1;
    encryption-algorithm des-cbc;
  }
  policy main mode ike policy {
    proposals cisco-compat;
    pre-shared-key ascii-text "$9$hEycR8-
ds4JDwY"; ## SECRET-DATA
  }
  establish-tunnels immediately;
}

Cisco IOS configuration:

crypto isakmp policy 1
  hash md5
  authentication pre-share
  crypto isakmp key test address 172.17.37.4
  crypto isakmp keepalive 10 2 periodic
!
crypto ipsec transform-set esp_des_set esp-des esp-md5-hmac
!
crypto map gre-to-juniper 1 ipsec-isakmp
  set peer 172.17.37.4
  set transform-set esp_des_set
  set pfs group1
  match address 110

access-list 110 permit ip host 172.17.38.4 host 172.17.37.4

interface tunnel1
  ip address 192.168.25.130 255.255.255.252
  tunnel mode gre ip
  tunnel destination 172.17.37.4
  tunnel source 172.17.38.4

interface fast0
  crypto map gre-to-juniper
```



### IPSec-over-GRE Tunnel Example: Part 3

The slide continues the example and also shows some relevant portions of the configuration from the Cisco router. (Traffic is encrypted as it leaves the Fast Ethernet 0/0 interface on the Cisco router.)

If you want to use a next-hop-style service set instead of an interface-style service set, you must configure filter-based forwarding. Although we do not cover filter-based forwarding in this class, this example demonstrates it. To use a next-hop-style service set to configure the JUNOS router in the example, you could use the following alternate configuration (we do not repeat certain unchanged portions of the IPSec configuration):

```
[edit]
lab@R1# show interfaces
gr-0/0/0 {
  unit 0 {
    tunnel {
      source 172.17.37.4;
      destination 172.17.38.4;
    }
    family inet {
      address 192.168.25.129/30;
    }
  }
}
```

*Continued on next page.*



## IPSec-over-GRE Tunnel Example (contd.)

```

sp-0/0/0 {
  unit 0 {
    family inet;
  }
  unit 1 {
    family inet;
    service-domain outside;
  }
  unit 2 {
    family inet;
    service-domain inside;
  }
}
fe-2/0/1 {
  unit 0 {
    family inet {
      filter {
        input shunt-gre-to-vpninstance;
      }
      address 172.17.37.4/29;
    }
  }
}

```

```

[edit]
lab@R1# show routing-options
interface-routes {
  rib-group inet intf-routes;
}
rib-groups {
  intf-routes {
    import-rib [ inet.0 vpn-test.inet.0 ];
  }
}

```

```

[edit]
lab@R1# show firewall family inet
filter shunt-gre-to-vpninstance {
  term gre {
    from {
      source-address {
        172.17.37.4/32;
      }
      destination-address {
        172.17.38.4/32;
      }
      protocol gre;
    }
    then routing-instance vpn-test;
  }
}

```

*Continued on next page.*

## IPSec-over-GRE Tunnel Example (contd.)

```

    term default {
        then accept;
    }
}

[edit]
lab@R1# show routing-instances
vpn-test {
    instance-type forwarding;
    routing-options {
        static {
            route 0.0.0.0/0 next-hop sp-0/0/0.2;
        }
    }
}

[edit]
lab@R1# show services
service-set gre-vpn {
    next-hop-service {
        inside-service-interface sp-0/0/0.2;
        outside-service-interface sp-0/0/0.1;
    }
    ipsec-vpn-options {
        local-gateway 172.17.37.4;
    }
    ipsec-vpn-rules vpn-to-cisco;
}
ipsec-vpn {
    rule vpn-to-cisco {
        term gre-tunnel {
            from {
                source-address {
                    172.17.37.4/32;
                }
                destination-address {
                    172.17.38.4/32;
                }
            }
            then {
                remote-gateway 172.17.38.4;
                dynamic {
                    ike-policy main_mode_ike_policy;
                    ipsec-policy dynamic_ipsec_policy;
                }
            }
        }
    }
    match-direction input;
}
[...]
```

## Agenda: IPSec VPNs

---

- IPSec VPN Overview
- IPSec VPN Configuration
- Implementation Considerations
- Monitoring



### Monitoring

The slide highlights the topic we discuss next.

## AS PIC IPSec VPN Monitoring

- Use `show services ipsec-vpn` hierarchy
  - Do not use `show ike` or `show ipsec` commands!

```

lab@R1> show services ipsec-vpn ?
Possible completions:
certificates      Show the certificates in cache
ike               Show Internet Key Exchange information
ipsec             Show IPSec information
lab@R1> show services ipsec-vpn ike ?
Possible completions:
security-associations Show services IKE security association information
lab@R1> show services ipsec-vpn ipsec ?
Possible completions:
security-associations Show IPSec security association information
statistics            Show IPSec statistics
lab@R1> show services ipsec-vpn ipsec security-associations
Service set: gre-vpn

Rule: vpn-to-cisco, Term: gre-tunnel, Tunnel index: 1
Local gateway: 172.17.37.4, Remote gateway: 172.17.38.4
Tunnel MTU: 1500
  Direction SPI      AUX-SPI  Mode   Type   Protocol
  inbound  1627908611    0      tunnel dynamic  ESP
  outbound 1109876153    0      tunnel dynamic  ESP

lab@R1> show ipsec security-associations
lab@R1>

```



## AS PIC IPSec VPN Monitoring

As we explained on page 8-8, the JUNOS software contains two different ways to configure and monitor IPSec implementations. You monitor IPSec and IKE operation on the ES PIC using the `show ike` and `show ipsec` commands. However, you monitor IPSec and IKE operation on the AS PIC (including J-series) using the `show services ipsec-vpn` command.

The slide shows that you can monitor AS PIC IPSec security associations using the `show services ipsec-vpn` hierarchy, but not the `show ipsec` command.

## Monitoring IKE

- Monitor IKE status with **show services ipsec-vpn ike security-associations**

```
lab@R1> show services ipsec-vpn ike security-associations
Remote Address  State      Initiator cookie  Responder cookie  Exchange type
172.17.38.4     Matured            64a26d33281b145f  e18f34870388bcf3  Main

lab@R1> show services ipsec-vpn ike security-associations detail
IKE peer 172.17.38.4
  Role: Initiator, State: Matured
  Initiator cookie: 64a26d33281b145f, Responder cookie: e18f34870388bcf3
  Exchange type: Main, Authentication method: Pre-shared-keys
  Local: 172.17.37.4:500, Remote: 172.17.38.4:500
  Lifetime: Expires in 3574 seconds
  Algorithms:
    Authentication      : sha1
    Encryption          : 3des-cbc
    Pseudo random function: hmac-sha1
  Traffic statistics:
    Input bytes      : 1068
    Output bytes    : 1356
    Input packets   : 5
    Output packets  : 11
  Flags: Caller notification sent
  IPSec security associations: 1 created, 0 deleted
  Phase 2 negotiations in progress: 1

  Negotiation type: Quick mode, Role: Responder, Message ID: 2297889794
  Local: 172.17.37.4:500, Remote: 172.17.38.4:500
  Local identity: ipv4(any:0, [0..3]=172.17.37.4)
  Remote identity: ipv4(any:0, [0..3]=172.17.38.4)
  Flags: Caller notification sent, Waiting for done
```

IKE Properties

IPSec SA  
Negotiation  
Information



### Monitoring IKE

You monitor the status of IKE sessions using the **show services ipsec-vpn ike security-associations** command. If you issue the command without arguments, it lists all IKE sessions and their status, one per line. If you use the **detail** argument, the router displays the properties used to secure the IKE session (not the IPsec tunnels themselves), and it displays the status of pending Phase 2 negotiations (the negotiation of the IPSec SAs themselves).

## Troubleshooting IKE (1 of 3)

- Having no active IKE SAs might be acceptable
  - JUNOS software only establishes IKE SAs when needed for IPSec tunnel negotiation
  - Default IPSec SA timeout is 28,800 seconds
  - Default IKE session timeout is 3600 seconds
  - If in doubt, clear an IPSec SA to force the router to reestablish an IKE session

```
lab@R1> show services ipsec-vpn ike security-associations
lab@R1> clear services ipsec-vpn ipsec security-associations gre-vpn
lab@R1> show services ipsec-vpn ike security-associations
Remote Address  State      Initiator cookie  Responder cookie  Exchange type
172.17.38.4     Matured          64a26d33281b145f e18f34870388bcf3  Main
```



### Active IKE SAs

The JUNOS software establishes an IKE SA when it must establish an IPSec SA. The router maintains an expiry timer for both IPSec SAs and IKE SAs. Once the IKE SA expires, the router creates a new IKE SA when it must next create an IPSec SA. However, it does not create a new IKE SA until it must establish an IPSec SA.

The default IPSec SA timeout is 28,800 seconds, while the default IKE timeout is 3600 seconds. Because of these timeouts, a stable network using the default timers has no active IKE SAs for approximately 7 hours. In this case, the lack of an IKE SA is completely normal.

Note that because of the way IKE works on JUNOS routers using the default parameters, it is possible to make a configuration change that blocks IKE communication and to not detect a problem until the next time an IPSec SA expires and the router must use IKE to reestablish the IPSec SA. This scenario might occur hours after the change. For this reason, it is wise to verify proper IKE operation after making changes that might block IKE traffic. (You should verify proper operation even if an active IKE session is listed as *Matured*.)

If you make a configuration change and you want to ensure that the endpoints can still communicate using IKE, you can clear the IPSec SAs. Clearing these SAs forces the router to attempt to reestablish an IKE session to negotiate the IPSec SAs.

## Troubleshooting IKE (2 of 3)

### ■ One matured IKE SA per peer

- Initiator and responder cookies should match on both sides
- Nonmatured SAs are still negotiating
- Example: One-way communication, IKE traffic from local to remote is blocked

```
lab@R1> show services ipsec-vpn ike security-associations
Remote Address  State      Initiator cookie  Responder cookie  Exchange type
172.17.38.4     Not matured      c2276b1df9df53ae 0000000000000000  Main
172.17.38.4     Not matured      45cffe5d160eb8ba 5c078bbbee4ebe9f  Main
```

```
lab@R1> show services ipsec-vpn ike security-associations detail
```

```
IKE peer 172.17.38.4
Role: Initiator, State: Not matured
Initiator cookie: c2276b1df9df53ae, Responder cookie: 0000000000000000
Exchange type: Main, Authentication method: Pre-shared-keys
[...]
Traffic statistics:
Input bytes : 0
Output bytes : 432
Input packets: 0
Output packets: 4
[...]

IKE peer 172.17.38.4
Role: Responder, State: Not matured
Initiator cookie: 45cffe5d160eb8ba, Responder cookie: 5c078bbbee4ebe9f
[...]
```

No response to our connection attempts

We responded to the peers connection, but the session is not mature



### Mature IKE SAs

When you examine properly established IKE SAs, they should be listed in a Matured state, and both endpoints should list the same initiator and responder cookies. The router considers IKE SAs to be mature when it has completed IKE Phase 1 negotiations with its peers. SAs that are listed as Not matured are not fully established.

The slide shows an example of an IKE session that failed to fully establish due to a one-way communication failure. In the example on the slide, a firewall filter is blocking IKE traffic from the router to the remote peer. The router is attempting to initiate an IKE session, but it is receiving no response from the remote side. Additionally, the router is attempting to respond to an IKE session initiated by the remote side, but the session is not fully establishing because the remote peer is not receiving the router's responses.

## Troubleshooting IKE (3 of 3)

### ■ You can enable IKE trace options

- Set under [edit services ipsec-vpn]

```
lab@R1# set traceoptions flag ike
```

- Monitor in kmd log (note retransmissions)

```
lab@R1> show log kmd | last 100
Oct 25 16:57:01 ike_policy_reply_isakmp_vendor_ids: Start
Oct 25 16:57:01 ike_st_o_private: Start
Oct 25 16:57:01 ike_encode_packet: Start, SA = { 0x19b46b53 d4611fe2 - 5edd463c f69aa98c } / 00000000,
nego = -1
Oct 25 16:57:01 ike_send_packet: Start, send SA = { 19b46b53 d4611fe2 - 5edd463c f69aa98c }, nego = -1,
src = 172.17.37.4:500, dst = 172.17.38.4:500
Oct 25 16:57:06 ike_retransmit_callback: Start, retransmit SA = { 19b46b53 d4611fe2 - 5edd463c
f69aa98c }, nego = -1
Oct 25 16:57:06 ike_send_packet: Start, retransmit previous packet SA = { 19b46b53 d4611fe2 - 5edd463c
f69aa98c }, nego = -1, src = 172.17.37.4:500, dst = 172.17.38.4:500
Oct 25 16:57:06 ike_udp_callback: Packet ready in source :
Oct 25 16:57:06 ike_get_sa: Start, SA = { 19b46b53 d4611fe2 - 00000000 00000000 } / 00000000, remote =
172.17.38.4:500
Oct 25 16:57:16 ike_retransmit_callback: Start, retransmit SA = { 19b46b53 d4611fe2 - 5edd463c
f69aa98c }, nego = -1
Oct 25 16:57:16 ike_send_packet: Start, retransmit previous packet SA = { 19b46b53 d4611fe2 - 5edd463c
f69aa98c }, nego = -1, src = 172.17.37.4:500, dst = 172.17.38.4:500
Oct 25 16:57:16 ike_udp_callback: Packet ready in source :
Oct 25 16:57:16 ike_get_sa: Start, SA = { 19b46b53 d4611fe2 - 00000000 00000000 } / 00000000, remote =
172.17.38.4:500
Oct 25 16:57:26 ike_retransmit_callback: Start, retransmit SA = { 19b46b53 d4611fe2 - 5edd463c
f69aa98c }, nego = -1
[...]
```



### IKE Traceoptions

You can log information about IKE operation by enabling traceoptions, as shown on the slide. By default, the log messages go to the kmd log. As you can see from the output, the router is reporting a number of retransmissions, further increasing suspicion of a one-way communication failure.

Troubleshoot this scenario the same way you normally troubleshoot a connectivity issue, first checking routing, and then checking access controls along the path.



## Monitoring IPsec SAs

- Monitor IPsec status with `show services ipsec-vpn ipsec security-associations`

```
lab@R1> show services ipsec-vpn ipsec security-associations
Service set: gre-vpn
```

```
Rule: sample-vpn, Term: gre-tunnel, Tunnel index: 1
Local gateway: 172.17.37.4, Remote gateway: 172.17.38.4
Tunnel MTU: 1500
Direction SPI      AUX-SPI      Mode      Type      Protocol
inbound  2065803868  0          tunnel    dynamic   ESP
outbound  681969259   0          tunnel    dynamic   ESP
```

```
lab@R1> show services ipsec-vpn ipsec security-associations extensive
Service set: gre-vpn
```

```
Rule: sample-vpn, Term: gre-tunnel, Tunnel index: 1
Local gateway: 172.17.37.4, Remote gateway: 172.17.38.4
Tunnel MTU: 1500
Local identity: ipv4(any:0,[0..3]=172.17.37.4)
Remote identity: ipv4(any:0,[0..3]=172.17.38.4)

Direction: inbound, SPI: 2065803868, AUX-SPI: 0
Mode: tunnel, Type: dynamic, State: Installed
Protocol: ESP, Authentication: hmac-shal-96, Encryption: 3des-cbc
Soft lifetime: Expires in 28682 seconds
Hard lifetime: Expires in 28772 seconds
Anti-replay service: Enabled, Replay window size: 64

Direction: outbound, SPI: 681969259, AUX-SPI: 0
Mode: tunnel, Type: dynamic, State: Installed
Protocol: ESP, Authentication: hmac-shal-96, Encryption: 3des-cbc
Soft lifetime: Expires in 28682 seconds
Hard lifetime: Expires in 28772 seconds
Anti-replay service: Enabled, Replay window size: 64
```

Secured  
addresses

Bidirectional  
tunnels with  
security  
parameters



### Monitoring IPsec SAs

You can monitor the status of IPsec SAs with the `show services ipsec-vpn ipsec security-associations` command.

The **extensive** output shows the creation of bidirectional tunnels for each set of address pairs between which communication is secured.

## Monitoring IPSec Traffic Statistics

- Monitor IPSec traffic statistics with `show services ipsec-vpn ipsec statistics`

- By default, statistics are per service set

```
lab@R1> show services ipsec-vpn ipsec statistics
PIC: sp-0/0/0, Service set: gre-vpn

ESP Statistics:
  Encrypted bytes:      620880
  Decrypted bytes:      777200
  Encrypted packets:    5499
  Decrypted packets:    5669
AH Statistics:
  Input bytes:          0
  Output bytes:         0
  Input packets:        0
  Output packets:       0
Errors:
  AH authentication failures: 0, Replay errors: 0
  ESP authentication failures: 0, ESP decryption failures: 0
  Bad headers: 0, Bad trailers: 0
```

- Use `detail` keyword for per-tunnel statistics



## Monitoring IPSec Traffic Statistics

After you configure an IPSec tunnel, you should verify that traffic not only still reaches the remote side, but that it is, in fact, secured. You can perform this verification by checking the IPSec traffic statistics to ensure that packets are being secured appropriately. You can see statistics per service set using the `show services ipsec-vpn ipsec statistics` command. If you add the `detail` keyword, you can see per-tunnel statistics. (You can determine the rule and term for which the statistics are being kept by cross-referencing the tunnel IDs in the detail output with the tunnel IDs in the output of the `show services ipsec-vpn ipsec security-associations extensive` command.)

## Review Questions

---

1. What is the purpose of IKE?
2. What is the purpose of IPSec?
3. What are proposals and how are they used?
4. When should you use an IPSec-over-GRE tunnel?
5. Why might there not be an active IKE SA for a configured and operational VPN?



### This Chapter Discussed:

- Configuring an IPSec VPN using the JUNOS software CLI;
- Configuring an IPSec-over-GRE tunnel using the JUNOS software CLI; and
- Monitoring VPN operations.

## Lab 6: IPSec VPNs

---

- Configure a next-hop-style VPN.
- Configure an IPSec-over-GRE interface-style VPN.
- Explore stateful firewall, NAT, and IPSec VPN interaction.



### Lab 6: IPSec VPNs

The slide highlights the objectives for this lab.



# **Advanced Juniper Networks Routing in the Enterprise**

## **Chapter 9: Class of Service**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - Explain the way CoS is processed on the J-series and M-series routers
  - Implement appropriate CoS policies given a traffic-prioritization design



### This Chapter Discusses:

- The way class of service (CoS) is processed on the J-series and M-series routers; and
- Implementing appropriate CoS policies given a traffic prioritization design.

## Agenda: Class of Service

---

- CoS Overview
- Traffic Classification
- Traffic Queueing
- Traffic Scheduling
- Example
- Troubleshooting



### CoS Overview

The slide lists the topics we discuss in this chapter. We discuss the highlighted topic first.

## What Is Class of Service?

- CoS is designed to:
  - Provide mechanisms for categorizing traffic
  - Allow the router to use those categorizations to meet performance requirements
- CoS is not designed to:
  - Make a network faster
  - Reduce congestion



### Uses of Class of Service

By default, routers treat all traffic transiting the router equally. All traffic entering the router is handled on a first-come, first-served basis. The router mixes together all traffic transiting the router and places it in the same input and output queues, which means the traffic is subject to the same potential for delays and drops. This method is called *best-effort* traffic processing.

The class-of-service (CoS) features of Juniper Networks routers are designed to allow the router to provide differentiated services to network traffic where best-effort traffic processing is insufficient. There are several components to the CoS toolkit. First, tools exist that allow you to configure the router to place traffic into different categories (called *forwarding classes*) that should receive the same services from the router. Second, components exist that allow you to specify the way the router should treat traffic in each forwarding class. Finally, tools exist that allow you to have the router mark packets with their category so that other devices in the network know how to categorize them.

*Continued on next page.*



## CoS Does Not Make A Network Faster

CoS allows you to treat traffic differently by providing a minimum bandwidth guarantee, low latency, low packet loss, or a combination of these things for categories of traffic. Consequently, deploying CoS can make some applications perform better. However, it cannot increase the total bandwidth of a link or decrease latency beyond the minimum limits imposed by the speed of light. So, in a network that is congested, CoS can manage the congestion by providing minimum bandwidth guarantees, decreasing latency, minimizing packet loss for certain traffic classes, or some combination of these processes. However, congestion still results in traffic being delayed or dropped. The CoS tools just help you control how different types of traffic are effected by this congestion.

## Typical Uses of CoS

- **Prioritization**
  - VoIP
  - Latency-sensitive traffic
  - Priority users
- **Congestion management**
  - Congestion avoidance using RED
  - Congestion control to ensure SLA maintenance
- **Bandwidth control**
  - Ensure that different classes of traffic receive allocated bandwidth



### CoS Allows Traffic Prioritization

You can use CoS to control the order in which traffic is forwarded through the router. Even in networks that are not fully utilizing the capacity on all their network links, there can be instantaneous contention for transmission to the wire. If two packets arrive at an output interface at the same time, the router will transmit one packet and queue the other. The delay in the queue might be minimal in a generally uncongested network; however, even a brief delay can be significant for latency-sensitive traffic (such as voice over IP [VoIP]). You can use traffic prioritization to ensure that latency-sensitive traffic is transmitted before other traffic. Prioritization also controls the way that extra bandwidth is allocated after all bandwidth guarantees have been met.

### Congestion Management

The router allows you to use the random early discard (RED) algorithm to avoid congestion. The router uses the RED algorithm to selectively drop random packets before congestion becomes critical. This should cause affected TCP sessions to go into slow-start mode, reducing the total amount of traffic clients are attempting to transmit through the congested link. Because of the algorithm used, higher-bandwidth data streams are the most likely to be affected, while lower-bandwidth streams (including most interactive sessions) are the least likely to be affected.

*Continued on next page.*

## Bandwidth Control

You can also configure the router to guarantee certain levels of bandwidth to traffic classes. This configuration ensures that minimum bandwidth guarantees are met during periods of congestion. In this way, it is possible to ensure that certain types of traffic receive a guaranteed minimum level of bandwidth on each link.

## CoS Terminology

- **Forwarding classes**
  - Identifies traffic that should receive common treatment
  - Used to assign traffic to output queues
- **Loss priority**
  - Identifies the priority the router should give to dropping a packet
  - Used to select the drop profile used in the RED process



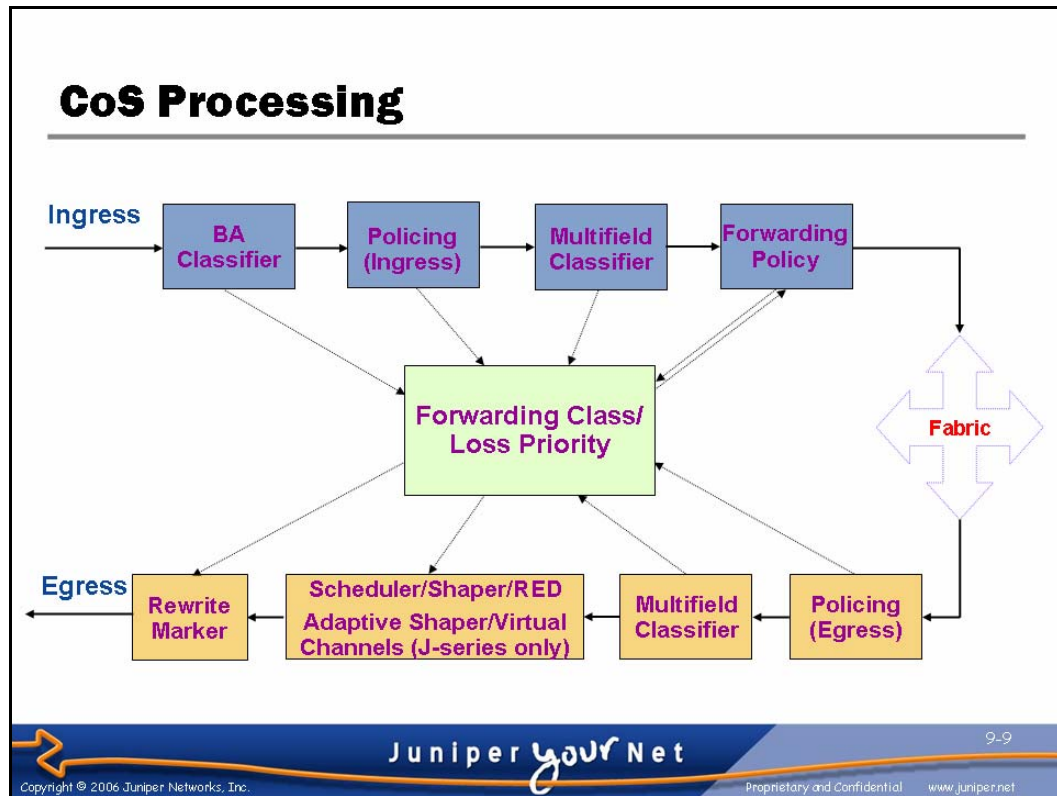
### Forwarding Classes

A forwarding class is an abstract concept the router uses to identify traffic that should receive common treatment. The router associates traffic with a forwarding class during the classification process. On output, the router assigns traffic to a particular output queue based on forwarding class and rewrites behavior aggregate markers based on forwarding class.

It is tempting, and sometimes helpful, to think of forwarding classes as output queues because there is usually a one-to-one mapping of forwarding classes and output queues. However, that is not necessarily true, as some platforms support more forwarding classes than queues. In these cases, combining the concepts of forwarding classes and output queues can be confusing.

### Loss Priority

You can associate a loss priority with a packet to tell the router what priority it should give to dropping this packet during congestion. If you choose to configure RED, you can choose different drop probabilities for traffic with different loss priorities. Configuring RED in this way is beyond the scope of this class.



### CoS Processing

The chart on the slide provides an overview of the way that CoS processing occurs in the JUNOS software. The arrowheads indicate the flow of information. Thus, if a process sets the forwarding class and loss priority, the arrow points towards the forwarding class/loss priority box. If the forwarding class and loss priority are used as inputs to a process, the arrow points away from the forwarding class/loss priority box.

You can configure the router to set the forwarding class and loss priority for a given packet based on the values of certain header fields, including DiffServ code point (DSCP), IP precedence, MPLS EXP, and IEEE 802.1p. These fields are usually set by edge routers and indicate the category of traffic. In this way, a core router does not need to recategorize traffic it receives from edge routers. Because these fields indicate the category of traffic to which the packet belongs, this kind of classification is sometimes called a *behavior aggregate* (BA) classifier.

You can also configure the router to set the forwarding class and loss priority with multifield classifiers on both ingress and egress. The multifield classifier is implemented within a firewall filter. Because multifield classifiers are implemented within firewall filters, you can select packets using all the selection criteria of a firewall filter and set the forwarding class and loss priority within the *then* clause. You can also use ingress and egress policers to modify the forwarding class and loss priority.

*Continued on next page.*

## CoS Processing (contd.)

You can use forwarding policy options to implement CoS-based forwarding (CBF). When multiple equal-cost paths to a destination exist, you can use CBF to specify which of the equal-cost paths will be used for different classes of traffic. Additionally, you can use forwarding policy options to reset the forwarding class and loss priority for packets destined for particular prefixes.

The router uses the forwarding class and loss priority on egress to assign traffic to queues, implement the RED algorithm, and rewrite BA headers.

On the J-series platform, CoS functions are performed in software and their availability and limitations are not dependent on the ingress or egress interface type. On the M-series platform, CoS functions are performed in hardware and have limits that vary dependent on the type of FPC and PIC. Also, some CoS features are only available on the J-series platform (such as virtual channels and adaptive shaping for Frame Relay). See the “Hardware Capabilities and Routing Engine Protocol Queue Assignments” chapter of the *JUNOS Class of Service Configuration Guide*.

## Agenda: Class of Service

---

- CoS Overview
- Traffic Classification
- Traffic Queueing
- Traffic Scheduling
- Example
- Troubleshooting



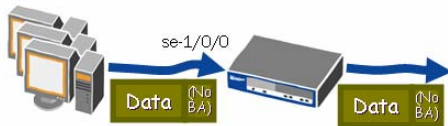
### Traffic Classification

The slide highlights the topic we discuss next.

## Classification Overview

### ■ In-the-box applications

- Multifield classifier
- No BA rewrite



### ■ Across-the-network applications

- Multifield classifier at edge
- BA in core



### In-the-Box Applications

There are two models for CoS deployment. In some networks, the only use for CoS is to provide differentiated forwarding services through a single router. In other networks, the CoS deployment is intended to provide differentiated services across multiple devices within a network.

Where the deployment is limited to a single router, it is often the case that the router classifies packets as they pass through the router using a multifield classifier, uses the classification to provide the configured service level, and then forwards the packets without any BA markings (DSCP, IP precedence, and so forth). However, when the deployment crosses multiple routers, the classification usually occurs differently.

### Across-the-Network Applications

When multiple devices in a network are configured to use CoS to provide differentiated services, it is usually beneficial to have the classification initially performed on the edge routers and then have that classification be transmitted within the network using a BA. In this model, the edge router classifies traffic using a multifield classifier as traffic enters the network. When the edge router transmits the packet into the network, it marks the packets with a BA. The remainder of the routers in the network read the BA and automatically set the correct forwarding class and loss priority without a multifield classifier.

*Continued on next page.*

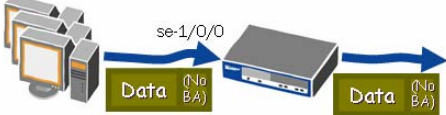


**Across-the-Network Applications (contd.)**

Using BAs in network-wide CoS applications provides several advantages. First, it ensures consistent CoS treatment of the traffic throughout the network. Second, it simplifies the task of creating and maintaining accurate multifield classifiers on each router. Without BAs, each router would need to have a multifield classifier capable of classifying all traffic entering the network and any classification change would need to be replicated on all routers. Third, in the case of Ethernet, setting the 802.1p bits as a BA allows CoS-aware Ethernet switches to also provide differentiated services to the traffic.

Packets are marked with BAs by setting bits that already exist within the Layer 3 header. Therefore, additional overhead is added by setting a Layer 3 BA.

## Multifield Classifiers



```

interfaces {
  se-1/0/0 {
    unit 0 {
      family inet {
        filter {
          input apply-cos-markings;
        }
      }
    }
  }
}

firewall {
  family inet {
    filter apply-cos-markings {
      term admin {
        from {
          source-address {
            192.168.200.0/25;
          }
        }
        then {
          forwarding-class expedited-forwarding;
          accept;
        }
      }
      term all-other-traffic {
        then accept;
      }
    }
  }
}

```

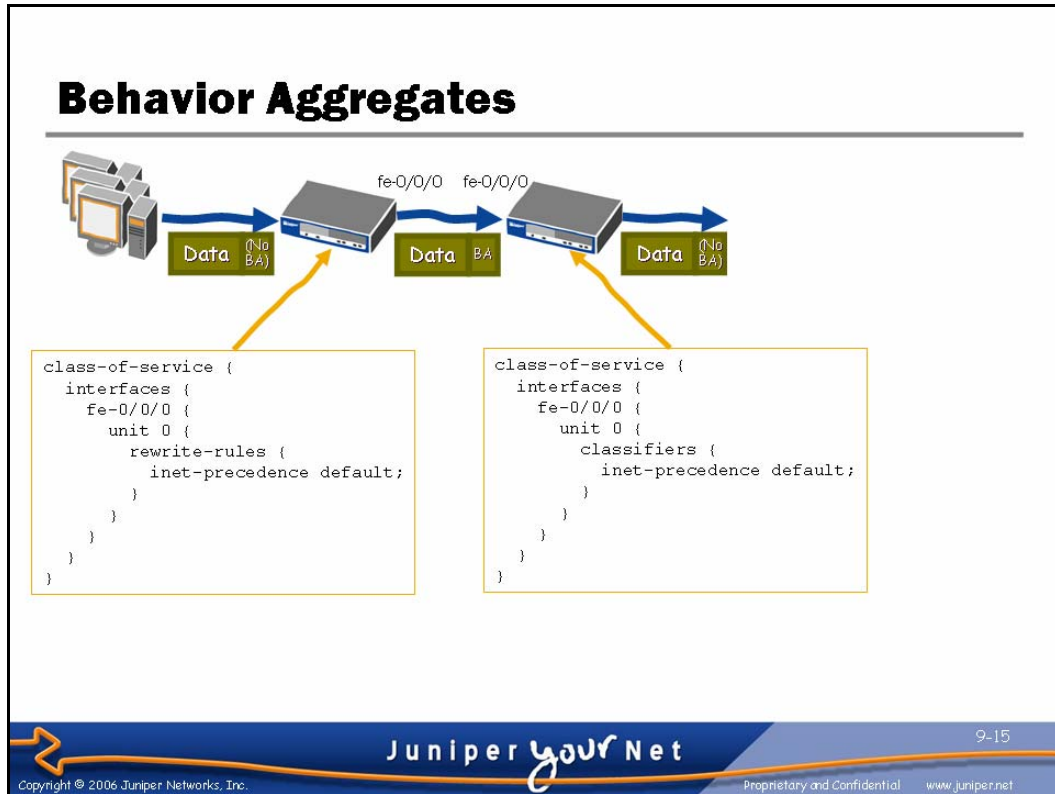
Juniper *your* Net

Copyright © 2006 Juniper Networks, Inc. Proprietary and Confidential www.juniper.net

### Multifield Classifiers

You configure multifield classifiers just like regular firewall filters. You place the forwarding class and loss priority in the *then* clause of each term. The default is to not change the forwarding class (from the forwarding class assigned by the BA classifier or, if the packet was not classified by a BA classifier, the default forwarding class).

Because the router applies multifield classifiers after BA classifiers, multifield classifiers always override the forwarding class and loss priority assigned by the BA.



## Behavior Aggregates

You configure routers to apply BA markers to packets leaving an interface by applying a rewrite rule to the outbound interface. By default, Layer 3 BA header fields are not modified as a packet is forwarded through a router, so it is only necessary to set these fields once, usually on an edge router. However, Layer 2 fields (such as the MPLS EXP and IEEE 802.1p fields) are not preserved, so you must configure the router to reapply Layer 2 BA header fields on every appropriate interface as a packet traverses the network.

You apply rewrite rules under the `[edit class-of-service interfaces]` hierarchy. To apply the default IP precedence markings, for example, type **set interface-name unit logical-unit-number rewrite-rules inet-precedence default**. You can apply rewrite rules at both Layer 2 and Layer 3; however, you cannot apply both IP precedence and DSCP rules to the same packets, because they use the same bits in the IP header.


*Continued on next page.*

## Behavior Aggregates (contd.)

Once a packet is marked with a BA, you can have routers read those markers and automatically assign packets the correct forwarding class and loss priority. To do this, you apply a BA classifier to the inbound interface. You apply BA classifiers to interfaces under the `[edit class-of-service interfaces]` hierarchy. To apply the default IP precedence BA classifier, for example, type **set interface-name unit logical-unit-number classifiers inet-precedence default**. You can apply BA classifiers at both Layer 2 and Layer 3; however, many platform-specific limitations govern the valid combinations of classifiers. See the *JUNOS Class of Service Configuration Guide* for detailed information.

Using the default BA classifiers and rewrite rules ensures that traffic from Queues 0–3 is mapped correctly to the corresponding forwarding classes throughout the network. You must use a custom classifier and rewrite rule if you want to consistently map traffic across the network to more than Queues 0–3 or if you want to use nondefault bit patterns for the CoS header fields. If you apply custom classifiers and rewrite rules, you must apply these rules to all routers within the network to ensure consistent classification.


## Policers



```

firewall {
  policer admin-traffic-policer {
    if-exceeding {
      bandwidth-limit 1m;
      burst-size-limit 3k;
    }
    then forwarding-class best-effort;
  }
}
family inet {
  filter apply-cos-markings {
    term admin {
      from {
        source-address {
          192.168.200.0/25;
        }
      }
      then {
        policer admin-traffic-policer;
        forwarding-class expedited-forwarding;
        accept;
      }
    }
    term all-other-traffic {
      then accept;
    }
  }
}

```


Juniper your Net
9-17

Copyright © 2006 Juniper Networks, Inc.
Proprietary and Confidential [www.juniper.net](http://www.juniper.net)

### Policers

Policers allow you to limit certain kinds of traffic to a specified bandwidth and burst size. You can configure the router to assign a different forwarding class or loss priority to traffic exceeding the configured limits. You can configure these policies like a regular traffic policer, using the `forwarding-class` and `loss-priority` statements in the `then` clause of the policer. The `forwarding-class` and `loss-priority` statements from the policer override any `forwarding-class` and `loss-priority` statements in the firewall filter's `then` statement.

For example, the configuration on the slide causes traffic within the policer's rate limit to be assigned the `expedited-forwarding` class. Traffic that exceeds the rate limiter is assigned the `best-effort` forwarding class.

## Agenda: Class of Service

---

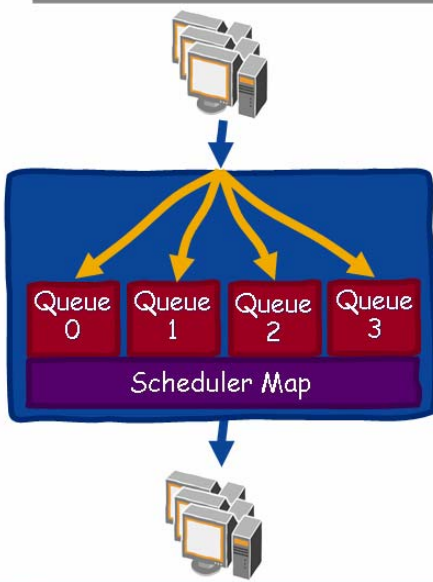
- CoS Overview
- Traffic Classification
- Traffic Queueing
- Traffic Scheduling
- Example
- Troubleshooting



### Traffic Queueing

The slide highlights the topic we discuss next.

## Queueing



The diagram illustrates the queueing process. At the top, a cluster of server icons represents incoming traffic. A blue arrow points from this traffic to a central blue box. Inside this box, four yellow arrows branch out to four red boxes labeled 'Queue 0', 'Queue 1', 'Queue 2', and 'Queue 3'. These four queues are positioned above a purple box labeled 'Scheduler Map'. A blue arrow points from the Scheduler Map to a cluster of server icons at the bottom, representing outgoing traffic.

- Overview:
  - Default of 4 queues (0–3)
  - Forwarding classes map to queues
  - Default queue/forwarding class associations:
    - 0: best-effort
    - 1: expedited-forwarding
    - 2: assured-forwarding
    - 3: network-control

Juniper your Net

Copyright © 2006 Juniper Networks, Inc. 9-19 Proprietary and Confidential www.juniper.net

### Queueing Overview

When traffic reaches the outbound interface, the router places traffic of each forwarding class in its own queue. The number of queues supported on outbound interfaces is hardware dependent. The J-series router supports eight queues.

The router associates each forwarding class with its own queue number. The default forwarding classes and their associated queue numbers are shown on the slide. By default, certain network control traffic (routing protocol messages, keepalives, and so forth) is sent to the Queue 3 (by default, the network-control forwarding class), while all other traffic is sent to Queue 0 (by default, the best-effort queue). Some form of classifier must be configured to send traffic to any other queue.

After the router places traffic in the correct queues, a scheduler defines how the interface should process traffic from each queue; however, the router treats all the traffic in each queue in the same way.

## Forwarding Class Definition

- To associate a forwarding class with a queue:

`set class-of-service forwarding-classes queue number forwarding-class-name`

- Examples:

```
[edit]
lab@London# set class-of-service forwarding-classes queue 4 very-important-data

[edit]
lab@London# set class-of-service forwarding-classes queue 0 unimportant-data

[edit]
lab@London# commit
commit complete

[edit]
lab@London# run show class-of-service forwarding-class
Forwarding class      Queue
unimportant-data      0
expedited-forwarding  1
assured-forwarding    2
network-control       3
very-important-data   4
```



### Forwarding Class Definition

You create forwarding classes by associating them with queues in the `[edit class-of-service forwarding-classes]` hierarchy. If you attempt to define forwarding classes for more queues than your hardware can support, the router displays an error message when you attempt to commit the configuration. You must associate a forwarding class name with a queue to use that queue. Default names are associated with Queues 0–3.

As the examples on the slide show, not only can you assign forwarding class names to non default queues, but you can also change the forwarding class names associated with the default queues. Even if you rename the forwarding classes associated with Queues 0 and 3, traffic is still sent to those queues. Additionally, the default BA classifiers and rewrite rules map traffic to and from Queues 0–3, regardless of the forwarding class names associated with those queues.

Forwarding classes are important because all other CoS rules reference forwarding classes, rather than queues. The mapping between forwarding class names and queue numbers is maintained only in the forwarding class definitions in the `[edit class-of-service forwarding-classes]` hierarchy.



## Agenda: Class of Service

---

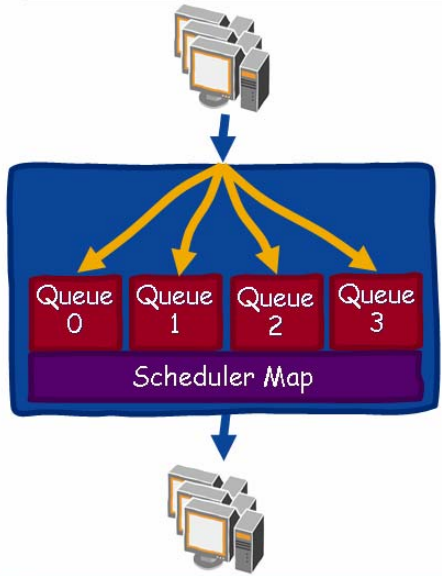
- CoS Overview
- Traffic Classification
- Traffic Queueing
- Traffic Scheduling
- Example
- Troubleshooting




### Traffic Scheduling

The slide highlights the topic we discuss next.

## Scheduling Overview



- Several components to scheduling:
  - Priority
  - Transmission rate
  - Buffer size
  - RED configuration
- Priority and transmission rate define the order
- Buffer size and RED configuration define how packets are stored and dropped



Copyright © 2006 Juniper Networks, Inc.

Proprietary and Confidential [www.juniper.net](http://www.juniper.net)

### Scheduling Overview

You control the way the router services queues by configuring schedulers and scheduler maps. A scheduler contains parameters that describe how a queue should be serviced. A scheduler is associated with a particular queue through a scheduler map.

### Priority and Transmission Rate

You define the order in which packets should be transmitted by defining a priority and a transmission rate for a queue. On M-series routers, the router always transmits packets from higher-priority queues before transmitting packets from lower-priority queues, except when the higher-priority queue is exceeding its configured transmission rate. On J-series routers, the router always transmits packets from higher-priority queues before transmitting packets from lower-priority queues, even if the higher-priority queue is exceeding its configured transmission rate. By default, all queues are low priority.

On M-series routers, you can configure a transmission rate for each queue, which controls how much bandwidth traffic from that queue can consume. By default, 95% of the bandwidth is assigned to Queue 0 and 5% of the bandwidth is assigned to Queue 3. All other queues are assigned a 0 transmission rate. By default, all queues can exceed their assigned transmission rate if other queues are not fully utilizing their assigned rates, unless you configure the transmission rate with the **exact** option.

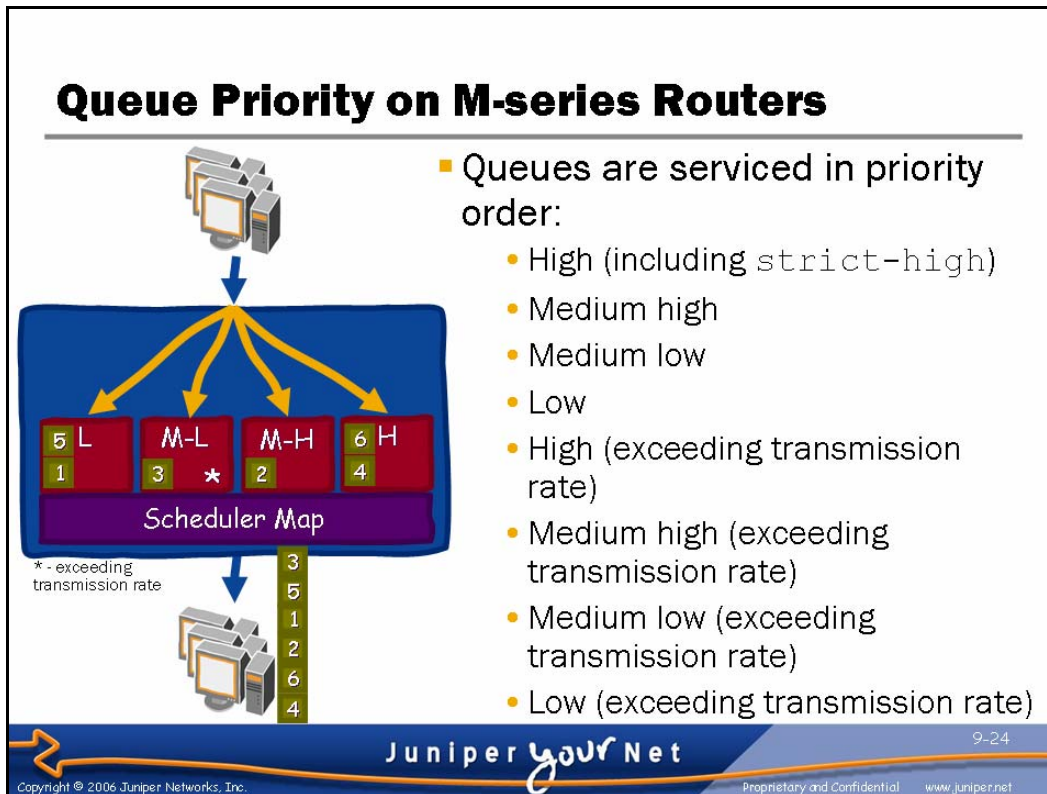
*Continued on next page.*

## Priority and Transmission Rate (contd.)

On J-series routers, you can configure a transmission rate for each queue, but that transmission rate only affects the way traffic is transmitted from queues of the same priority level. By default, 95 per cent of the bandwidth is assigned to Queue 0 and 5 per cent of the bandwidth is assigned to Queue 3. All other queues are assigned a 0 transmission rate. All queues can exceed their assigned transmission rate. If you configure the transmission rate with the **exact** option, that queue will not be able to exceed the assigned transmission rate while other queues of equal or greater priority have traffic waiting for transmission; however, that queue will be able to exceed the assigned transmission rate while other lower priority queues have traffic waiting for transmission. In this way, it is possible for a higher-priority queue to starve lower-priority queues on J-series routers.

## Buffer Size and RED

The size of each queue is determined by the configured buffer size. By default, Queue 0 is assigned 95 percent of the available buffer space, while Queue 3 is assigned 5 percent of the available buffer space. By default, all other queues have 0 percent of the available buffer space; therefore, if you use queues other than 0 and 3, you should assign buffers to those queues. As the buffer fills, packets become more likely to be dropped by the RED algorithm. You can specify an exact RED drop profile for each queue; however, the default configuration is to not drop packets until the queue is full, and then to drop packets when the queue is full.



### Queue Priority on M-series Routers

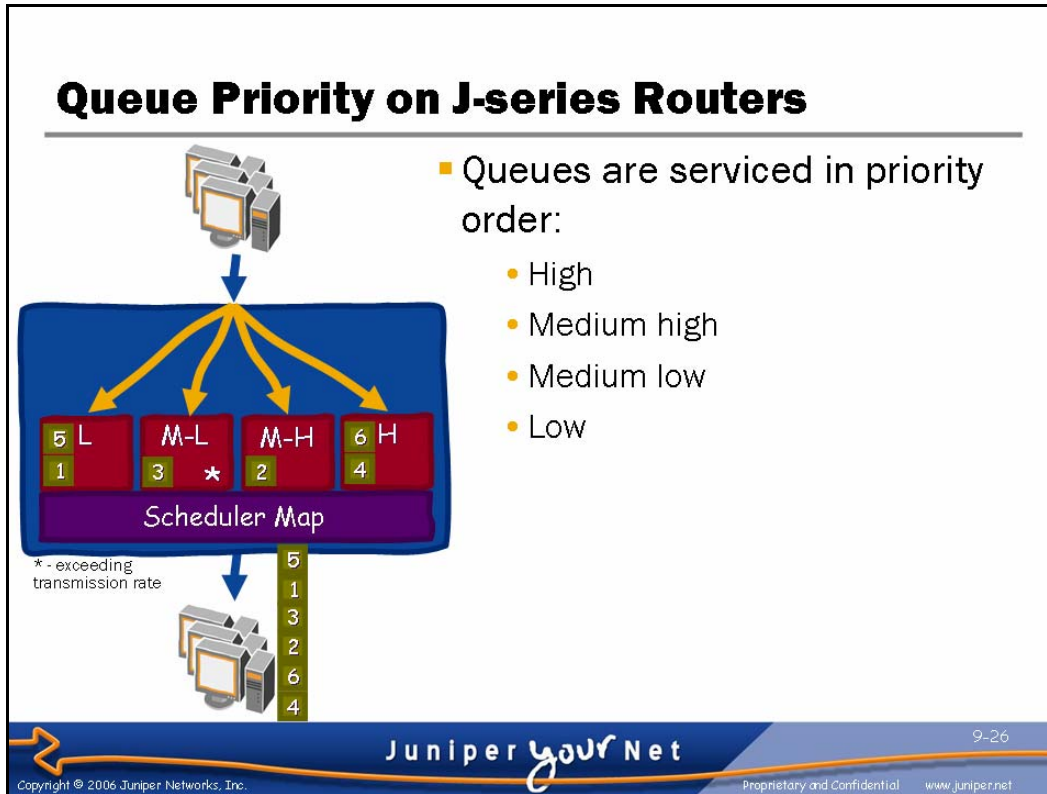
This slide shows how the M-series routers use queue priority and transmission rate to determine the order in which to transmit packets. In this example, assume the packets arrived in the order indicated by their number. In the default CoS configuration, all the packets would have arrived in Queue 0 and would have been transmitted in the order they arrived (first 1, then 2, 3, 4, 5, and finally 6). However, because the packets were assigned to different forwarding classes by a classifier, they were placed in different output queues. And, because the schedulers assigned to those queues indicated different priorities and transmission rates, the packets were actually transmitted in a very different order.

First, the router serviced the high-priority queue, transmitting those packets. Then, it transmitted the packet from the medium-high queue. Next, it transmitted the packets from the low queue. And, finally, it transmitted the packet from the medium-low queue. The router transmitted the packets in the low queue before those in the medium-low queue because the medium-low queue was exceeding its configured transmission rate.

*Continued on next page.*

### Queue Priority (contd.)

Note that this example assumes that the output interface was busy when these six packets arrived (so they all had to be queued) and that no new packets were placed in these queues while these six packets were being transmitted. Reality is usually more complicated than this example, as packets are constantly arriving and being placed in queues. New traffic arriving in the queues would change the transmission order. For example, If a packet arrived in the high-priority queue while Packet 1 was being transmitted, and the high-priority queue was within its configured transmission rate, the router would transmit the packet from the high-priority queue prior to transmitting Packet 5.



### Queue Priority on J-series Routers

This slide illustrates how the J-series routers use queue priority to determine the order in which to transmit packets. Unlike M-series routers, J-series routers do not consider transmission rate when determining how to schedule packets from queues of different priorities. In this example, this fact results in the router transmitting the packet from the medium-low priority queue (Packet 3) before it transmits the packets from the low priority queue (Packets 1 and 5).

## Scheduler Definition

- Configure schedulers under `[edit class-of-service schedulers]`

Example:

```
[edit class-of-service schedulers]
lab@HongKong# set sched-best-effort transmit-rate percent 40

[edit class-of-service schedulers]
lab@HongKong# set sched-best-effort buffer-size percent 40

[edit class-of-service schedulers]
lab@HongKong# set sched-best-effort priority low

[edit class-of-service schedulers]
lab@HongKong# show
sched-best-effort {
    transmit-rate percent 40;
    buffer-size percent 40;
    priority low;
}
```



Juniper your Net

9-27

Copyright © 2006 Juniper Networks, Inc.

Proprietary and Confidential www.juniper.net

## Configuring Schedulers

You configure schedulers under the `[edit class-of-service schedulers]` hierarchy. The scheduler's name has no special significance. The router associates it with a particular queue in a scheduler map. In this example, the scheduler is intended to be used for traffic in the best-effort forwarding class; therefore, we named it *sched-best-effort* to remind us of its purpose; however, we could have named it anything we wanted and still associated it with the best-effort forwarding class in the scheduler map.

In this example, we configured the buffer size to be the same as the transmit rate. This is a good practice that should perform acceptably under many circumstances; however, there are cases where different buffer sizes are appropriate. You can find more details in the *JUNOS Class of Service Configuration Guide*.

Also, we did not configure a custom drop profile in this example. The default drop profile works well in many circumstances. Adjusting RED parameters by creating custom drop profiles can be quite complex. More details on that topic are also available in the *JUNOS Class of Service Configuration Guide*.

## Scheduler Map Definition

- Scheduler maps associate schedulers with queues
  - Configured under `[edit class-of-service scheduler-maps]`

### Example:

```
[edit class-of-service scheduler-maps]
lab@HongKong# set class-example forwarding-class best-effort scheduler sched-best-effort

[edit class-of-service scheduler-maps]
lab@HongKong# set class-example forwarding-class network-control scheduler sched-network-control

[edit class-of-service scheduler-maps]
lab@HongKong# set class-example forwarding-class admin scheduler sched-admin

[edit class-of-service scheduler-maps]
lab@HongKong# show
class-example {
    forwarding-class best-effort scheduler sched-best-effort;
    forwarding-class network-control scheduler sched-network-control;
    forwarding-class admin scheduler sched-admin;
}
```



## Creating Scheduler Maps

Scheduler maps associate schedulers with particular queues. The queues are referenced by the name of the forwarding class that is associated with them. You configure scheduler maps under the `[edit class-of-service scheduler-maps]` hierarchy.

In the example on the slide, we associate three schedulers with three different queues. (The admin forwarding class is associated with a particular queue in the `[edit class-of-service forwarding-classes]` hierarchy.)



## Applying a Scheduler Map to an Interface

- Scheduler maps must be associated with outbound interfaces under `[edit class-of-service interfaces]`
  - Wildcards are allowed for interface names and units
  - Per-unit scheduling requires `per-unit-scheduler` in the interface configuration under `[edit interfaces]`

### Example:

```
[edit class-of-service interfaces]
lab@HongKong# set se-1/0/0 scheduler-map class-example

[edit class-of-service interfaces]
lab@HongKong# set fe-* scheduler-map class-example

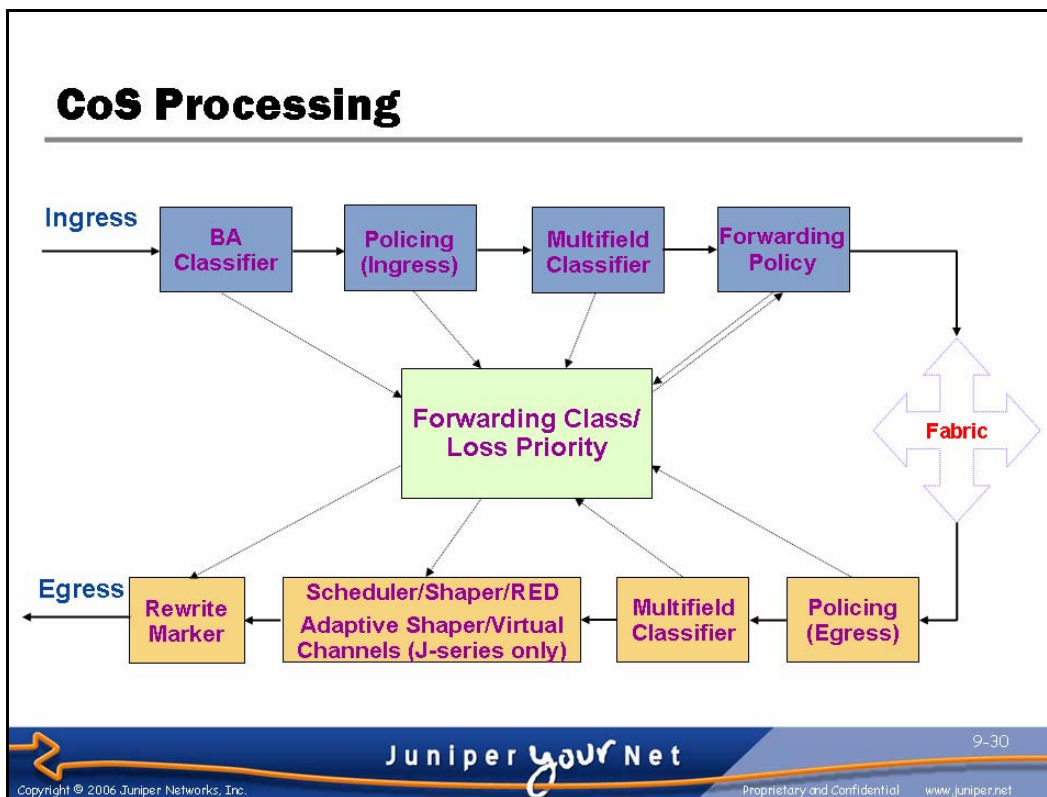
[edit class-of-service interfaces]
lab@HongKong# show
se-1/0/0 {
    scheduler-map class-example;
}
fe-* {
    scheduler-map class-example;
}
```



## Applying a Scheduler Map to an Interface

The final step in configuring the router to service its queues is to associate the scheduler map with an outbound interface. When packets arrive at the output interface, the router places the packets in the appropriate queue for the forwarding class of the traffic. The router determines the order in which to transmit packets from the queues by referring to the parameters configured in the scheduler associated with that queue. The scheduler map tells the router which scheduler should be associated with each queue on a particular interface. You can use different scheduler maps on different interfaces to specify different scheduling parameters for queues on a per-interface basis.

The scheduler applies to all traffic across a physical interface unless `per-unit-scheduler` is configured for the interface under the `[edit interfaces]` hierarchy. In that case, the router schedules packets on a per-unit basis, and you can also define different scheduler maps for each unit.



### CoS Processing Overview

We first introduced the diagram shown on this slide on page 9-9. Now that we have covered the concepts, we will review the diagram to ensure that you understand how these concepts fit into the big picture.

We discussed most of these concepts in this chapter. We discussed policers in Chapter 2. We will discuss virtual channels in Chapter 10.

## Agenda: Class of Service

---

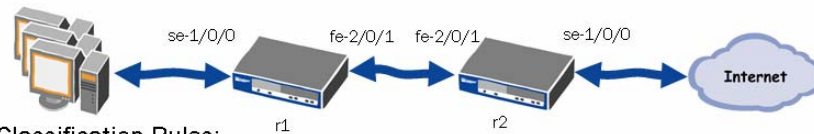
- CoS Overview
- Traffic Classification
- Traffic Queueing
- Traffic Scheduling
- ➔ Example
- Troubleshooting



### Example

The slide highlights the topic we discuss next.

## Example: Topology and Goals



### Classification Rules:

- Traffic to/from professors (192.168.25.64/26) should be placed in the professor forwarding class.
- Traffic to/from students (192.168.25.128/25) should be placed in the student forwarding class, provided the traffic from the students doesn't exceed 1Mb/s.
- Traffic from students in excess of 1Mb/s should be placed in the best-effort forwarding class.

### Queues:

- 0: best-effort
- 1: students
- 2: professors
- 3: network-control

### Scheduling Rules:

- network-control traffic should be given the highest priority and 5% of the available bandwidth.
- Traffic from professors should be given the next-highest priority (medium-high) and be allocated 50% of the bandwidth. They should be allowed to use leftover bandwidth.
- Traffic from students should be given the next-highest priority and allocated 40% of the bandwidth. They should be allowed to use leftover bandwidth.
- best-effort traffic should be given the lowest priority and allocated 5% of the bandwidth. They should not be allowed to use extra bandwidth.

## Topology and Goals

The example on the slide shows a fairly simple topology with an across-the-network CoS application. We show the forwarding classes and queue mappings, as well as the traffic classification and scheduling rules.

For simplicity, we use the same scheduling rules for all traffic on all interfaces. Note that once traffic is associated with queues other than Queue 0 and Queue 3, you must configure a scheduler for those queues on *all* interfaces. Failure to do so might lead to serious performance problems for traffic that exits an interface through a queue for which no scheduler is defined.

## Example: R1 Ingress Multifield Classifier

```

interfaces {
  se-1/0/0 {
    unit 0 {
      family inet {
        filter {
          input apply-cos-markings;
        }
      }
    }
  }
}
firewall {
  policer student-policer {
    if-exceeding {
      bandwidth-limit 1m;
      burst-size-limit 28125;
    }
    then forwarding-class best-effort;
  }
  family inet {
    filter apply-cos-markings {
      term from-professors {
        from {
          source-address {
            192.168.25.64/26;
          }
        }
        then {
          forwarding-class professors;
          accept;
        }
      }
      term from-students {
        from {
          source-address {
            192.168.25.128/26;
          }
        }
        then {
          policer student-policer;
          forwarding-class students;
          accept;
        }
      }
      term default {
        then accept;
      }
    }
  }
}

```

### R1 Ingress Multifield Classifier

This slide shows the configuration necessary to implement the defined classification rules on R1.

## Example: R2 Ingress Multifield Classifier

```

interfaces {
  se-1/0/0 {
    unit 0 {
      family inet {
        filter {
          input apply-cos-markings;
        }
      }
    }
  }
}
firewall {
  family inet {
    filter apply-cos-markings {
      term to-professors {
        from {
          destination-address {
            192.168.25.64/26;
          }
        }
        then {
          forwarding-class professors;
          accept;
        }
      }
      term to-students {
        from {
          destination-address {
            192.168.25.128/26;
          }
        }
        then {
          forwarding-class students;
          accept;
        }
      }
      term default {
        then accept;
      }
    }
  }
}

```



### R2 Ingress Multifield Classifier

This slide shows the configuration necessary to implement the specified classification rules on R2.

## Example: Forwarding Class, BA, and Scheduler

```

class-of-service {
  forwarding-classes {
    queue 1 students;
    queue 2 professors;
  }
  interfaces {
    fe-2/0/1 {
      scheduler-map professor-student-scheduler;
      unit 0 {
        classifiers {
          inet-precedence default;
        }
        rewrite-rules {
          inet-precedence default;
        }
      }
    }
    se-1/0/0 {
      scheduler-map professor-student-scheduler;
    }
  }
  scheduler-maps {
    professor-student-scheduler {
      forwarding-class network-control scheduler sched-network-control;
      forwarding-class professors scheduler sched-professors;
      forwarding-class students scheduler sched-students;
      forwarding-class best-effort scheduler sched-best-effort;
    }
  }
}

schedulers {
  sched-network-control {
    transmit-rate percent 5;
    buffer-size percent 5;
    priority high;
  }
  sched-professors {
    transmit-rate percent 50;
    buffer-size percent 50;
    priority medium-high;
  }
  sched-students {
    transmit-rate percent 40;
    buffer-size percent 40;
    priority medium-low;
  }
  sched-best-effort {
    transmit-rate percent 5;
    buffer-size percent 5;
    priority low;
  }
}

```

### Remaining CoS Configuration

This slide shows the remaining CoS configuration (for both routers).

## Agenda: Class of Service

---

- CoS Overview
- Traffic Classification
- Traffic Queueing
- Traffic Scheduling
- Example
- Troubleshooting



### Troubleshooting

The slide highlights the topic we discuss next.



## Useful Commands (1 of 2)

- **show class-of-service interface interface**

```
lab@London> show class-of-service interface fe-2/0/1
Physical interface: fe-2/0/1, Index: 142
Queues supported: 8, Queues in use: 4
Scheduler map: professor-student-scheduler, Index: 15041
Input scheduler map: <default>, Index: 3

Logical interface: fe-2/0/1.0, Index: 68
Object      Name          Type      Index
Rewrite     ipprec-default ip         29
Classifier   ipprec-default ip         11
```

- **show interfaces interface detail**

```
lab@London> show interfaces fe-2/0/1 detail
[...]
Egress queues: 8 supported, 8 in use
Queue counters:      Queued packets  Transmitted packets  Dropped packets
0 best-effort        1                1                    0
1 students           0                0                    0
2 professors         0                0                    0
3 network-cont       0                0                    0
```



## Useful Troubleshooting Commands

You can quickly see most of the CoS configuration for an interface using the **show class-of-service interface interface** command. This command displays the active configuration, including default values. You can use other **show class-of-service** commands to display details about various components of the CoS configuration. For example, to see how the `ipprec-default` classifier maps the type-of-service field in the IP header to a forwarding classes, execute the **show class-of-service classifier name `ipprec-default`** command.

One of the best ways to ensure that the CoS configuration is performing as expected is to make sure traffic is being placed into the expected queues on output. On most interfaces, you can see summary statistics for each queue using the **show interfaces detail** and **show interfaces extensive** commands. For some encapsulation types, these statistics are not available with this command.

## Useful Commands (2 of 2)

### ■ `show interfaces queue interface`

```
lab@London> show interfaces queue fe-2/0/1
Physical interface: fe-2/0/1, Enabled, Physical link is Up
  Interface index: 142, SNMP ifIndex: 62
Forwarding classes: 8 supported, 8 in use
Egress queues: 8 supported, 8 in use
Queue: 0, Forwarding classes: best-effort
  Queued:
    Packets      :           1           0 pps
    Bytes        :          42           0 bps
  Transmitted:
    Packets      :           1           0 pps
    Bytes        :          42           0 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
Queue: 1, Forwarding classes: students
  Queued:
    Packets      :           0           0 pps
    Bytes        :           0           0 bps
[...]
```



### Another Useful Command

You can also see how traffic is being queued using the `show interfaces queue` command. This command shows much more detailed statistics for each queue and is available for all interfaces.

## Troubleshooting Hints

### ■ Hints:

- Add a counter to an existing multifield classifier
- Use **monitor traffic** to look for behavior aggregate markings in the headers of traffic destined to the router
- Add an input firewall filter that counts traffic assigned to each forwarding class, matches on BA fields, or both
- Look for drops in **show interfaces queue** output
- Use **show interfaces queue** to see if a queue is exceeding its transmission rate



### Troubleshooting Hints

This slide provides some hints for helping to track down problems with a CoS deployment. If a class of traffic is not performing as expected, you should ensure that the traffic is being properly (and consistently) classified and that the traffic is not exceeding the limits of its queue.

You can ensure that the traffic is being assigned the correct forwarding class by adding a counter or a log statement to the multifield classifier. This option allows you to determine if the traffic is matching the expected firewall filter term. You can use the **monitor traffic** command to determine if the correct BA field is being set on packets destined to the router. You can also set up an input firewall filter that matches on a packet's assigned forwarding class and counts or logs packets. Because firewall filters are applied after BA classifiers, the firewall filter lets you match on the forwarding class set by the BA classifier. You can also use an input firewall filter to match on some BA fields.

Another cause of CoS problems is correctly classified traffic exceeding the limits of its queue. This scenario might be the cause when the performance problems center around a single output interface. You can troubleshoot this problem by examining the output of the **show interfaces queue** command for drops and to ensure that the traffic rate of a queue is not exceeding its configured transmission rate. If a traffic class experiences poor performance because it is exceeding its transmission rate, you might need to adjust the transmission rates to meet performance expectations, or it might be that the performance is simply the best that can be expected given the other CoS constraints.

## Review Questions

---

1. What is the difference between a multifold classifier and a behavior aggregate classifier? When should you use each?
2. How are forwarding classes related to queues?
3. What is the difference between a scheduler and a scheduler map?
4. How do transmission rates effect prioritization?



### This Chapter Discussed:

- The way CoS is processed on the J-series and M-series routers; and
- Implementing appropriate CoS policies given a traffic prioritization design.

## Lab 7: Class-of-Service Configuration

- Assign forwarding classes to queues.
- Use a multifield classifier to assign traffic to forwarding classes.
- Create schedulers and scheduler maps to implement CoS constraints.
- Monitor CoS performance.



### Lab 7: Class-of-Service Configuration

The slide lists the objectives for this lab.





# **Advanced Juniper Networks Routing in the Enterprise**

## **Chapter 10: Branch Office Connectivity**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - List three branch office connectivity solutions and the traffic considerations that apply to each solution
  - Implement these branch office connectivity solutions given proper constraints and parameters



### This Chapter Discusses:

- Several branch office connectivity solutions and the traffic considerations that apply to each solution; and
- Implementing these branch office connectivity solutions, given proper constraints and parameters.



## **Agenda: Branch Office Connectivity**

---

- Overview of Connectivity Options
- Routing and Security Implications
- CoS Considerations



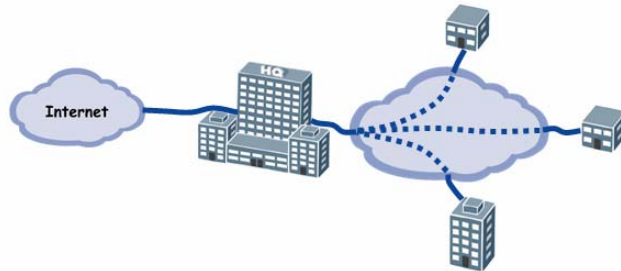
### **Overview of Connectivity Options**

The slide lists the topics we discuss in this chapter. We discuss the highlighted topic first.

## Branch-Office Connectivity

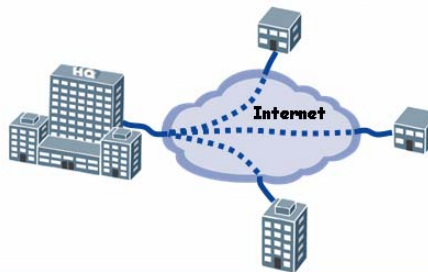
### ■ Traditional

- Frame Relay
- ATM
- Leased lines



### ■ Recent

- MPLS-based VPNs
- IPsec VPNs



### Traditional Branch Office Connectivity

Traditionally, branch offices have been connected to corporate networks through Frame Relay, ATM, or leased line connections to a regional headquarters or the corporate headquarters. These connections composed a private network over which branch offices could reach corporate resources securely and efficiently. Often, Internet access—if it was provided at all—was provided by this same private network, allowing network administrators to centrally monitor and control all Internet traffic.

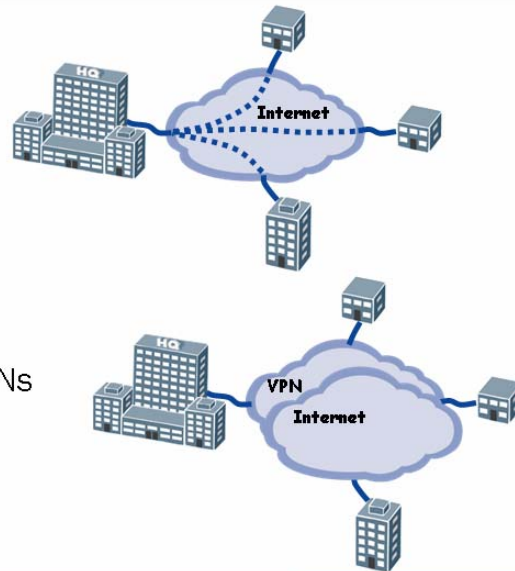
### Recent Branch Office Connectivity

In recent years, VPNs have become much more common and cost effective, especially when compared to leased lines and other branch office network connectivity solutions. This solution has led many companies to begin decommissioning their private Frame Relay, ATM, or leased line networks and move their corporate network connectivity to VPNs.

Network designers should consider several questions when making this transition. First, they should understand the various options available for implementing VPNs. Second, they should understand the routing and security implications of their implementation decisions. Finally, there are some CoS considerations they should understand.

## VPN Options

- Do-it-yourself
  - IPSec VPN
  
- Provider-provisioned
  - Usually MPLS-based VPNs
  - Layer 2 or Layer 3



### Customer-Provisioned VPNs

One of the ways to provide corporate network connectivity over VPNs is with customer-provisioned VPNs. Often, these VPNs take the form of IPSec tunnels that connect branch offices to regional or corporate headquarters, following a similar topology to the Frame Relay or leased line networks that they replace. To implement these VPNs, each branch office must have an Internet connection over which the VPN traffic can flow.

### Provider-Provisioned VPNs

The other common VPN option is provider-provisioned VPNs. Often, these VPNs are provisioned using MPLS within the provider's network to properly mark and separate traffic. These MPLS-based VPNs can be either Layer 2 or Layer 3 VPNs. With a Layer 2 VPN, a provider transparently connects two remote customer sites as if they were directly connected to each other. Although the provider transports the data frames through its network, this is transparent to the customer, and the two ends appear to be directly connected. Another kind of VPN called *virtual private LAN service* (VPLS) emulates a Layer 2 LAN, allowing multiple sites to appear to be directly connected to the same LAN! With a Layer 3 VPN, a provider connects multiple customer sites and provides Layer 3 routing services between them. The provider must have proper Layer 3 routing information for each site, whether through static routes or a dynamic routing protocol.

*Continued on next page.*

### **Provider-Provisioned VPNs (contd.)**

In any of these kinds of provider-provisioned VPNs, it is usually possible for each site to have both Internet connectivity and VPN connectivity provided on the same physical circuit, but it is not necessary that they have both.

## **Agenda: Branch Office Connectivity**

---

- Overview of Connectivity Options
- Routing and Security Implications
- CoS Considerations



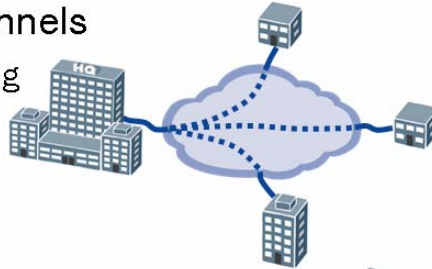
### **Routing and Security Implications**

The slide highlights the topic we discuss next.

## Internal Routing

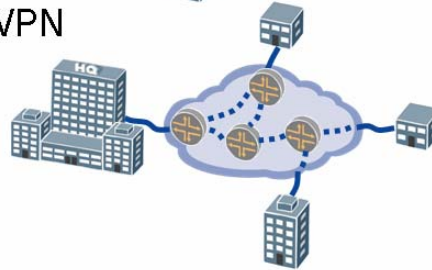
### ■ Layer 2 VPN or IPSec tunnels

- Enterprise controls routing
- Intelligent metrics
- Nonchatty protocol



### ■ Provider-provisioned Layer 3 VPN

- Provider controls routing
- Static routes, BGP, or OSPF



## Layer 2 VPNs and IPSec Tunnels

With both Layer 2 VPNs and customer-provisioned IPSec tunnels, the enterprise controls the Layer 3 routing. If you will ever have more than one path available for traffic to follow from your site to the corporate network, running an IGP will help choose the best path and will provide automatic failover between paths. If your network uses an IGP over VPNs, it is important to choose intelligent metrics. A good metric choice is related to the actual delay encountered over a link. One cause of latency is the limits of how fast a signal can travel a particular distance (in fiber-optic networks, for example, this speed is bound by the speed of light).

Default IGP metrics do not account for latency. However, because one virtual Layer 2 connection might connect two locations relatively close to each other while another might span an entire continent, it is important to choose IGP metrics that do account for the latency induced by distance. A good (and easy) method for initially setting metrics for VPNs that cross connections of equivalent speed is to make the metric equal to the distance (in miles or kilometers) between two locations. You can easily get this figure by using one of many Web sites to get driving directions between the two locations in question.

Because all the IGP traffic must cross the VPN and the charge for the VPN might be based on the traffic exchanged, it is important to choose a protocol that will be less chatty. In general, OSPF or IS-IS is a good choice.

*Continued on next page.*

## Provider-Provisioned Layer 3 VPNs

When you use a provider-provisioned Layer 3 VPN, the provider controls the Layer 3 routing. Therefore, it must have accurate routing information for your corporate network. The easiest method of controlling this information is through static routes. If your sites do not have redundant connections to the VPN and your routing information changes very infrequently, static routes are an acceptable choice that will most likely prove to be the least work for both you and the provider. However, if you have redundant connections or your routing information changes, you will need to use a dynamic routing protocol.

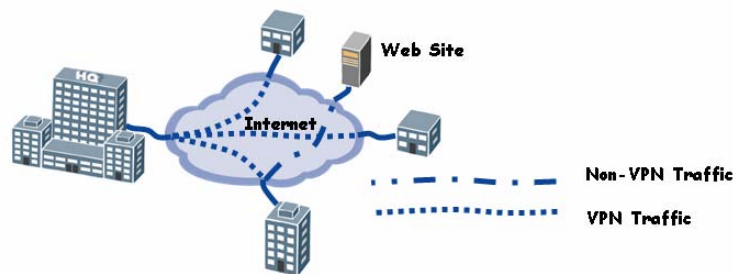
Your provider will likely specify for you the dynamic routing protocol that it wants to use for the Layer 3 VPN service. The most common is BGP. Another possibility is OSPF. In general, BGP is preferred over OSPF.

In the BGP deployment, extra configuration might be needed if the same AS number is used by multiple offices attached to the VPN. By default, routes announced by each location would be rejected by the other locations because they would fail the AS-path checks that prevent routing information loops. The preferred solution to this problem is to use different AS numbers for every site. If this is not possible, the provider can configure features to remove the AS number from the AS path prior to sending updates to the VPN sites. If neither of these solutions is possible, you can use the **set routing-options autonomous-system *asn* loops *number*** command, where *asn* is the local AS number and *number* is the number of times the local AS number should be allowed to appear in the path of BGP-received routes.

## Split Tunneling

### ■ Split tunneling:

- Allows direct Internet access from remote sites
- Cheaper, more efficient traffic flows
- Distributed traffic control and security



## Split Tunneling

When you configure VPNs, there are two primary ways to configure routing for traffic destined to locations outside the corporate network. In one configuration, all traffic is transmitted to a central location (or one of several central locations) where you can centrally control Internet access and you can apply a standard corporate security policy. In the other configuration (commonly called *split tunneling*), only traffic destined for locations on the corporate network flows over the VPN, while traffic destined for locations on the Internet is handled locally.

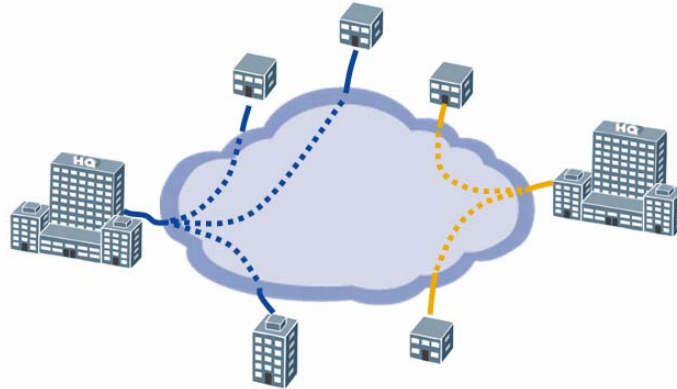
Split tunneling usually results in cheaper and more efficient handling of traffic. When the cost of Internet access is based on usage, having Internet traffic backhauled to a central location results in being charged twice for Internet traffic (once when it is sent between the branch office and the central location, and again when it is sent between the central location and the destination on the Internet). Additionally, having traffic sent through a central location some distance away can add latency, reducing performance.

However, if Internet access is provided at each site through split tunnelling, security functions must be performed at each site separately. Each location must implement some combination of stateful firewalls, intrusion detection and prevention, traffic logging, and any other functions required by the company's security policy. Depending on how often changes must be made and the complexity of the security policy, managing these functions effectively at many different locations might also prove to be quite complex and costly.



## Encryption

- What keeps the VPN *private*?



### Encryption

The final security implication is one of data security. Most provider-provisioned Layer 3 VPNs provide no encryption to protect information against third-party interception. The only thing preventing data from falling into the wrong hands is the configuration on the provider's routers that associates various ports on their routers with various VPNs. If your data is sensitive enough that it needs more security than that, you should consider deploying IPSec encryption for the data.

The slide shows an example of a typical provider network where multiple customers' VPNs are provided over the same network. The only thing keeping the two VPNs separate is the configuration that defines which ports are part of each VPN.

## **Agenda: Branch Office Connectivity**

---

- Overview of Connectivity Options
- Routing and Security Implications
- CoS Considerations

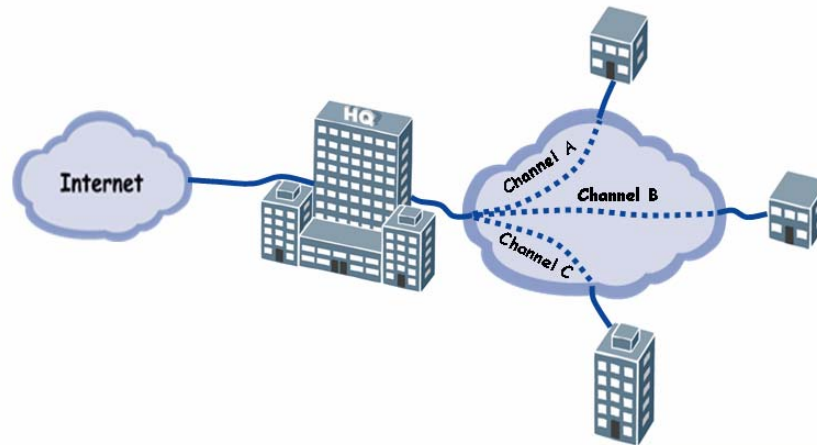


### **CoS Considerations**

The slide highlights the topic we discuss next.

## Virtual Channels

- Separate traffic into channels
  - Allow per-channel shaping and CoS



### Virtual Channels

One of the problems that you can encounter with VPNs is that the central site frequently has a higher-capacity connection to the network than the remote sites. In these cases, it is possible for the central site to overload a remote site's connection. The worst-case scenario is that the central site tries to send a very high amount of traffic to a single site. Because this traffic is competing to be transmitted from the central site, traffic to other sites is delayed. At the same time, the central site is sending so much traffic to the single site that it exceeds the capacity of that site's connection to the point that the provider drops or delays high-priority traffic (such as VoIP). You can avoid this scenario by configuring virtual channels.

Virtual channels are only available on J-series routers. Using an outbound filter, you specify which virtual channel traffic should use. You then can configure traffic shaping and other CoS parameters on a per-channel basis for that outbound interface.

## Virtual Channel Configuration (1 of 2)

```

class-of-service {
    virtual-channels {
        ChannelA;
        ChannelB;
        ChannelC;
        default-vc;
    }
    virtual-channel-groups {
        vpn-vc-group {
            ChannelA {
                scheduler-map vpn-cos;
                shaping-rate 1m;
            }
            ChannelB {
                scheduler-map vpn-cos;
                shaping-rate 1m;
            }
            ChannelC {
                scheduler-map vpn-cos;
                shaping-rate 1m;
            }
            default-vc {
                scheduler-map vpn-cos;
                default;
            }
        }
    }
}

firewall {
    filter choose-vc {
        term SiteA {
            from {
                destination-address {
                    10.12.1.0/24;
                }
            }
            then {
                virtual-channel ChannelA;
                accept;
            }
        }
        term SiteB {
            from {
                destination-address {
                    10.12.2.0/24;
                }
            }
            then {
                virtual-channel ChannelB;
                accept;
            }
        }
        [...]
        term default {
            then accept;
        }
    }
}

```



### Virtual Channel Configuration

This slide shows the configuration of virtual channels. On the left, you see the virtual channels first defined and then put in a group named *vpn-vc-group* with specific CoS parameters applied to each parameter. All traffic that is not placed into one of the user-defined virtual channels is placed in the *default-vc* virtual channel.

On the right side of the slide, you see the firewall filter that assigns traffic to the virtual channels. It is applied outbound on the output interface.

## Virtual Channel Configuration (2 of 2)

```

interfaces {
    fe-2/0/0 {
        per-unit-scheduler;
        vlan-tagging;
        unit 699 {
            description "Layer 2 VPN";
            vlan-id 699;
            family inet {
                filter {
                    output choose-vc;
                }
                address 172.16.1.1/24;
            }
        }
    }
}

class-of-service {
    interfaces {
        fe-2/0/0 {
            unit 699 {
                virtual-channel-group vpn-vc-group;
            }
        }
    }
}

```



Juniper your Net

10-15

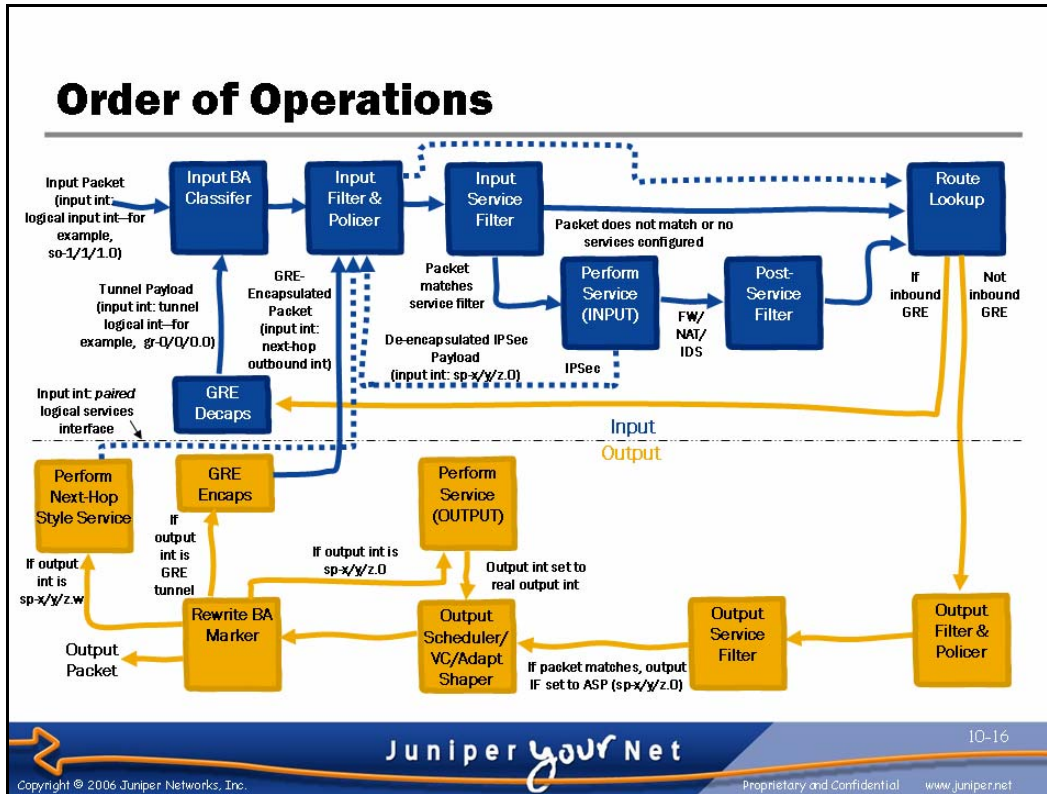
Copyright © 2006 Juniper Networks, Inc.

Proprietary and Confidential www.juniper.net

### More Virtual Channel Configuration

On this slide, you see that the Layer 2 VPN is provisioned as a VLAN on a Fast Ethernet interface. To accommodate different CoS configurations for each VLAN, we configured the interface with `per-unit-scheduler`. You must use per-unit scheduling when you use virtual channel groups for scheduling on an interface. Also, we applied the `choose-vc` filter outbound on the unit used for the VPN.

Under the `[edit class-of-service interfaces]` hierarchy, we configured the router to use the virtual channel group `vpn-vc-group` for scheduling and shaping.



### Order of Operations

It is important that you know the order in which the router performs certain functions. The order in which the router performs packet processing affects the information that is available to the router at any point in the processing. As you configure a router, you should consider how the various elements of the configuration interact with each other. It is especially important that you consider these things as your router's configuration grows more complex.

## Order of Operations—Some Implications

- Must consider the information that is available at any given point
  - Example: IPsec VPNs:
    - Incoming: Does a BA classifier run on the inner packet?
    - Outgoing: Should an output filter on the output interface match the original or encrypted packet?
  - Example: GRE tunnels:
    - Incoming: Does a BA classifier run on the inner packet?
    - Outgoing: Why are my outgoing packets blocked?



### Some Implications to the Order of Operations

This slide lists some examples of the practical implications of the order of operations. For example, consider IPsec VPNs. When the router first receives the IPsec packet, the router processes the outer packet through a BA classifier, policer, and the remainder of the normal incoming packet processing shown on the previous slide. The router maintains this forwarding class and loss priority throughout the packet's processing, even as the AS PIC de-encapsulates the packet. When the AS PIC decapsulates an IPsec-encrypted packet, it preserves the original type-of-service (ToS) bits from the IP header of the encrypted packet. However, the router does not then process the de-encapsulated packet through a BA classifier. Therefore, if you want to change the forwarding class and loss priority based on the ToS bits of the de-encapsulated (inner) packet, you must write a multifield classifier that matches DSCP or IP precedence values and assigns the forwarding class and loss priority appropriately, and apply that classifier to the correct interface (see the previous slide).

Continuing to use IPsec VPNs as an example, assume that you want both to assign traffic to virtual channels and also to secure the traffic within an IPsec VPN. What match criteria should you use in the output filter on the interface that connects to your ISP? If you use an interface-style service set, the output firewall filter is evaluated before the packet is encrypted; therefore, you should match the unencrypted packets. If you use a next-hop-style service set, the output firewall filter is evaluated after the packet is encrypted; therefore, you should match the encrypted packets.

*Continued on next page.*

### Some Implications to the Order of Operations (contd.)

For a different example, consider GRE tunnels. Like IPsec VPNs, when an AS or Tunnel PIC de-encapsulates a GRE packet, it preserves the ToS bits from the de-encapsulated (inner) packet. Unlike IPsec VPNs, when an AS or Tunnel PIC de-encapsulates a GRE packet and sends the de-encapsulated packet to the FPC, the router runs a BA classifier against the packet and assigns the forwarding class and loss priority appropriately. If you do not configure a BA classifier on the `gr-` interface, the router uses the default BA classifier and assigns traffic to the best-effort or network-control forwarding classes. If you do configure a BA classifier on the `gr-` interface, the router uses it.

When an AS or Tunnel PIC encapsulates GRE traffic, it sends the encapsulated packet to the FPC and tells the PFE that the input interface is the next-hop output interface. The PFE then performs normal input processing on the packet, beginning with the input policer. Therefore, you must configure the *input* processing on the *output* interface to account for these packets. Specifically, if you configure an interface policer, it counts the *outbound* GRE packets as input traffic. Additionally, you must allow the *outbound* packets through the *input* filters on the *outbound* interface, and you must ensure that the router excludes the *outbound* packets from being processed through Layer 3 *input* services on the *outbound* interface.

Appendix A provides some detailed examples using the order of operations to understand the way configuration elements interact.



## Review Questions

1. What are the differences between Layer 2 and Layer 3 VPNs?
2. What are virtual channels?
3. If using a next-hop-style service set to encrypt traffic within an IPSec tunnel, what match conditions do you use in an output filter on the connection to your ISP to select this traffic?
4. If you use an interface-style service set in scenario listed in Question #3, what match conditions do you use?



### This Chapter Discussed:

- Several branch office connectivity solutions and the traffic considerations that apply to each solution; and
- Implementing these branch office connectivity solutions, given proper constraints and parameters.

## **Lab 8: Branch Office Connectivity (Optional)**

- Establish Frame Relay connectivity.
- Migrate to IPsec VPNs.
- Migrate to provider-provisioned Layer 2 VPNs.



### **Lab 8: Branch Office Connectivity (Optional)**

The slide lists the objectives for this lab.



# **Advanced Juniper Networks Routing in the Enterprise**

## **Appendix A: Life of a Packet**

## Chapter Objectives

- After successfully completing this chapter, you will be able to:
  - Explain how a JUNOS router processes packets



### This Chapter Discusses:

- How a JUNOS router processes packets and the way various configuration options will impact packet processing.

This is not intended to be an exhaustive guide, but is intended primarily to help explain some advanced—and difficult—concepts to JUNOS users, including the following:

- GRE tunnels (especially the *recirculation* of encapsulated packets);
- Interface-style service sets; and
- Next-hop-style service sets.

## Agenda: Life of a Packet

---

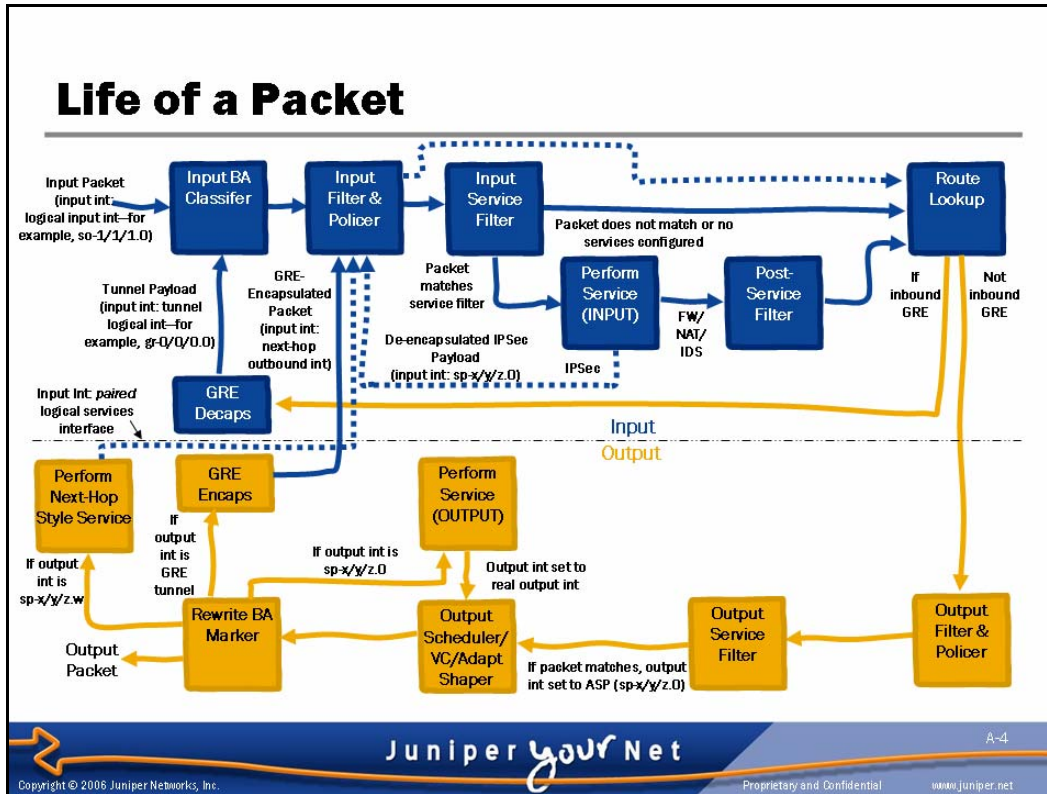
### → Overview

- Example 1: Interface-Style Service Sets
- Example 2: Next-Hop-Style Service Sets



### Overview

The slide lists the topics we cover in this chapter. We discuss the highlighted topic first.



### Life of a Packet

The diagram on the slide provides an overview of the packet flow through Juniper T-series, M-series, and J-series routers. On M-series and T-series routers, various hardware components perform these functions. On J-series routers, the `fwdd` software process performs most of these functions. For simplicity, we use the term `ASP` to refer to all Adaptive Services PICs, Adaptive Services Modules, and the equivalent software processes on the J-series routers. Also, we use the terms `PIC`, `FPC`, and `PFE` to refer both to those parts of the T-series and M-series routers as well as the equivalent software processes on the J-series routers. If it is necessary to make a distinction, we make that distinction clear.

Some differences exist between platforms. For example, on the J4350 and J6350 routers, the encryption functionality can be performed in a hardware acceleration card. Also, different hardware platforms support different class-of-service (CoS) features. Despite these differences, you apply configuration at the same points on all JUNOS routers to implement the functionality that the hardware supports.

The purpose of this diagram is to show when the router applies portions of your configuration, rather than to show actual packet flow through router components. Each box does not necessarily represent a different piece of hardware or a different software process. Rather, the boxes represent points where the router applies pieces of the configuration to packets it is processing.

The blue (dark) lines and boxes are intended to represent places where *input* processing is occurring, while the gold (light) boxes indicate *output* processing.

## Agenda: Life of a Packet

---

- Overview
- ➔ Example 1: Interface-Style Service Sets
- Example 2; Next-Hop Style Service Sets



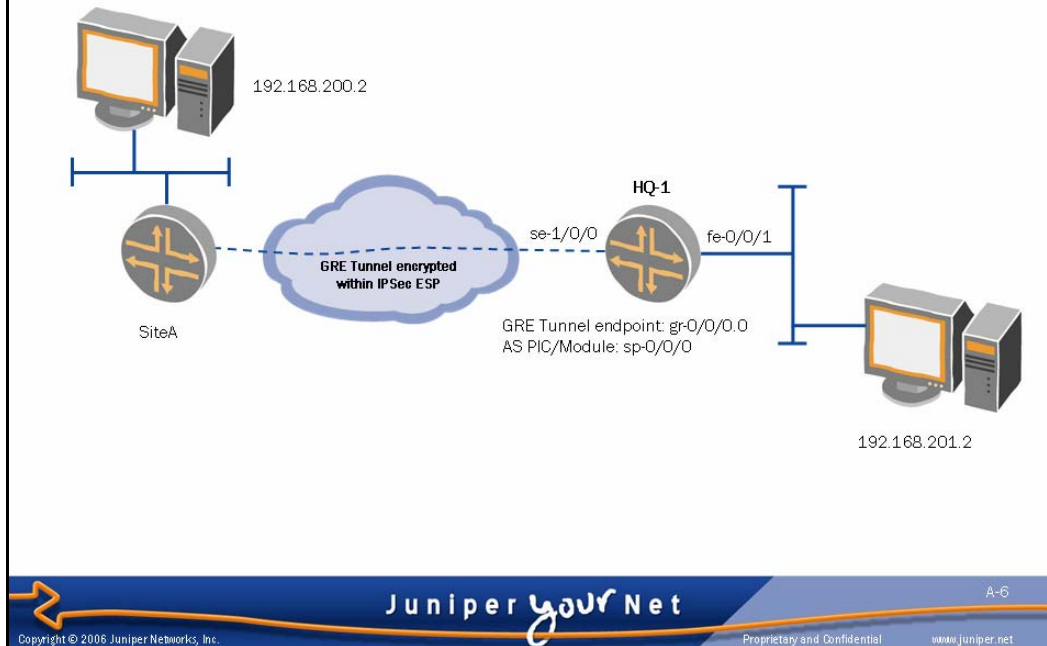
### Example 1: Interface-Style Service Sets

The slide highlights the topic we discuss next.

Hopefully, the diagram on the previous slide will become more clear as we proceed through some examples. Note that the examples are purposely complex to highlight the interaction of various features on the router.

This example will demonstrate using interface-style service sets to implement a stateful firewall, encrypt a GRE tunnel within a VPN, and perform CoS on this traffic.

## Example 1: Interface-Style Service Set



### Example 1: Overview and Configuration

In this example, we examine packet flow through the router on the right side of the slide (HQ-1). This router has a generic routing encapsulation (GRE) tunnel established between it and the branch router (on the left). The `se-1/0/0.0` interface IP address is the local endpoint of the GRE tunnel. This GRE tunnel is encrypted within an IPSec VPN. The routes uses the same endpoints for this VPN as the GRE tunnel.

Further, traffic over the GRE tunnel is secured by a stateful firewall that allows all outbound traffic from HQ-1 to the branch office, but denies all inbound connections.

To satisfy its performance requirements, this company implemented CoS throughout its network. The router uses a multifield classifier to classify traffic as it enters the `fe-0/0/1.0` interface. Routers throughout the network maintain that classification by rewriting and processing the IP precedence behavior aggregate (BA). To not overrun the slower branch office connection, a virtual-channel configuration limits traffic heading to the branch office (SiteA) to 768 kbps.

The following shows the configuration of HQ-1:

```
version 8.0R1.9;
system {
  host-name HQ-1;
  root-authentication {
    encrypted-password "$1$KI99zGk6$MbYFuBbpLffu9tn2.sI7l1"; ## SECRET-DATA
  }
  login {
```

*Continued on next page.*



**Example 1: Overview and Configuration (contd.)**

```

user lab {
    uid 2000;
    class super-user;
    authentication {
        encrypted-password "$1$84J5Maes$cni5Hrazbd/IEHr/50oY30"; ##
SECRET-DATA
    }
}
}
services {
    ftp;
    ssh;
    telnet;
}
syslog {
    user * {
        any emergency;
    }
    file messages {
        any notice;
        authorization info;
    }
    file interactive-commands {
        interactive-commands any;
    }
}
}
interfaces {
    gr-0/0/0 {
        unit 0 {
            tunnel {
                source 172.17.38.4;
                destination 172.17.37.4;
            }
            family inet {
                service {
                    input {
                        service-set sffw-outbound-only;
                    }
                    output {
                        service-set sffw-outbound-only;
                    }
                }
                address 192.168.25.130/30;
            }
        }
    }
}

```

*Continued on next page.*

**Example 1: Overview and Configuration (contd.)**

```

sp-0/0/0 {
  unit 0 {
    family inet;
  }
}
fe-0/0/1 {
  unit 0 {
    family inet {
      filter {
        input classify-packets;
      }
      address 192.168.201.1/24;
    }
  }
}
se-1/0/0 {
  per-unit-scheduler;
  unit 0 {
    family inet {
      filter {
        output choose-vc;
      }
      service {
        input {
          service-set vpn-to-SiteA service-filter
match-vpn-to-SiteA-no-gre;
          service-set sffw-outbound service-filter no-gre;
        }
        output {
          service-set vpn-to-SiteA service-filter
match-vpn-to-SiteA;
          service-set sffw-outbound;
        }
      }
      address 172.17.38.4/29;
    }
  }
}
}
routing-options {
  static {
    route 0.0.0.0/0 next-hop 172.17.38.1;
    route 192.168.200.0/24 next-hop gr-0/0/0.0;
  }
}
class-of-service {
  virtual-channels {
    SiteA-vc;
    default-vc;
  }
}

```

*Continued on next page.*

**Example 1: Overview and Configuration (contd.)**

```

virtual-channel-groups {
  wan-vc-group {
    SiteA-vc {
      scheduler-map standard-scheduler-map;
      shaping-rate 768k;
    }
    default-vc {
      scheduler-map standard-scheduler-map;
      default;
    }
  }
}
interfaces {
  gr-* {
    scheduler-map standard-scheduler-map;
    unit * {
      classifiers {
        inet-precedence default;
      }
      rewrite-rules {
        inet-precedence default;
      }
    }
  }
  sp-0/0/0 {
    scheduler-map standard-scheduler-map;
  }
  se-1/0/0 {
    unit 0 {
      virtual-channel-group wan-vc-group;
      classifiers {
        inet-precedence default;
      }
      rewrite-rules {
        inet-precedence default;
      }
    }
  }
  fe-0/0/1 {
    scheduler-map standard-scheduler-map;
  }
}
scheduler-maps {
  standard-scheduler-map {
    forwarding-class network-control scheduler my-nc-sched;
    forwarding-class expedited-forwarding scheduler my-ef-sched;
    forwarding-class assured-forwarding scheduler my-af-sched;
    forwarding-class best-effort scheduler my-be-sched;
  }
}

```

*Continued on next page.*

**Example 1: Overview and Configuration (contd.)**

```

schedulers {
  my-be-sched {
    transmit-rate percent 10;
    buffer-size percent 10;
    priority low;
  }
  my-af-sched {
    transmit-rate percent 25;
    buffer-size percent 25;
    priority medium-low;
  }
  my-ef-sched {
    transmit-rate percent 50;
    buffer-size percent 50;
    priority medium-high;
  }
  my-nc-sched {
    transmit-rate percent 15;
    buffer-size percent 15;
    priority high;
  }
}
}
firewall {
  family inet {
    filter classify-packets {
      term interactive-applications {
        from {
          protocol tcp;
          source-port [ 22 23 ];
        }
        then {
          forwarding-class expedited-forwarding;
          accept;
        }
      }
      term sip {
        from {
          protocol [ udp tcp ];
          port 5060;
        }
        then {
          forwarding-class assured-forwarding;
          accept;
        }
      }
    }
  }
}

```

*Continued on next page.*

**Example 1: Overview and Configuration (contd.)**

```

term rtp {
  from {
    protocol udp;
    port 16384-32767;
  }
  then {
    forwarding-class assured-forwarding;
    accept;
  }
}
term default {
  then accept;
}
}
service-filter no-gre {
  term ignore-gre {
    from {
      protocol gre;
    }
    then skip;
  }
  term default {
    then service;
  }
}
service-filter match-vpn-to-SiteA {
  term vpn-to-SiteA {
    from {
      source-address {
        172.17.38.4/32;
      }
      destination-address {
        172.17.37.4/32;
      }
    }
    then service;
  }
  term default {
    then skip;
  }
}
service-filter match-vpn-to-SiteA-no-gre {
  term ignore-gre {
    from {
      protocol gre;
    }
    then skip;
  }
}

```

*Continued on next page.*

**Example 1: Overview and Configuration (contd.)**

```

term vpn-to-SiteA {
    from {
        source-address {
            172.17.37.4/32;
        }
        destination-address {
            172.17.38.4/32;
        }
    }
    then service;
}
term default {
    then skip;
}
}
}
filter choose-vc {
    term SiteA {
        from {
            destination-address {
                172.17.37.4/32;
            }
        }
        then {
            virtual-channel SiteA-vc;
            accept;
        }
    }
    term default {
        then accept;
    }
}
}
services {
    stateful-firewall {
        rule allow-all-telnet-ping {
            match-direction input;
            term allow-all-telnet {
                from {
                    source-address {
                        172.17.37.0/24;
                        172.17.38.0/24;
                    }
                }
                applications junos-telnet;
            }
            then {
                accept;
            }
        }
    }
}

```

*Continued on next page.*

**Example 1: Overview and Configuration (contd.)**

```

term allow-all-ping {
    from {
        applications junos-icmp-ping;
    }
    then {
        accept;
    }
}
}
rule allow-vpn-traffic {
    match-direction input-output;
    term allow-ike-ipsec {
        from {
            applications [ junos-ipsec-esp junos-ike ];
        }
        then {
            accept;
        }
    }
}
rule allow-all-outbound {
    match-direction output;
    term default {
        then {
            accept;
        }
    }
}
}
nat {
    pool external-IP {
        address 172.17.38.4/32;
        port automatic;
    }
    rule translate-private {
        match-direction output;
        term match-192-168-201 {
            from {
                source-address {
                    192.168.201.0/24;
                }
            }
            then {
                translated {
                    source-pool external-IP;
                    translation-type source dynamic;
                }
            }
        }
    }
}

```

*Continued on next page.*

**Example 1: Overview and Configuration (contd.)**

```

    term default {
        then {
            no-translation;
        }
    }
}
}
service-set sffw-outbound {
    stateful-firewall-rules allow-all-outbound;
    stateful-firewall-rules allow-all-telnet-ping;
    stateful-firewall-rules allow-vpn-traffic;
    nat-rules translate-private;
    interface-service {
        service-interface sp-0/0/0;
    }
}
service-set sffw-outbound-only {
    stateful-firewall-rules allow-all-outbound;
    interface-service {
        service-interface sp-0/0/0;
    }
}
service-set vpn-to-SiteA {
    interface-service {
        service-interface sp-0/0/0;
    }
    ipsec-vpn-options {
        local-gateway 172.17.38.4;
    }
    ipsec-vpn-rules HQ-to-SiteA;
}
ipsec-vpn {
    rule HQ-to-SiteA {
        term gre-tunnel {
            from {
                source-address {
                    172.17.38.4/32;
                }
                destination-address {
                    172.17.37.4/32;
                }
            }
            then {
                remote-gateway 172.17.37.4;
                dynamic {
                    ike-policy main_mode_ike_policy;
                    ipsec-policy dynamic_ipsec_policy;
                }
            }
        }
    }
}

```

*Continued on next page.*

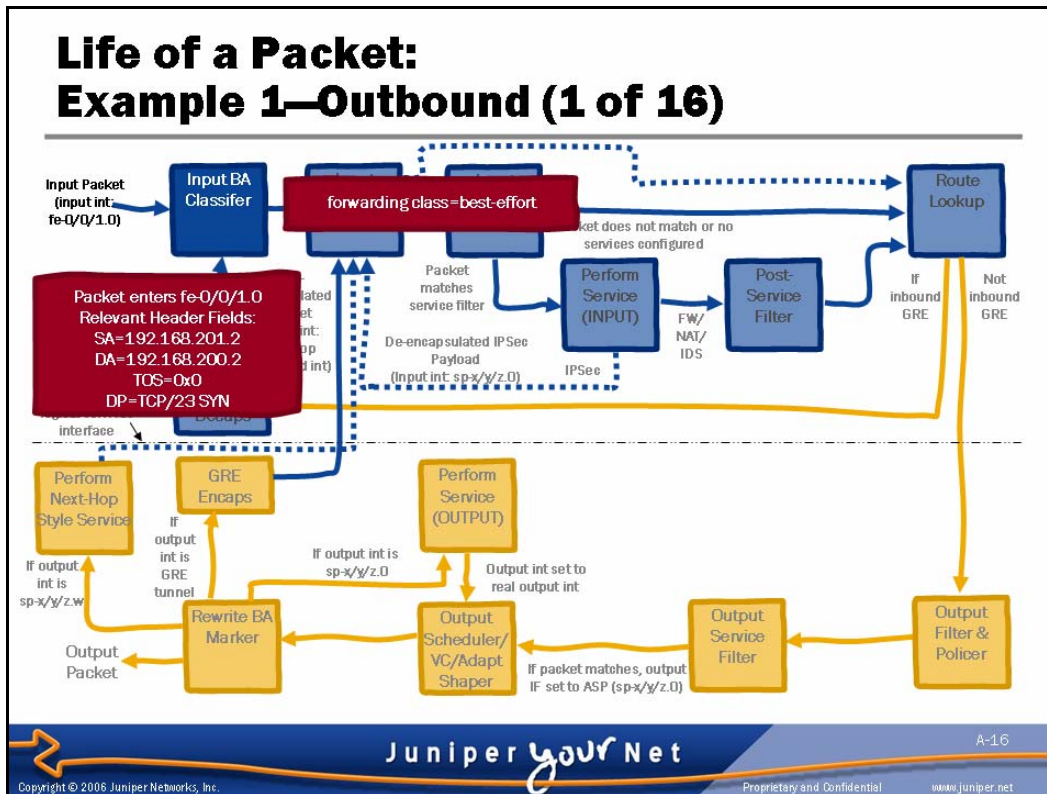


**Example 1: Overview and Configuration (contd.)**

```

    }
    match-direction output;
  }
  ipsec {
    proposal esp_shal_3des_ipsec_proposal {
      protocol esp;
      authentication-algorithm hmac-shal-96;
      encryption-algorithm 3des-cbc;
    }
    policy dynamic_ipsec_policy {
      perfect-forward-secrecy {
        keys group2;
      }
      proposals esp_shal_3des_ipsec_proposal;
    }
  }
  ike {
    proposal psk_shal_3des_ike_proposal {
      authentication-method pre-shared-keys;
      authentication-algorithm shal;
      encryption-algorithm 3des-cbc;
    }
    policy main_mode_ike_policy {
      mode main;
      proposals psk_shal_3des_ike_proposal;
      pre-shared-key ascii-text "$9$KqT3AtORcl0BlMLNY2UjH"; ##
SECRET-DATA
    }
  }
  traceoptions {
    flag ike;
  }
  establish-tunnels immediately;
}

```



### Original Packet

This example explores the way the router processes the first two packets in a Telnet session opened from 192.168.201.2 to 192.168.200.2. Because this user is attempting to begin a new TCP session, the SYN bit is set in the packet's TCP header. The packet is destined for TCP port 23 (the Telnet port). The type-of-service (ToS) bits in the IP header are set to 0x0 when the packet arrives.

Because the packet arrives on the fe-0/0/1.0 interface, the router sets the input interface to fe-0/0/1.0 and the router processes the packet through the BA classifier configured for this interface. (Be aware of the input interface because this interface changes as the router continues processing the packet.)

Although no BA classifier is configured on fe-0/0/1.0, the router will use a default IP precedence classifier, as you can see from the following output:

```
lab@HQ-1> show class-of-service interface fe-0/0/1.0
Logical interface: fe-0/0/1.0, Index: 76
Object      Name                               Type      Index
Classifier  ipprec-compatibility             ip        12
```

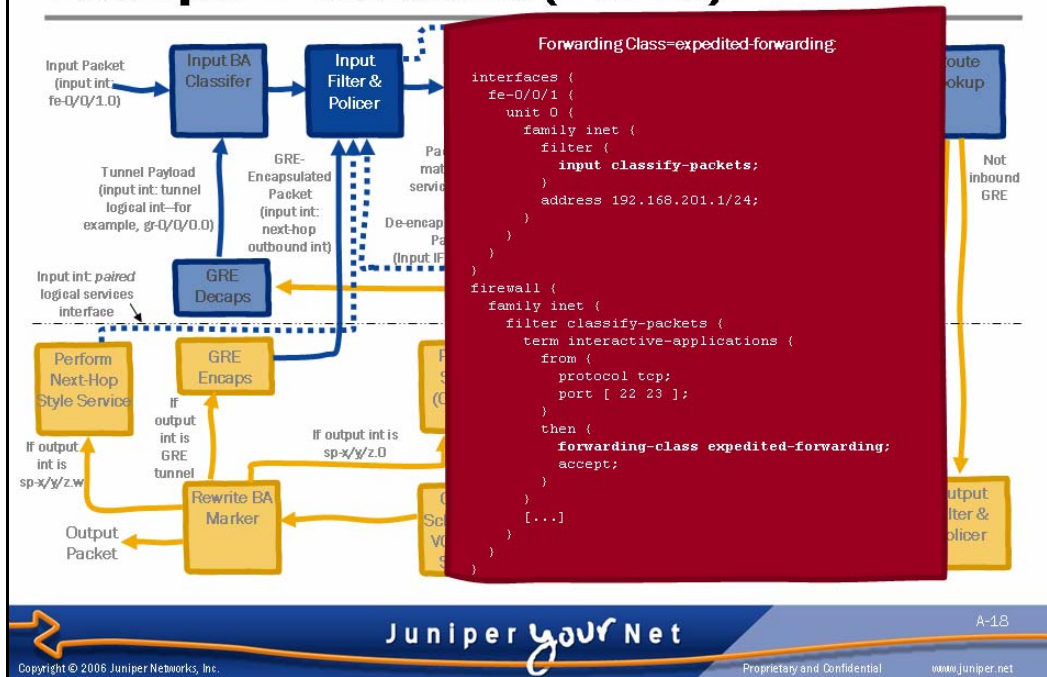
*Continued on next page.*

### Original Packet (contd.)

The router uses this classifier if no other classifier is configured. It places all network-control traffic in the network-control forwarding class and places all other traffic in the best-effort forwarding class. This classifier is designed to work with the default scheduler, which only schedules traffic from the best-effort and network-control forwarding classes. As you can see, if the IP ToS bits are set to 0x0, the `ipprec-compatibility` classifier will place the traffic in the best-effort forwarding class:

```
lab@HQ-1> show class-of-service classifier name ipprec-compatibility
Classifier: ipprec-compatibility, Code point type: inet-precedence, Index: 12
  Code point      Forwarding class      Loss priority
  000             best-effort           low
  001             best-effort           high
  010             best-effort           low
  011             best-effort           high
  100             best-effort           low
  101             best-effort           high
  110             network-control       low
  111             network-control       high
```

## Life of a Packet: Example 1—Outbound (2 of 16)



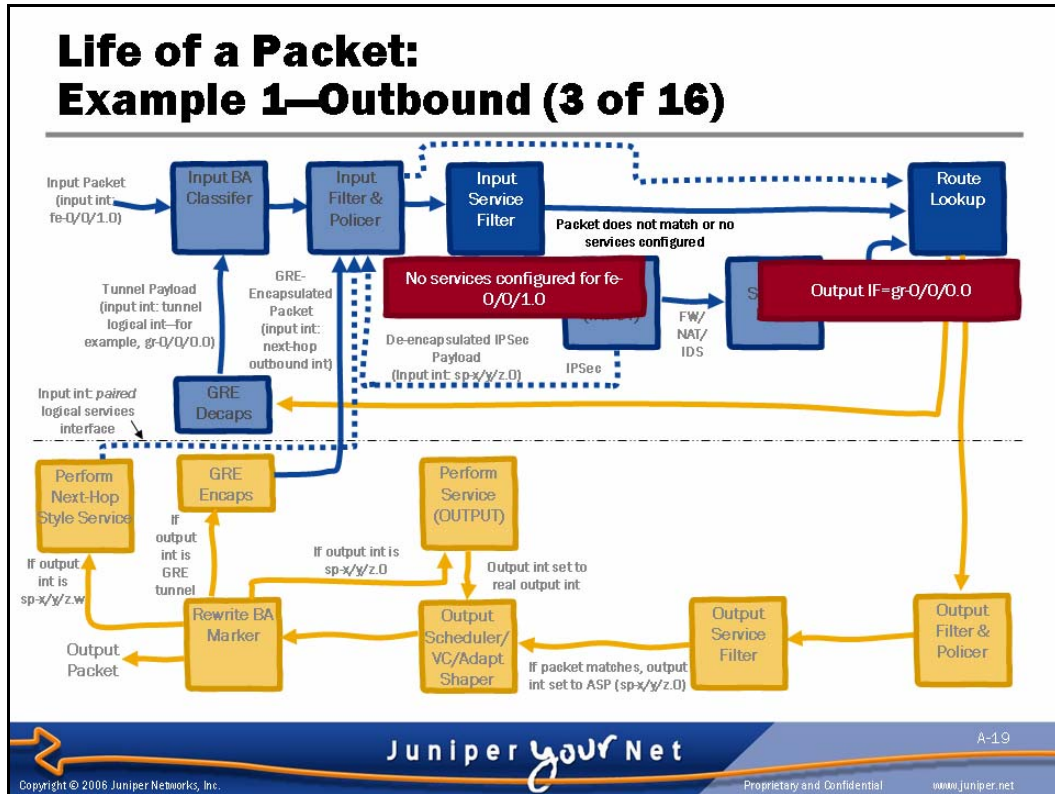
### Input Filter and Policer

The PFE next processes the input policer, if one is configured. Then, it processes the input filter. (If the filter references a policer, it is processed at the same time.)

In this case, the router is configured to filter input IP packets on the fe-0/0/1.0 interface using a filter called *classify-packets*. The packet matches the first term of *classify-packets* and is assigned to the expedited-forwarding forwarding class. The router maintains this packet's forwarding class assignment throughout the remainder of packet processing unless it is changed by later steps in the packet processing:

```

firewall {
  family inet {
    filter classify-packets {
      term interactive-applications {
        from {
          protocol tcp;
          source-port [ 22 23 ];
        }
        then {
          forwarding-class expedited-forwarding;
          accept;
        }
      }
    }
  }
}
[...]
```

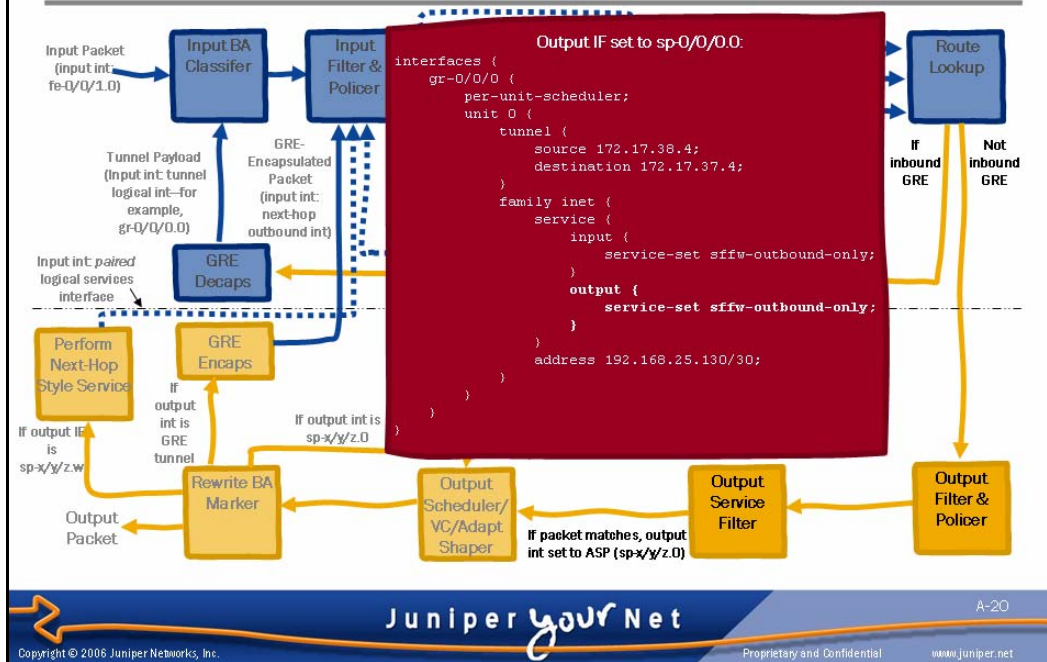


### Input Services and Route Lookup

The PFE next checks the packet against service filters, if any services are configured. In this case, no service is configured on `fe-0/0/1.0`, so the PFE next performs a lookup in the forwarding table. This lookup returns a next-hop interface of `gr-0/0/0.0` (due to the static route for `192.168.200.0/24` pointing to this interface). The PFE sets the output interface to `gr-0/0/0.0` and moves to processing output filters for that interface.

Note that had the input filter caused the packet to be processed by a different RIB (via filter-based forwarding [FBF]), the PFE would have skipped the service filters and proceeded directly to a route lookup in the appropriate forwarding table. To use interface-style services and FBF together, you should implement the FBF using a post-service filter.

## Life of a Packet: Example 1—Outbound (4 of 16)



### Output Filter and Services

At the beginning of this slide, the packet's output interface is set to `gr-0/0/0.0`. Because no output policer or output filter is set on the `gr-0/0/0.0` interface, the router proceeds to processing the service filters for the `gr-0/0/0.0` interface.

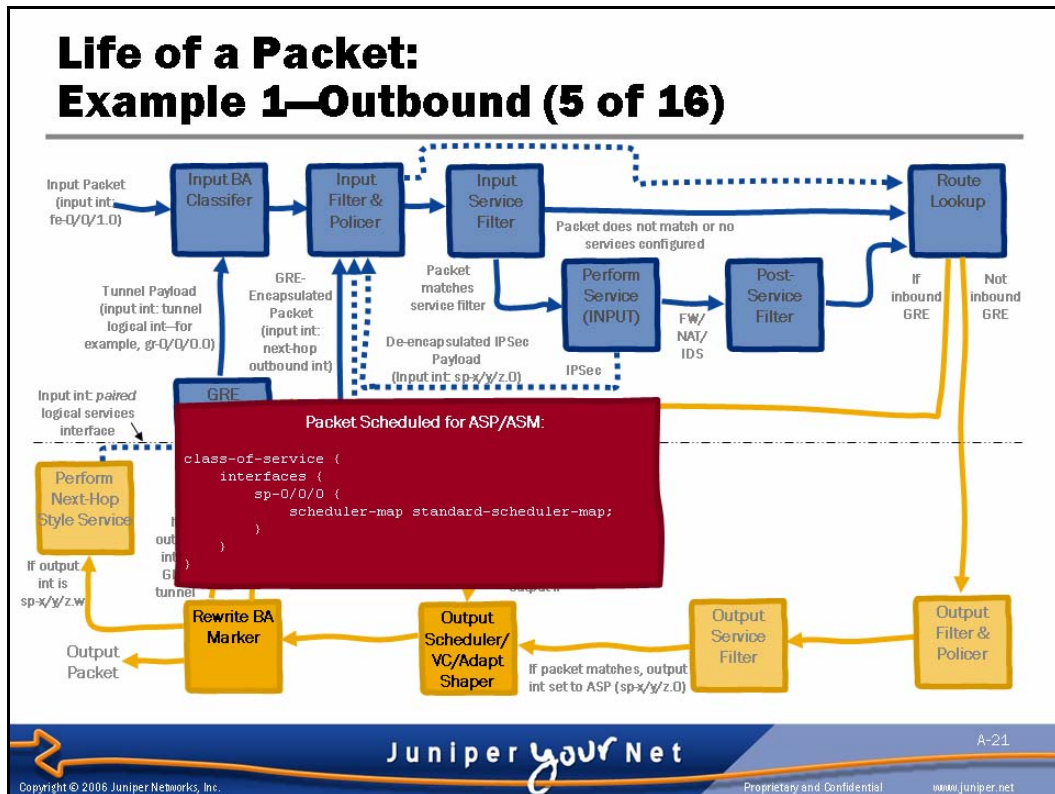
Even though no service filter is configured on the `gr-0/0/0.0` services, there is still a default filter. The default filter matches all unicast traffic and causes it to be serviced, while it causes all multicast traffic to be forwarded without passing through the services. You can see an indication of the default filter below (notice how there is an auto-generated filter for the service `sffw-outbound-only` configuration line):

```

lab@Montreal# run show firewall filter ?
Possible completions:
  <filtername>      Filter name
  __default_bpdu_filter__
  __service-sffw-outbound
  __service-sffw-outbound-no-gre
  __service-sffw-outbound-only
  __service-vpn-to-SiteA-match-vpn-to-SiteA
  __service-vpn-to-SiteA-match-vpn-to-SiteA-no-gre
  classify-packets
  counter          Counter name
  
```

Because the packet matches the service filter, the PFE changes its output interface to `sp-0/0/0.0` (Unit 0 of the service-interface configured for the service set).

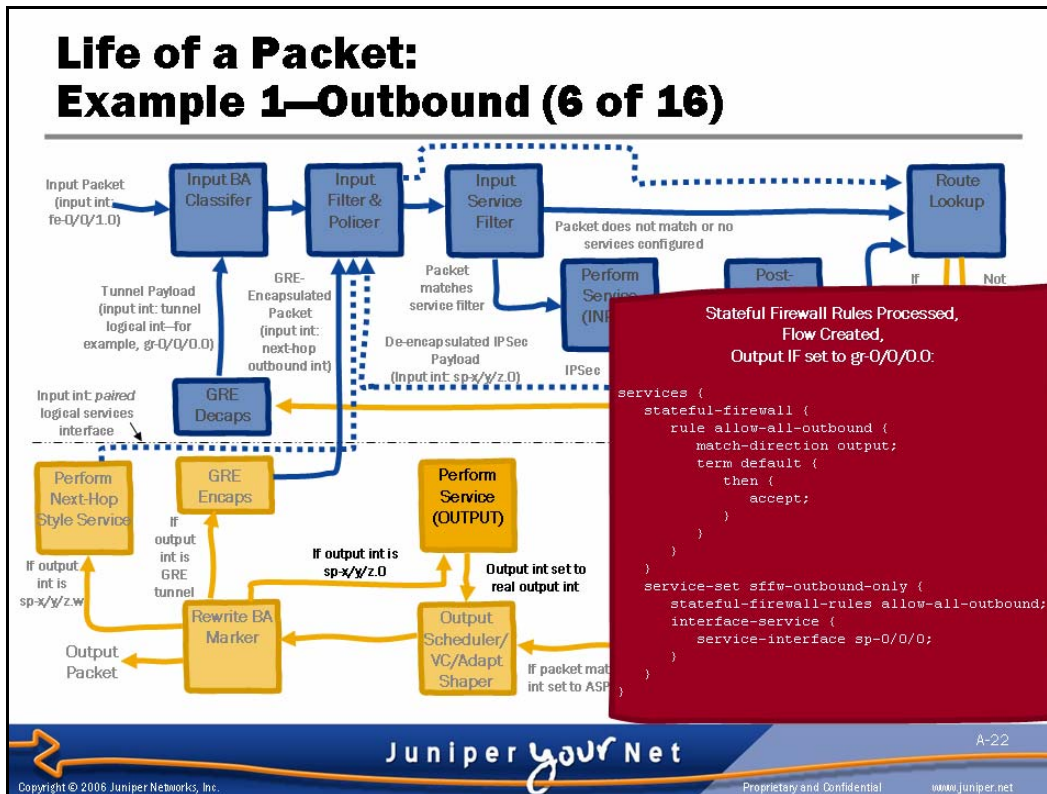




### CoS Processing (for `sp-0/0/0.0`)

The router next processes the packet through the configuration for the `sp-0/0/0.0` interface under `[edit class-of-service interfaces]`. Note that the router is now processing the packet with the output interface set to `sp-0/0/0.0`, so it is applying the configuration associated with that interface (rather than `gr-0/0/0.0`).

For the AS PIC to process the packet, the router must transmit the entire packet to the AS PIC. The router will use `standard-scheduler-map` to schedule the traffic for transmission from the FPC to the PIC. Even though traffic might appear to be staying *inside* the router when it is sent to the AS PIC, the router still applies CoS configuration when transmitting packets to the AS PIC just as if they were being transmitted out any other interface. As with any other interface, you must apply a scheduler map to the services interface(s) when your configuration uses any queues other than the default queues. Failure to do so might result in the router dropping traffic due to a lack of buffers for the nondefault queues.

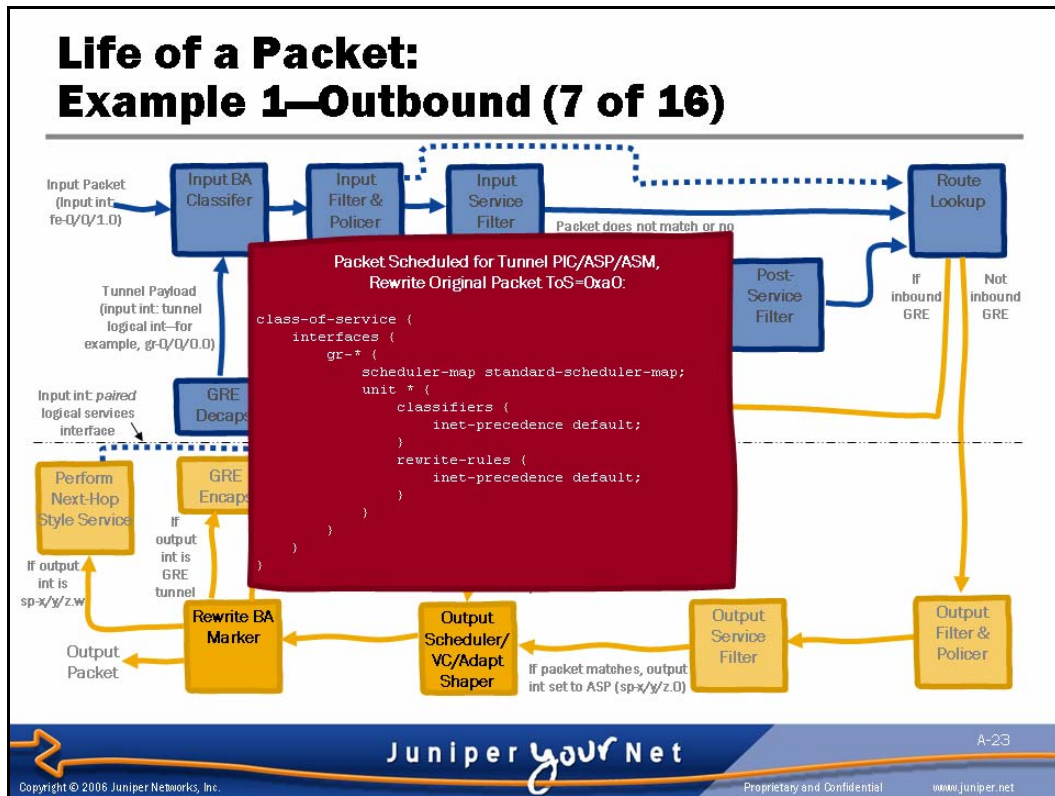


### Services Performed

The router now sends the entire packet to the ASP, which performs the referenced service(s). In this case, the ASP processes the packet through the stateful firewall rules. It finds that it is accepted by the rule. (Because the packet is outbound and the match direction is output, it matches the rule and is accepted. Had the packet been inbound, it would not have matched the rule and would have been dropped.)

The ASP creates appropriate entries in its flow table and then sends the packet back to the PFE with the output interface reset to `gr-0/0/0.0` (the original output interface).

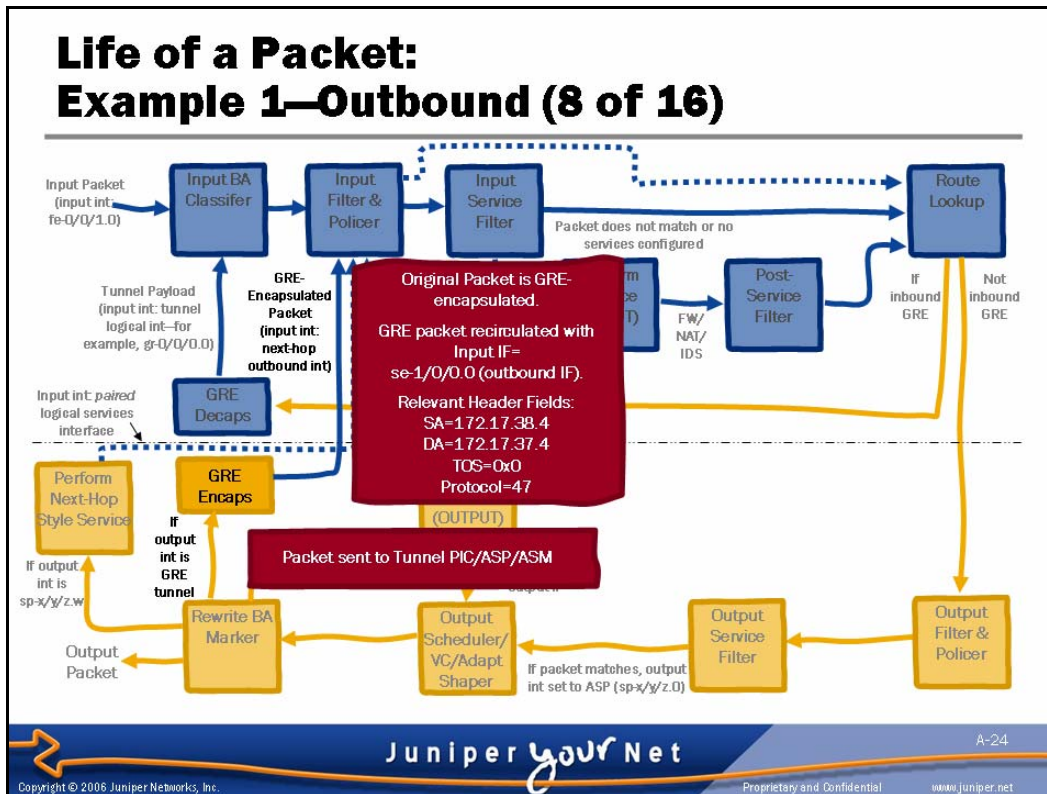




### CoS Processing (for `gr-0/0/0.0`)

The router next processes the packet through the configuration for the `gr-0/0/0.0` interface under `[edit class-of-service interfaces]`. As with the services interface, the router still applies CoS configuration when transmitting packets to the AS PIC just as if they were being transmitted out any other interface.

The router will use *standard-scheduler-map* to schedule the traffic to the PIC, and it will rewrite the ToS bits in the original packet to set the IP Precedence correctly. Because this packet is still in the expedited-forwarding forwarding class, the ToS field is 0xa0.

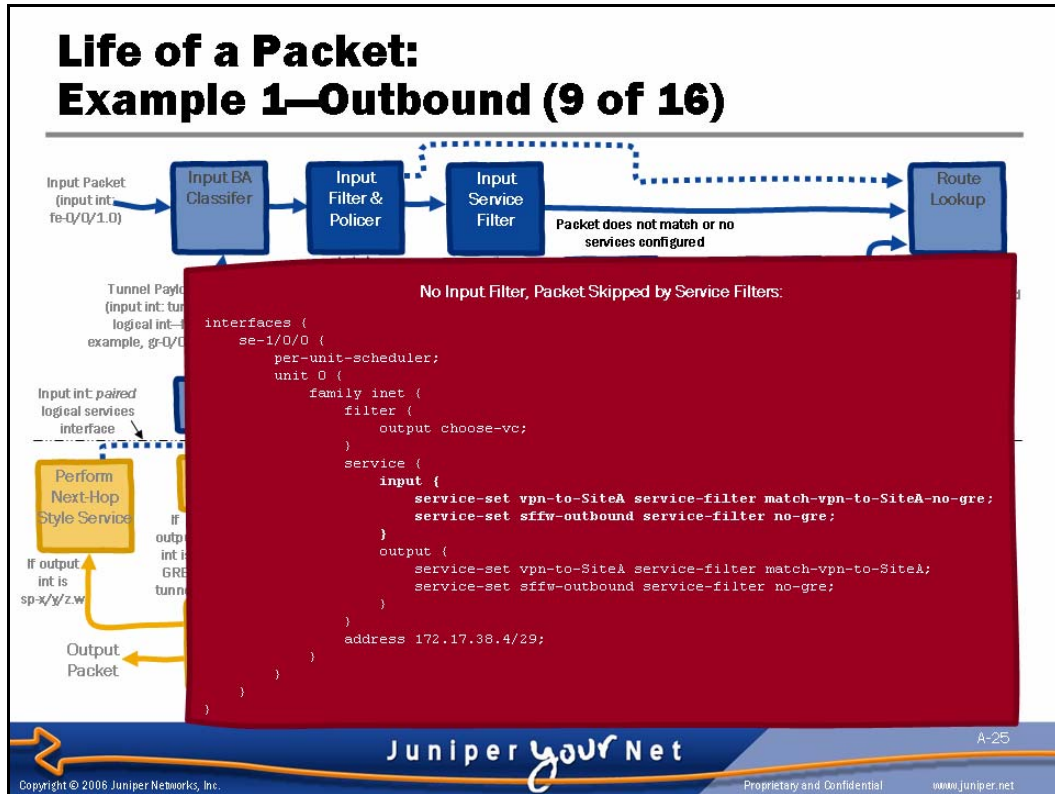


## GRE Encapsulation

This slide illustrates what is, perhaps, one of the most confusing quirks about tunnels in JUNOS software: the way the encapsulated packet is recirculated.

After the router completes CoS processing for `gr-0/0/0.0`, the entire packet is sent to the Tunnel PIC or ASP for encapsulation. The PIC performs the GRE encapsulation. It does maintain the forwarding class of the packet, but it does *not* copy the ToS bits from the IP header of the original packet to the IP header of the GRE packet.

Once the GRE packet is ready, it is sent to the PFE for processing with the *input interface set to the next-hop outbound interface for the destination of the tunnel*. In this case, because the next hop to 172.17.37.4 (the destination of the tunnel) is se-1/0/0.0, the GRE packet's input interface will be set to se-1/0/0.0. You must understand this process because it has important implications for input filters and service filters (as you will see).

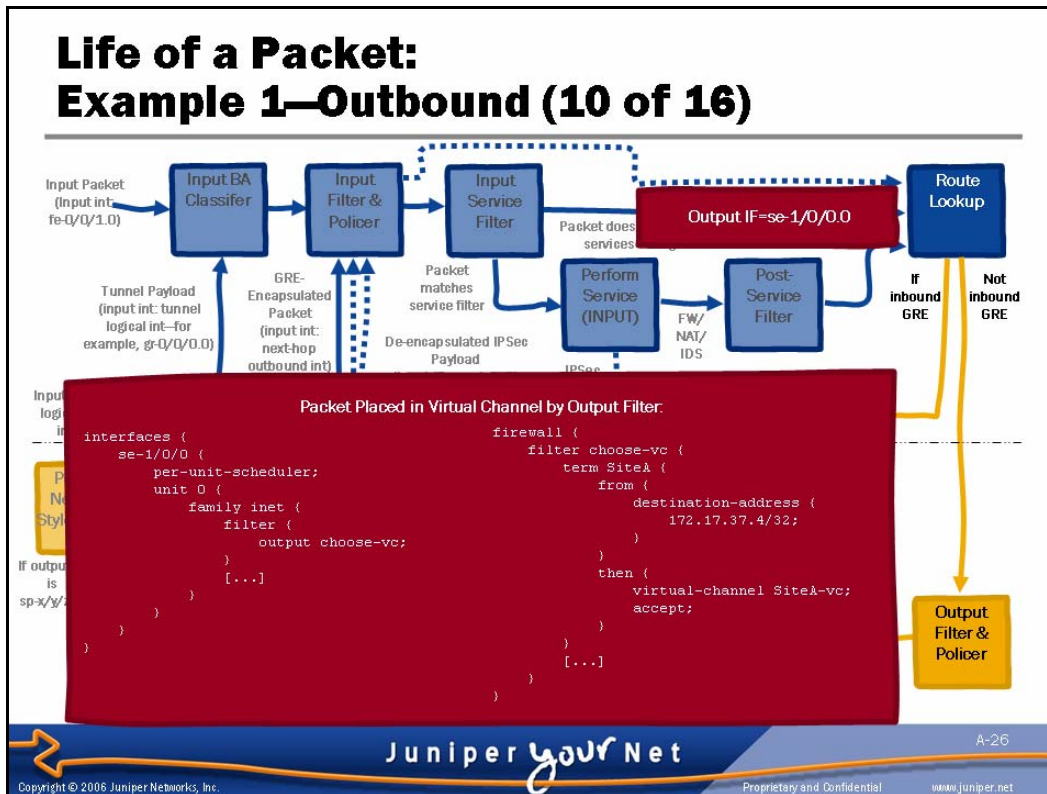


### Input Filter and Service Filter (for se-1/0/0.0)

The router now processes the GRE-encapsulated packet through the input policer, filters and service filters for se-1/0/0.0. No input policer or input filter is defined for se-1/0/0.0; however, if an input filter existed, it would need to permit these outbound GRE packets. If the input filter were to deny these GRE packets, they would be dropped by the PFE.

The service filters applied to the input services were written to prevent GRE packets from being processed by the input services. (You can see these service filters on page A-11.) If the service filters did cause the outbound GRE packets to be processed through the services, the ASP would drop the traffic (because it would not be permitted in the inbound direction by any of the rules).

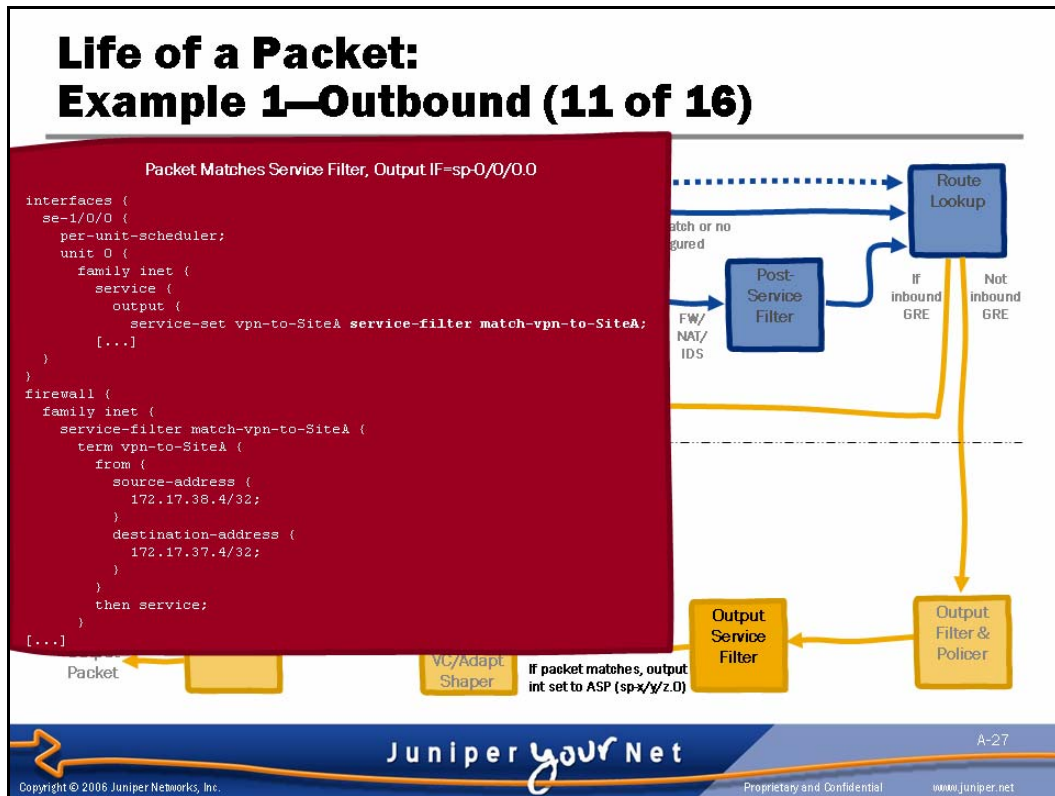
As you can see, you must take the recirculation of tunneled packets into account when configuring *input* policers, filters, and service filters for the *outbound* interface of the tunnel.



### Route Lookup and Output Filter

After processing the service filters, the PFE proceeds to perform a lookup in the forwarding table. In this case, the lookup returns `se-1/0/0.0` (the output interface for the destination of the tunnel).

The PFE then processes the output filter for `se-1/0/0.0`. The output filter places traffic destined to the remote site in a virtual channel called `SiteA-vc`. Because the GRE packets (rather than the original packets or ESP-encapsulated packets) are processed through this filter, this filter must only match on the GRE tunnel endpoint in order to match all traffic destined for SiteA via the VPN.

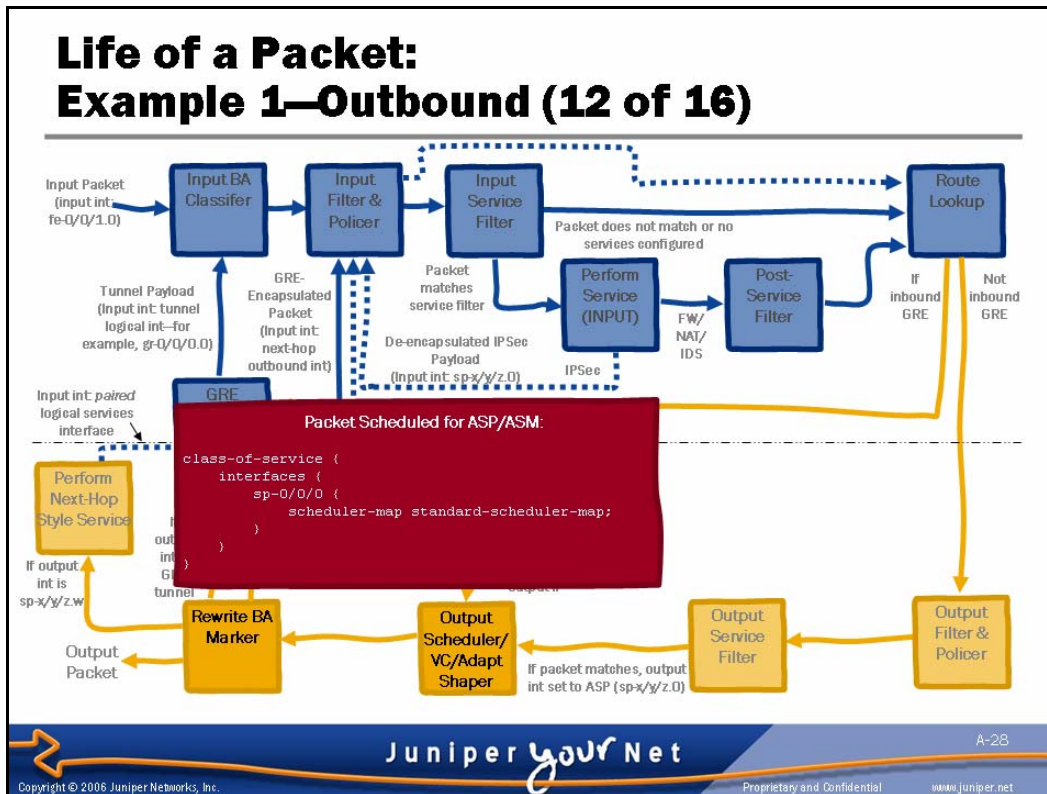


### Output Service Filter

The PFE next processes the packet against the output service filter for `se-1/0/0.0` (the current output interface). The packet matches the first service filter (`match-vpn-to-SiteA`), so the associated service is performed. The service filters are evaluated in order, and the router performs the first service with a matching service filter. Therefore, as soon as a match is found, further processing of service filters ceases.

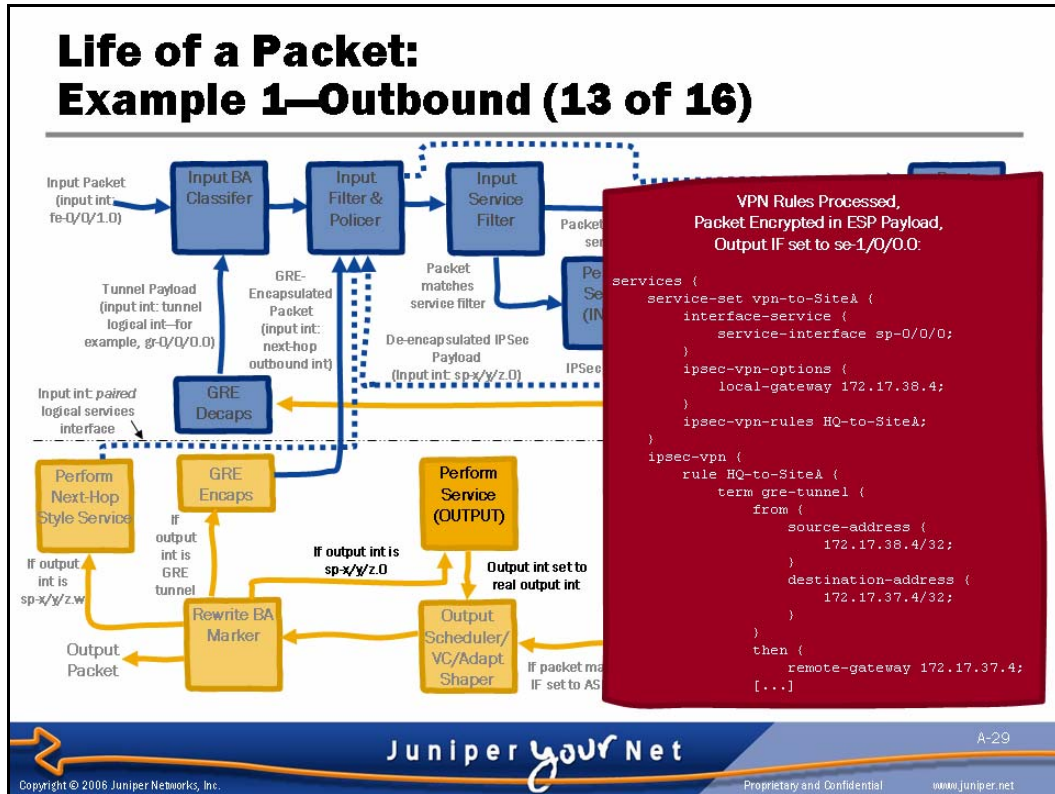
Because the packet matches the service filter, the PFE changes its output interface to `sp-0/0/0.0` (Unit 0 of the service interface configured for the service set).





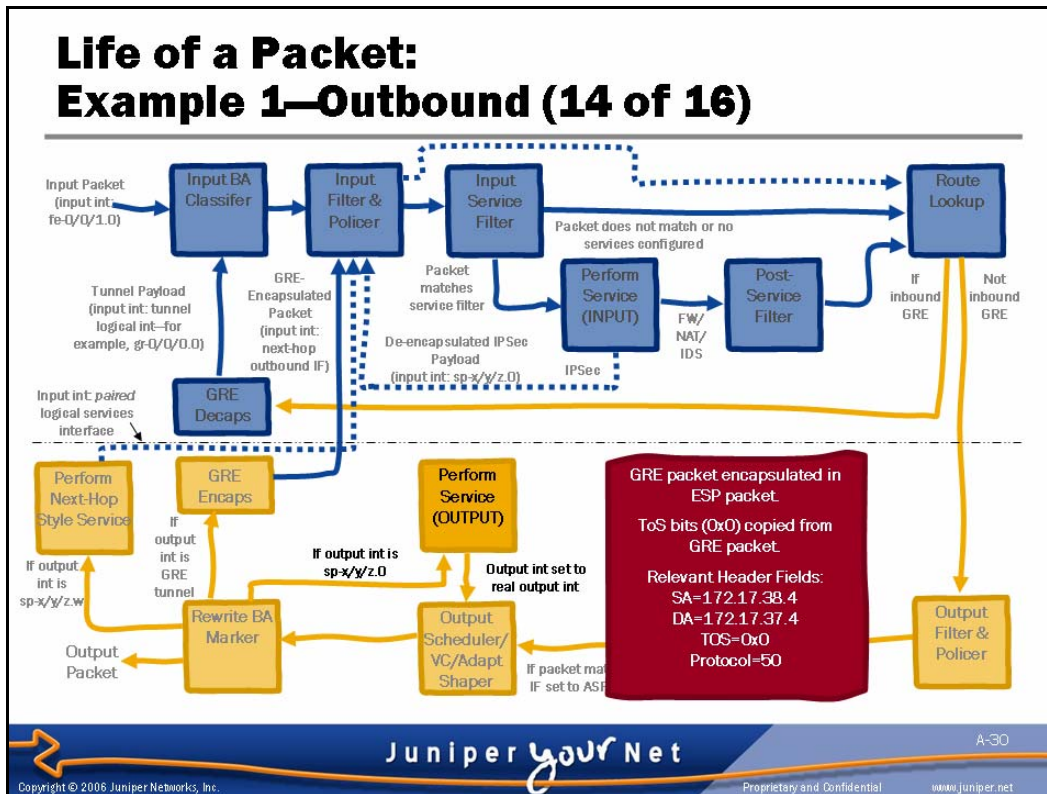
### CoS Processing (for `sp-0/0/0.0`)

The router next processes the packet through the configuration for the `sp-0/0/0.0` interface under `[edit class-of-service interfaces]`. The router uses *standard-scheduler-map* to schedule the traffic for transmission from the FPC to the PIC.



### Services Performed

After the PFE sends the entire packet to the ASP, the ASP processes the packet through the rules associated with the service set and finds that the packet matches a VPN rule. The VPN rule calls for the packet to be encapsulated and encrypted within an IPsec Encapsulating Security Payload (ESP) packet.



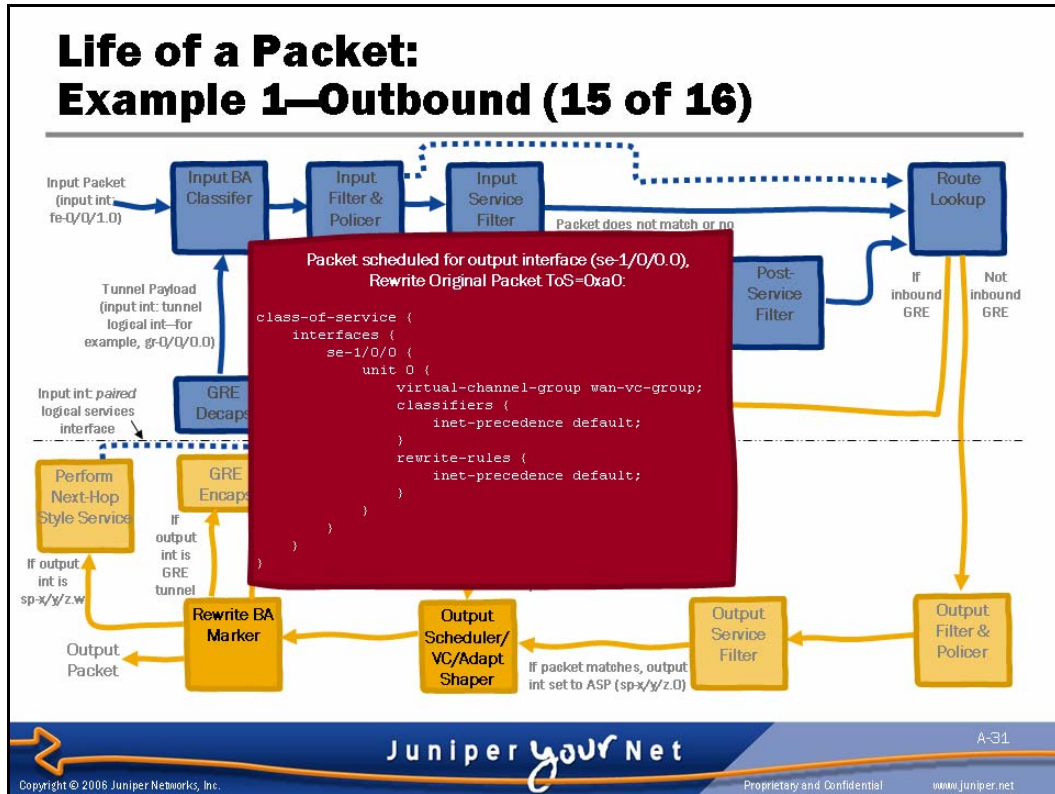
### Encrypted Packet

This slide shows some of the details of the ESP packet. The ASP encrypts the GRE packet within an ESP packet. It sends the ESP packet with a source address of 172.17.38.4 (the configured local-gateway) and a destination address of 172.17.37.4 (the configured remote-gateway). The protocol of the ESP packet is 50 (the protocol assigned for ESP).

Additionally, the ASP copies the ToS bits from the GRE packet (0x0) to the ToS bits of the ESP packet. This copying enables routers along the path that use a BA classifier to use the ToS bits to classify the ESP packet in the same way as they would the encrypted GRE packet, even though they will not have access to read the ToS bits of the encrypted GRE packet. Note that the ASP does not change the forwarding class association of the packet (currently, the expedited-forwarding forwarding class).

This new packet is then sent back to the PFE with the output interface set to se-1/0/0.0.

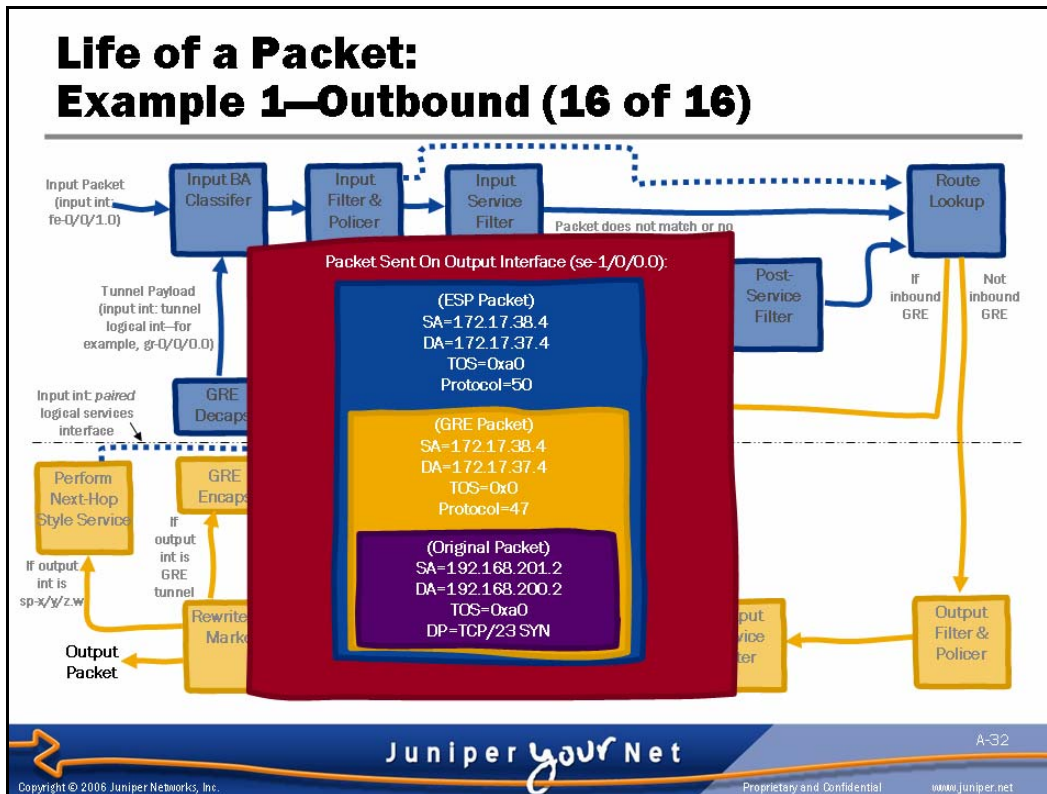




### CoS Processing (for `se-1/0/0.0`)

Once the PFE receives the packet from the ASP, it begins processing the CoS parameters for `se-1/0/0.0`. In this case, it schedules the packet for transmission according to the virtual channel configuration. This packet was assigned to the *SiteA-vc* virtual channel. (The CoS information is maintained with a packet throughout its processing.)

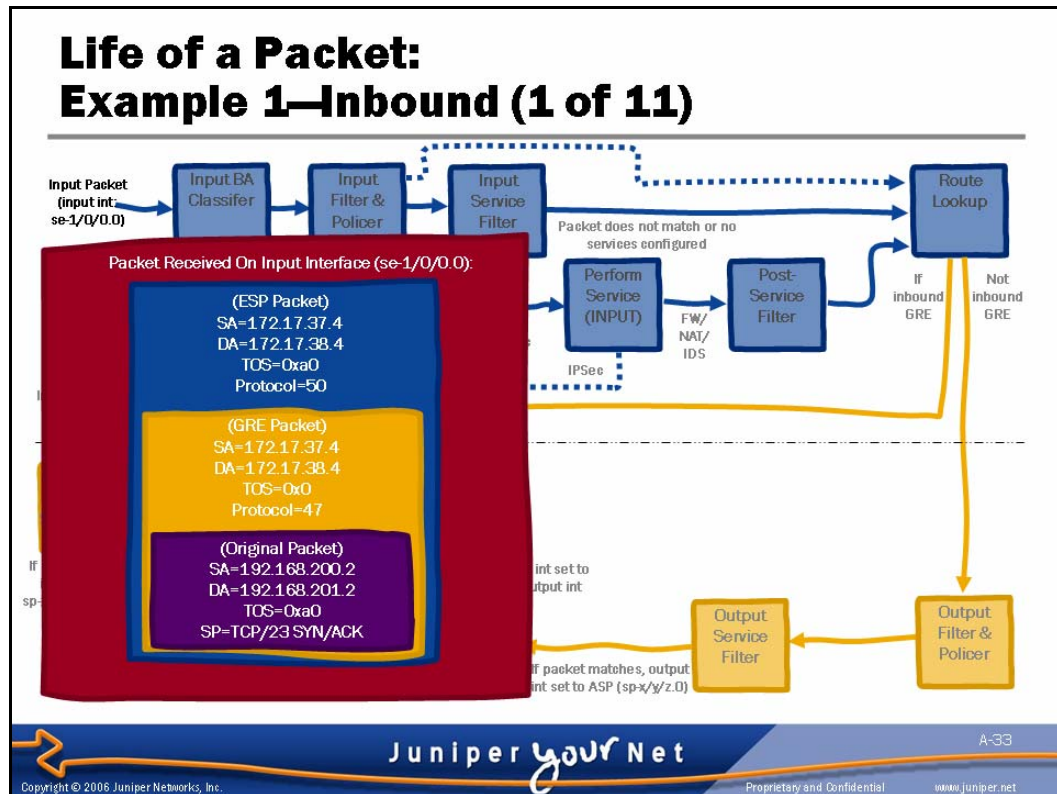
Also, as the packet is transmitted, the router rewrites the ToS bits of the ESP packet's IP header to `OxaO` (the appropriate value for the expedited-forwarding forwarding class).



### Packet Transmitted

After this entire process is completed, the PIC transmits the packet onto the wire.

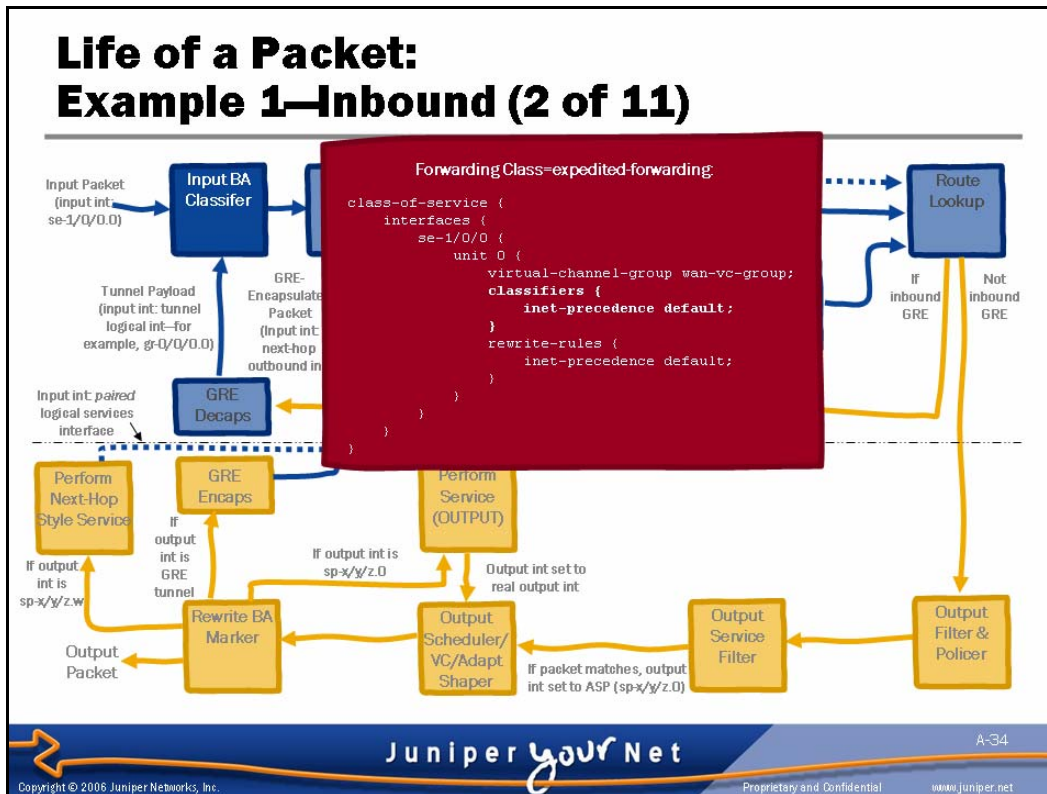
The transmitted packet is represented on the slide. The original packet (changed only by having its ToS bits set to 0xa0) is encapsulated within a GRE packet. The GRE packet is encrypted and encapsulated within an ESP packet. The ESP packet is sent on the wire with its ToS bits set to 0xa0 by a rewrite rule.



### Received Packet

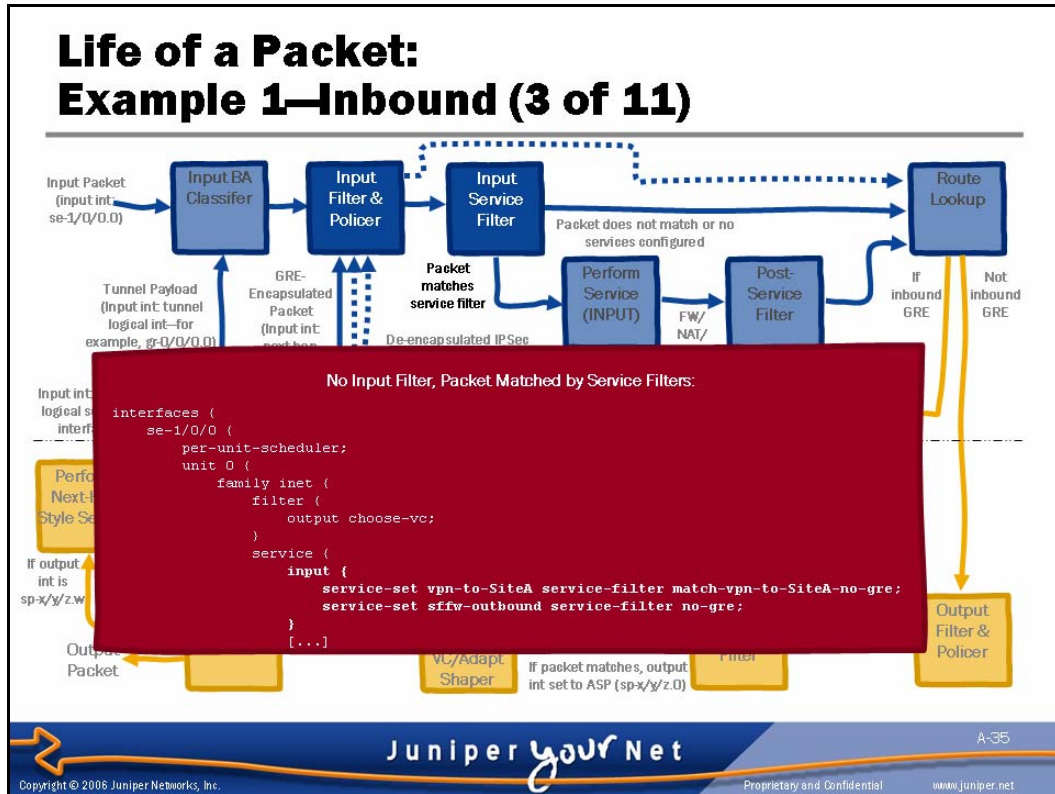
Once the remote Telnet server (192.168.200.2) receives the SYN from 192.168.201.2, it responds with a SYN/ACK. When that packet is received by HQ-1, it is encapsulated within a GRE packet, which is itself encrypted and encapsulated within an ESP packet (as shown on the slide).

We now examine how the router processes this packet. We must examine both directions because some differences exist between the inbound and outbound packet processing.



### BA Classifier (se-1/0/0.0)

The router begins by processing the BA classifier for se-1/0/0.0 (the input interface). The classifier causes this packet to be assigned the expedited-forwarding class.

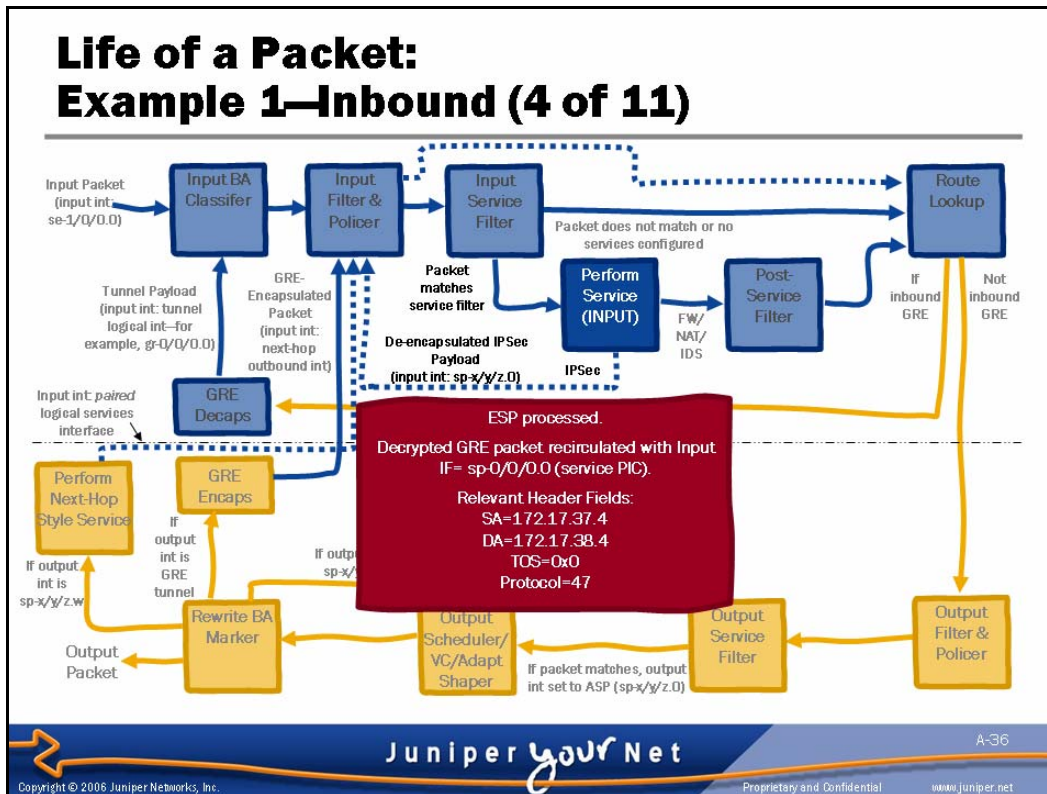


### Input Filter and Service Filters

No input policer or input filter is configured on `se-1/0/0.0`; however, there are service filters. The packet matches the first service filter (`match-vpn-to-SiteA-no-gre`) and is sent to the appropriate ASP for processing.

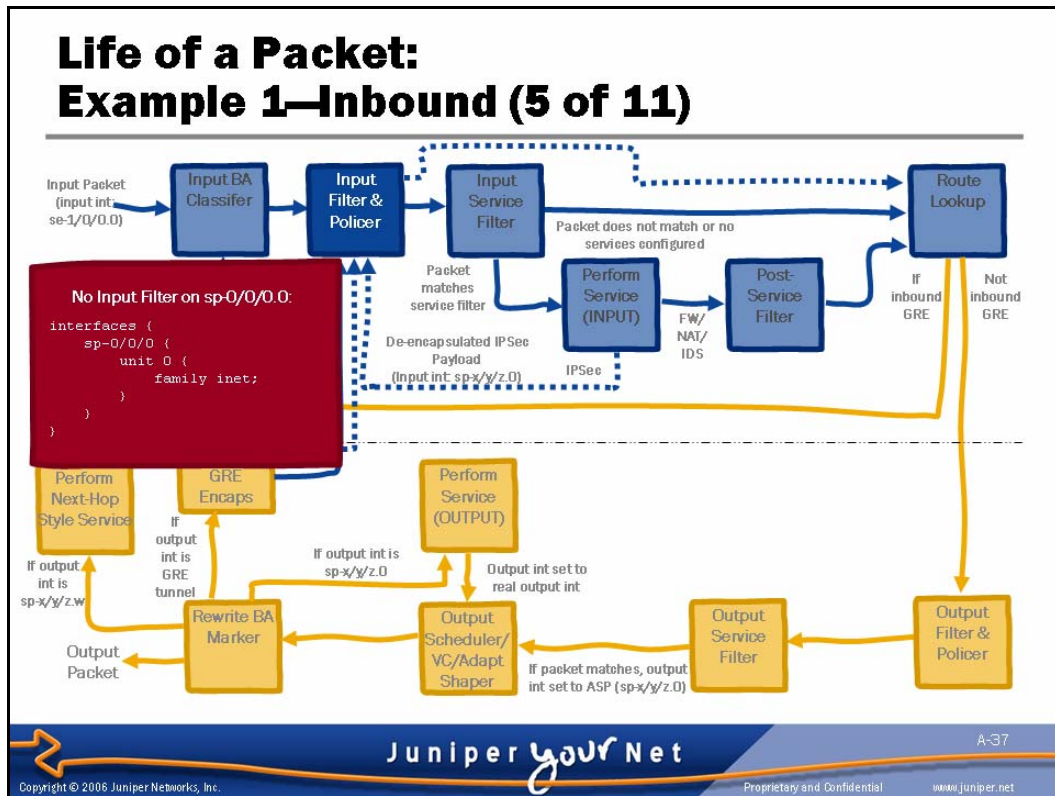
Note that for inbound interface-style IPsec ESP packets only, there is an alternate path to the service being performed. Because it is possible that the return traffic for the VPN will enter using a different interface than the outbound interface, the router installs a `/72` entry in the forwarding table that matches exactly the source address of the `remote-peer`, the destination address of the `local-peer` and IP protocols 50 and 51 (ESP and AH, respectively). In those cases, the packet is processed like any other input packet (including passing through any other services, if configured) until it reaches the `route lookup` function, when the PFE sends it to the appropriate ASP for further processing. At that point, it resumes processing at the point where the next slide begins.





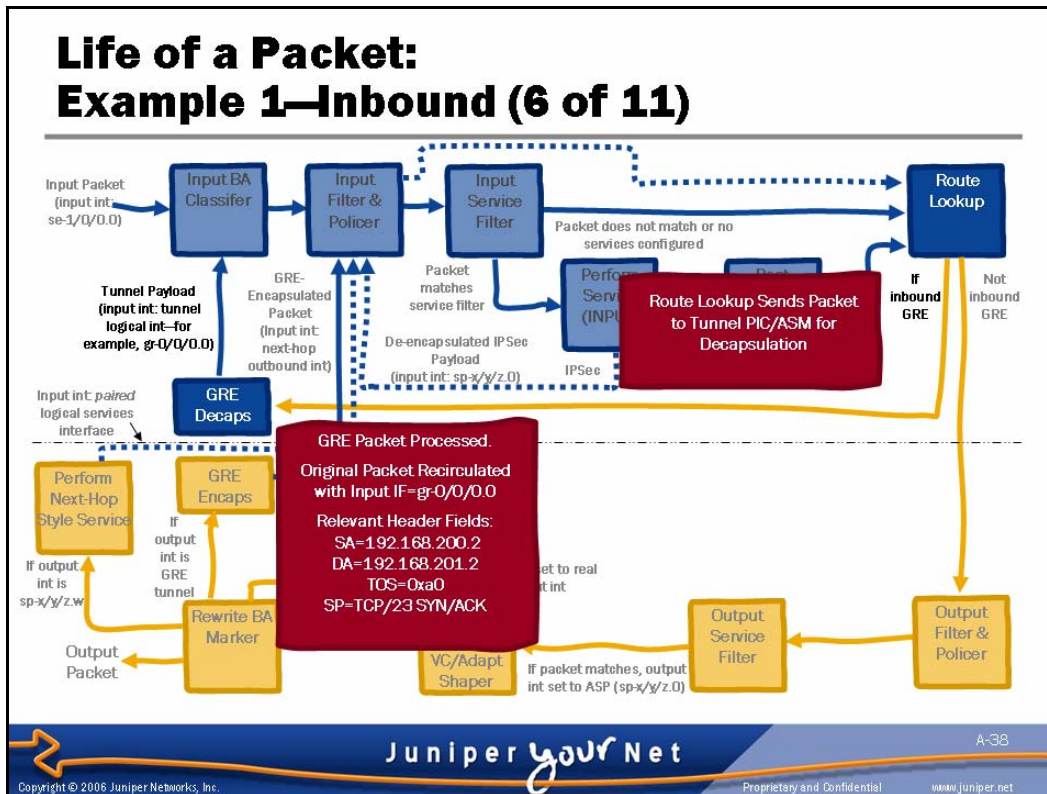
### Service Processing

The ASP processes the ESP packet and decrypts the encapsulated GRE packet. The ToS bits from the IP header of the ESP packet are *not* copied to the IP header of the decrypted GRE packet, but the ASP does maintain the packet's forwarding class association (currently, the expedited-forwarding forwarding class). The GRE packet is then sent back to the PFE with an input interface of `sp-0/0/0.0` (Unit 0 of the service interface configured for the service set).



### Input Filter Processing (for `sp-0/0/0.0`)

The PFE processes the GRE packet against the input policer and input filter on `sp-0/0/0.0`. However, because none is configured, the PFE proceeds directly to a route lookup. (The dotted line that skips the input service filter represents it not being possible to configure services on service interfaces.)



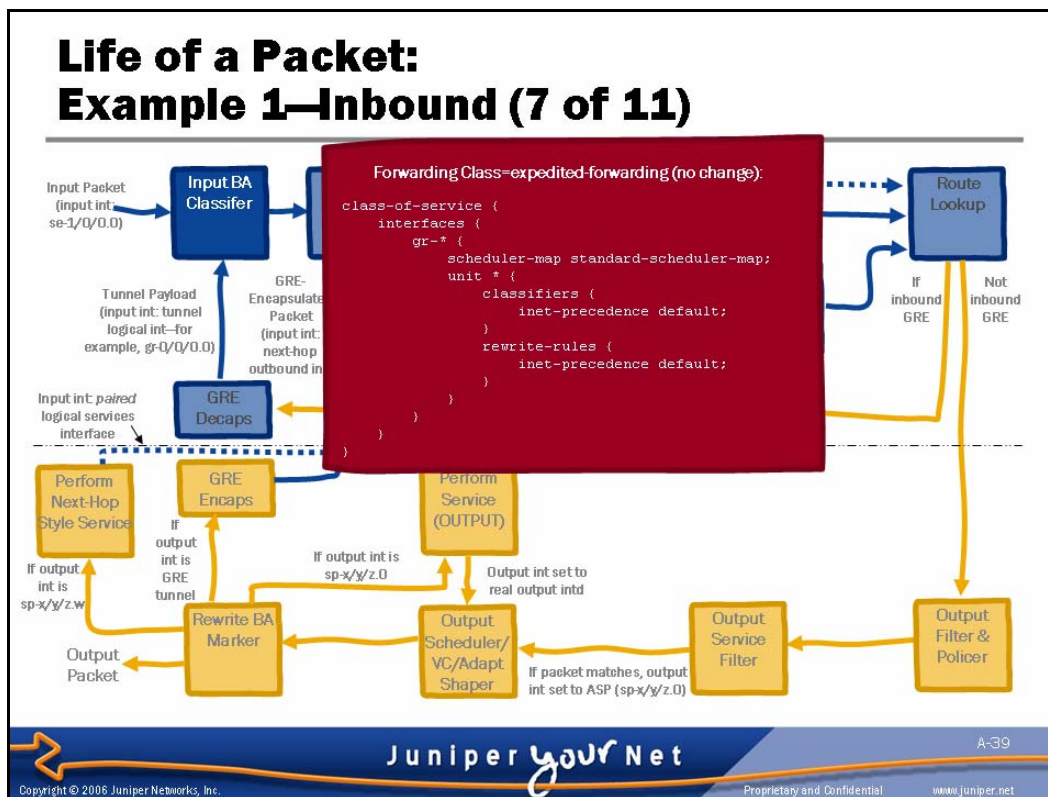
### GRE Decapsulation

The PFE performs a route lookup in the forwarding table and finds that the packet matches a /72 entry for a source address of the remote tunnel endpoint, a destination address of the local tunnel endpoint, and a protocol number of 47 (the protocol number used for GRE). This process causes the entire packet to be sent to the Tunnel PIC or ASP for processing:

```
lab@Montreal# run show route forwarding-table
Routing table: inet
Internet:
Destination          Type RtRef Next hop          Type Index NhRef Netif
[...]
172.17.38.0/29        intf    0 ff.3.0.21          ucst   322    3 so-0/1/2.0
172.17.38.0/32        dest    0 172.17.38.0         recv   324    1 so-0/1/2.0
172.17.38.4/32        intf    0 172.17.38.4         locl   321    1
172.17.38.4.172.17.37.4.47/72
dest 0 locl 343 1
172.17.38.4.172.17.37.4.50/72
user 0 service 331 3
172.17.38.4.172.17.37.4.51/72
user 0 service 331 3
[...]
```

When the packet is decapsulated, it is sent to the PFE with a source interface of the tunnel's logical interface (`gr-0/0/0.0`). The IP header of the encapsulated packet is maintained (including the original ToS bits), and the forwarding class is maintained.

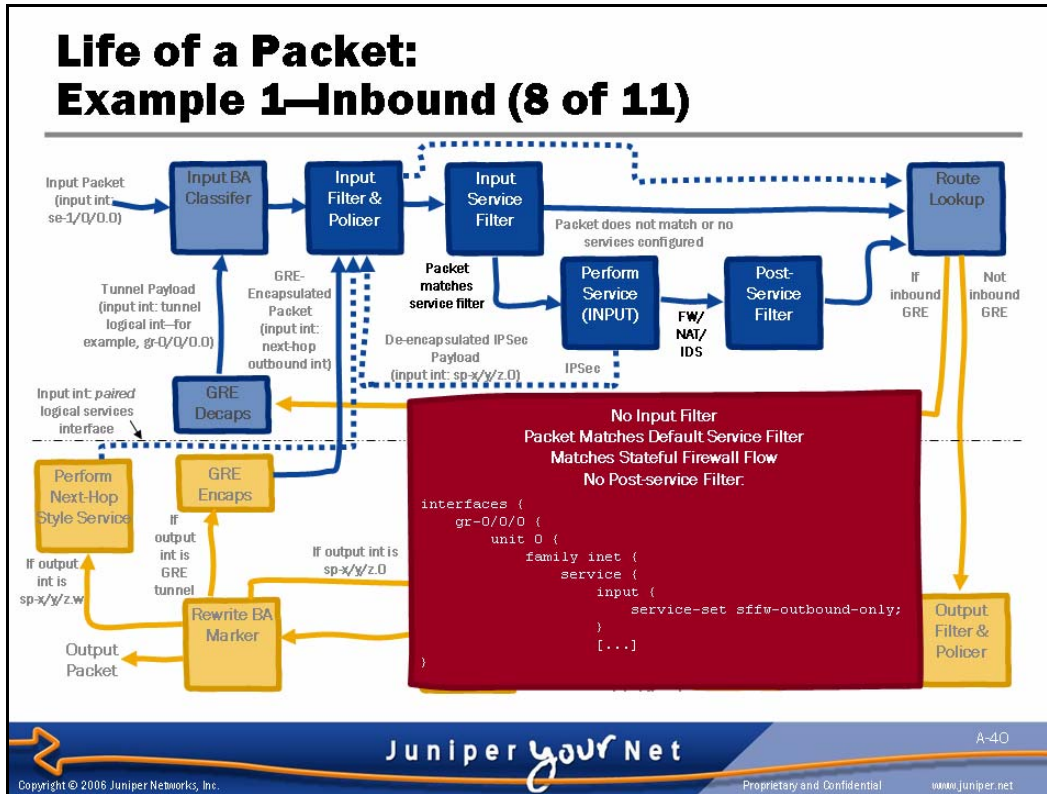




### BA Classifier (for gr-0/0/0.0)

At this point, the router processes the de-encapsulated packet through the BA classifier configured for gr-0/0/0.0. Because the IP header of the encapsulated packet had the ToS field set to 0x0, the inet-precedence default classifier sets the forwarding class to expedited-forwarding, which is the same as the previous forwarding class of the packet.

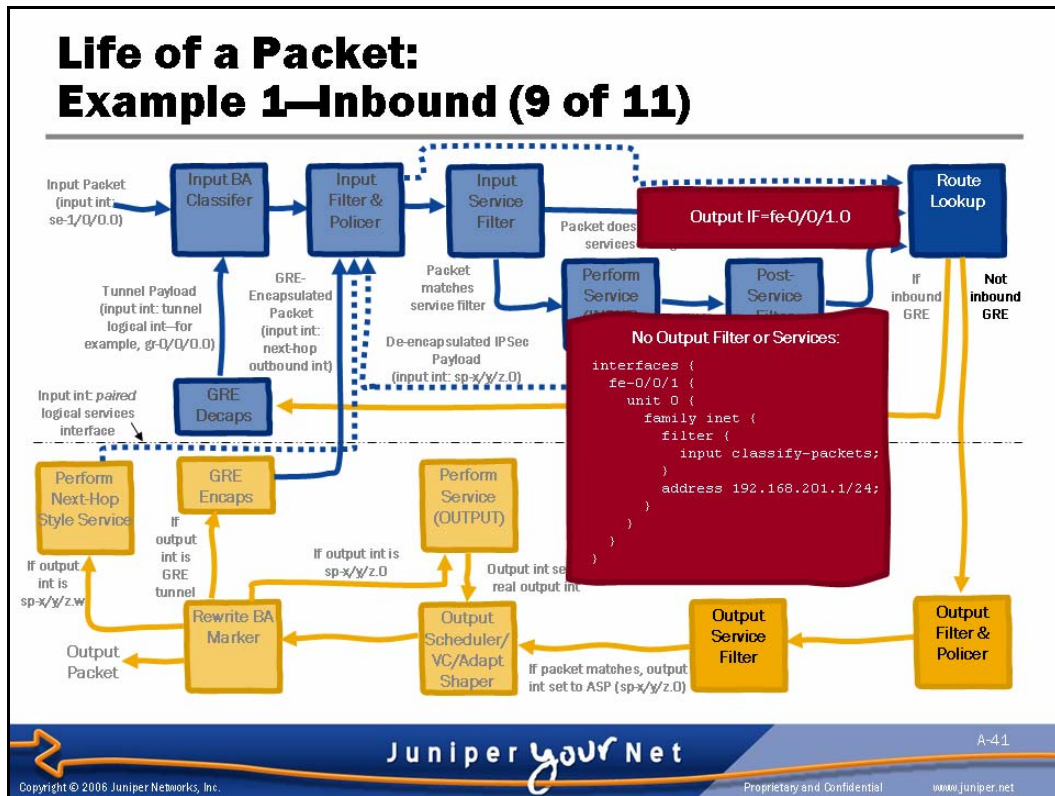
However, as was explained on page A-16, if no classifier were configured on gr-0/0/0.0, the router would still have used a default classifier. This default classifier would have set this packet in the best-effort forwarding class because that classifier maps a ToS field of 0x0 (IP precedence bits 101) to the best-effort forwarding class. Also, had the decapsulated packet had a different value in the ToS field of its IP header (or the classifier configured on gr-0/0/0.0 had returned a different forwarding class for any reason), the router would have changed the forwarding class of the packet to the value returned by the BA classifier configured for gr-0/0/0.0.



### Inbound Filter and Services (for `gr-0/0/0.0`)

Because no input policer or input filter is configured for `gr-0/0/0.0`, the PFE processes the packet through the inbound service filter. Because the packet is a unicast packet, it matches the default service filter and is sent to the ASP for processing. The ASP processes the rules for the service set and finds a matching flow in its stateful firewall flow table. Therefore, the packet is accepted. Because this service set does not include Network Address Translation (NAT) or intrusion detection service (IDS) configuration, those services are not performed, and the packet is transmitted back to the PFE.

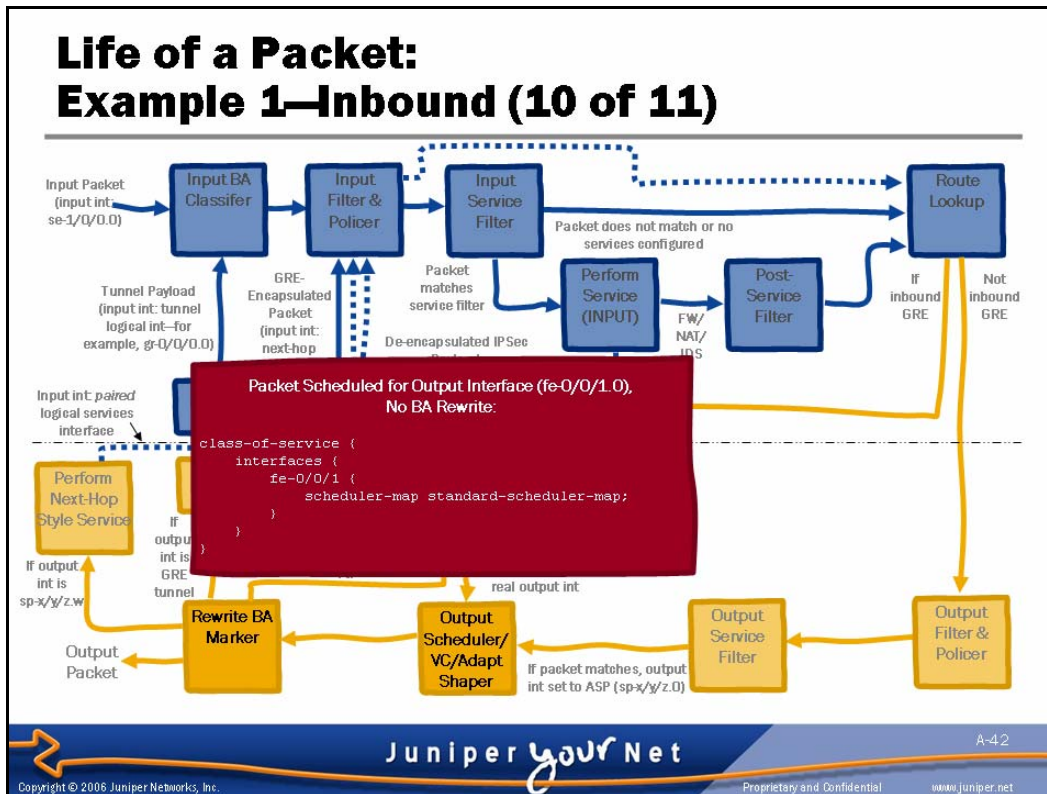
The PFE would next perform a post-service filter; however, none is configured, so the packet proceeds to a route lookup.



## Route Lookup

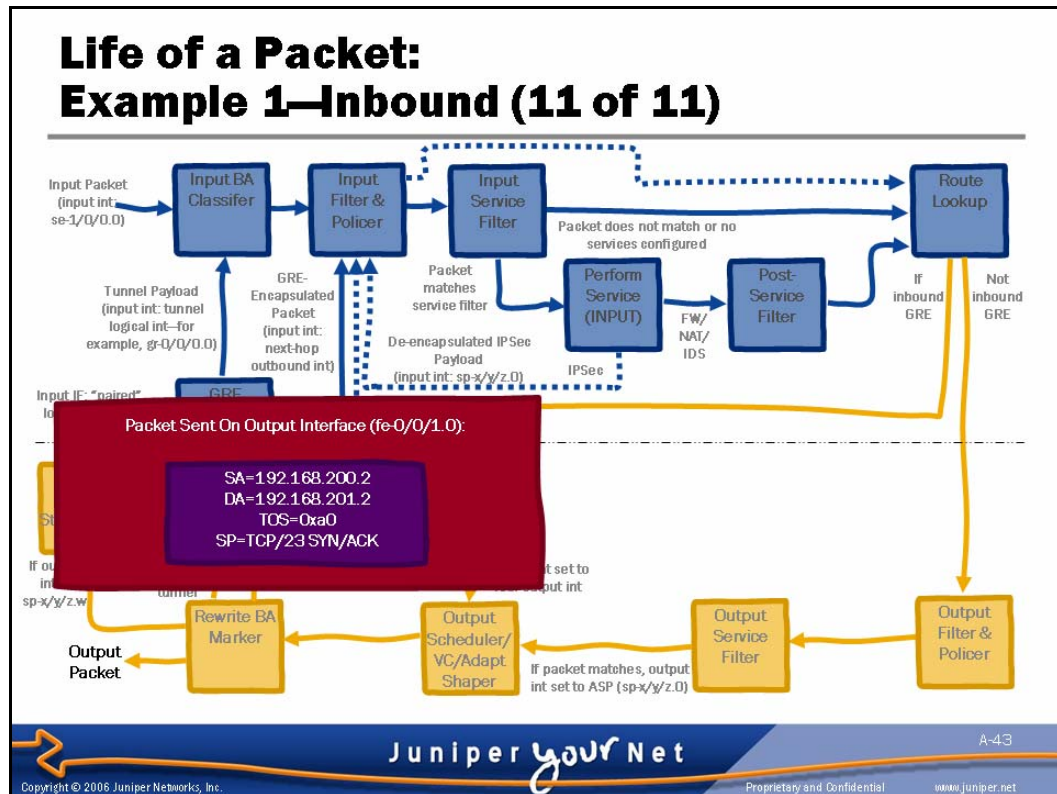
The PFE looks up the destination address in the forwarding table and determines the output interface to be `fe-0/0/1.0`.

It next processes the packet through the output filters and service filters for the output interface, if any are configured. Because no output filter or service is configured on this interface, the packet moves to the CoS processing.



### CoS Output Processing

The packet is scheduled for transmission according to the scheduler map configured for fe-0/0/1.0. Because no rewrite rule is configured for fe-0/0/1.0, the original ToS field of the IP header is maintained.



### Packet Transmitted

After this entire process is completed, the PIC transmits the packet onto the wire.

The ESP packet was processed. The enclosed GRE packet was decrypted and processed. The de-encapsulated original packet was processed and transmitted to its destination.

As was indicated previously, this example is purposely complex and shows many revolutions of packets through a router. Not all packets will pass through this many steps. In addition, on the M-series and T-series routers, most of the processing shown is performed in hardware at line-rate and is quite fast.

## Agenda: Life of a Packet

---

- Overview
- Example 1: Interface-Style Service Sets
- Example 2: Next-Hop-Style Service Sets

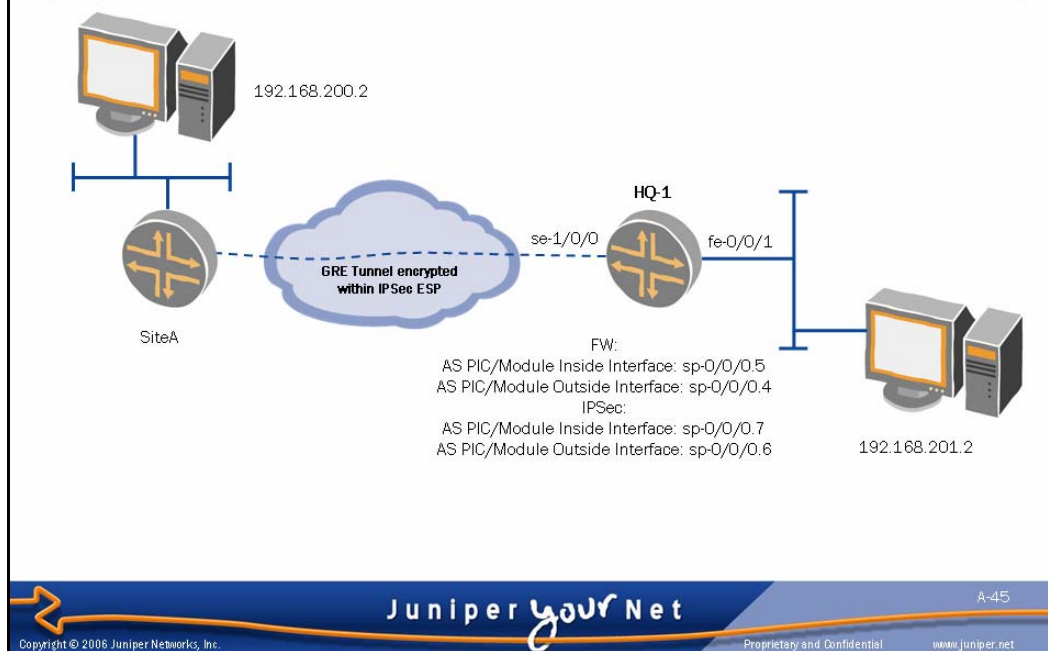


### Example 2: Next-Hop-Style Service Sets

The slide highlights the topic we discuss next.



## Example 2: Next-Hop-Style Service Set



### Example 2: Overview and Configuration

This example will demonstrate using next-hop-style service sets to implement a stateful firewall, encrypt a GRE tunnel within a VPN, and perform CoS on this traffic.

We again examine packet flow through the router on the right side of the slide (HQ-1). This router has a GRE tunnel established between it and the branch router (on the left). The GRE tunnel uses a second IP on the `se-1/0/0.0` interface as its local endpoint. This GRE tunnel is encrypted within an IPsec VPN. The VPN uses the preferred IP address on the `se-1/0/0.0` interface as its local endpoint. (Recall that when using next-hop-style service sets, the GRE endpoints should be different from the VPN endpoints for reasons that will become obvious as we look at this example.)

Further, the corporate headquarters' LAN (192.168.201.0/24) is secured by a stateful firewall that allows all outbound connections from the LAN, but denies all inbound connections. (Note that this is implemented slightly differently than in Example 1.)

To satisfy its performance requirements, this company has implemented CoS throughout its network. Traffic is classified by a multfield classifier as it enters the `fe-0/0/1.0` interface, and that classification is maintained by rewriting and processing the IP precedence BA. To not overrun their slower branch office connection, traffic heading to the branch office (SiteA) is limited to 768 kbps using a virtual-channel configuration.

Although this is a purposely complicated example, it should show the differences between the way the router processes interface-style and next-hop-style services.

*Continued on next page.*

**Example 2: Overview and Configuration (contd.)**

The configuration of HQ-1:

```

version 8.0R1.9;
system {
    host-name HQ-1;
    root-authentication {
        encrypted-password "$1$KI99zGk6$MbYFuBbpLffu9tn2.sI7l1"; ## SECRET-DATA
    }
    login {
        user lab {
            uid 2000;
            class super-user;
            authentication {
                encrypted-password "$1$84J5Maes$cni5Hrazbd/IEHr/50oY30"; ##
SECRET-DATA
            }
        }
    }
    services {
        ftp;
        ssh;
        telnet;
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any notice;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
    gr-0/0/0 {
        unit 0 {
            tunnel {
                source 172.17.38.5;
                destination 172.17.37.5;
            }
            family inet {
                address 192.168.25.130/30;
            }
        }
    }
}

```

*Continued on next page.*



**Example 2: Overview and Configuration (contd.)**

```

sp-0/0/0 {
  unit 0 {
    family inet;
  }
  unit 4 {
    family inet;
    service-domain outside;
  }
  unit 5 {
    family inet;
    service-domain inside;
  }
  unit 6 {
    family inet;
    service-domain outside;
  }
  unit 7 {
    family inet;
    service-domain inside;
  }
}
fe-0/0/1 {
  unit 0 {
    family inet {
      filter {
        input classify-packets;
      }
      address 192.168.201.1/24;
    }
  }
}
se-1/0/0 {
  per-unit-scheduler;
  unit 0 {
    family inet {
      filter {
        output choose-vc;
      }
      address 172.17.38.4/29 {
        preferred;
      }
      address 172.17.38.5/29;
    }
  }
}
}
routing-options {
  static {
    route 0.0.0.0/0 next-hop 172.17.38.1;
    route 192.168.200.0/24 next-hop gr-0/0/0.0;
  }
}

```

*Continued on next page.*

**Example 2: Overview and Configuration (contd.)**

```

route 192.168.201.0/24 next-hop sp-0/0/0.4;
route 172.17.37.5/32 next-hop sp-0/0/0.7;
}
}
class-of-service {
  virtual-channels {
    SiteA-vc;
    default-vc;
  }
  virtual-channel-groups {
    wan-vc-group {
      SiteA-vc {
        scheduler-map standard-scheduler-map;
        shaping-rate 768k;
      }
      default-vc {
        scheduler-map standard-scheduler-map;
        default;
      }
    }
  }
}
interfaces {
  gr-* {
    scheduler-map standard-scheduler-map;
    unit * {
      classifiers {
        inet-precedence default;
      }
      rewrite-rules {
        inet-precedence default;
      }
    }
  }
  sp-0/0/0 {
    scheduler-map standard-scheduler-map;
  }
  se-1/0/0 {
    unit 0 {
      virtual-channel-group wan-vc-group;
      classifiers {
        inet-precedence default;
      }
      rewrite-rules {
        inet-precedence default;
      }
    }
  }
  fe-0/0/1 {
    scheduler-map standard-scheduler-map;
  }
}
scheduler-maps {

```

*Continued on next page.*

**Example 2: Overview and Configuration (contd.)**

```

standard-scheduler-map {
    forwarding-class network-control scheduler my-nc-sched;
    forwarding-class expedited-forwarding scheduler my-ef-sched;
    forwarding-class assured-forwarding scheduler my-af-sched;
    forwarding-class best-effort scheduler my-be-sched;
}
}
schedulers {
    my-be-sched {
        transmit-rate percent 10;
        buffer-size percent 10;
        priority low;
    }
    my-af-sched {
        transmit-rate percent 25;
        buffer-size percent 25;
        priority medium-low;
    }
    my-ef-sched {
        transmit-rate percent 50;
        buffer-size percent 50;
        priority medium-high;
    }
    my-nc-sched {
        transmit-rate percent 15;
        buffer-size percent 15;
        priority high;
    }
}
}
firewall {
    family inet {
        filter classify-packets {
            term interactive-applications {
                from {
                    protocol tcp;
                    source-port [ 22 23 ];
                }
                then {
                    forwarding-class expedited-forwarding;
                    accept;
                }
            }
            term sip {
                from {
                    protocol [ udp tcp ];
                    port 5060;
                }
                then {

```

*Continued on next page.*

**Example 2: Overview and Configuration (contd.)**

```

        forwarding-class assured-forwarding;
        accept;
    }
}
term rtp {
    from {
        protocol udp;
        port 16384-32767;
    }
    then {
        forwarding-class assured-forwarding;
        accept;
    }
}
term default {
    then accept;
}
}
}
filter choose-vc {
    term SiteA {
        from {
            destination-address {
                172.17.37.4/32;
            }
        }
        then {
            virtual-channel SiteA-vc;
            accept;
        }
    }
    term default {
        then accept;
    }
}
}
routing-instances {
    inside-zone {
        instance-type virtual-router;
        interface fe-0/0/1.0;
        interface sp-0/0/0.5;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop sp-0/0/0.5;
            }
        }
    }
}
}
services {
    stateful-firewall {

```

*Continued on next page.*

**Example 2: Overview and Configuration (contd.)**

```

rule allow-all-outbound {
    match-direction input;
    term default {
        then {
            accept;
        }
    }
}

service-set nh-sffw-outbound-only {
    stateful-firewall-rules allow-all-outbound;
    next-hop-service {
        inside-service-interface sp-0/0/0.5;
        outside-service-interface sp-0/0/0.4;
    }
}

service-set nh-vpn-to-SiteA {
    next-hop-service {
        inside-service-interface sp-0/0/0.7;
        outside-service-interface sp-0/0/0.6;
    }
    ipsec-vpn-options {
        local-gateway 172.17.38.4;
    }
    ipsec-vpn-rules HQ-to-SiteA;
}

ipsec-vpn {
    rule HQ-to-SiteA {
        term gre-tunnel {
            from {
                source-address {
                    172.17.38.5/32;
                }
                destination-address {
                    172.17.37.5/32;
                }
            }
            then {
                remote-gateway 172.17.37.4;
                dynamic {
                    ike-policy main_mode_ike_policy;
                    ipsec-policy dynamic_ipsec_policy;
                }
            }
        }
    }
    match-direction input;
}

ipsec {
    proposal esp_sha1_3des_ipsec_proposal {
        protocol esp;
    }
}

```

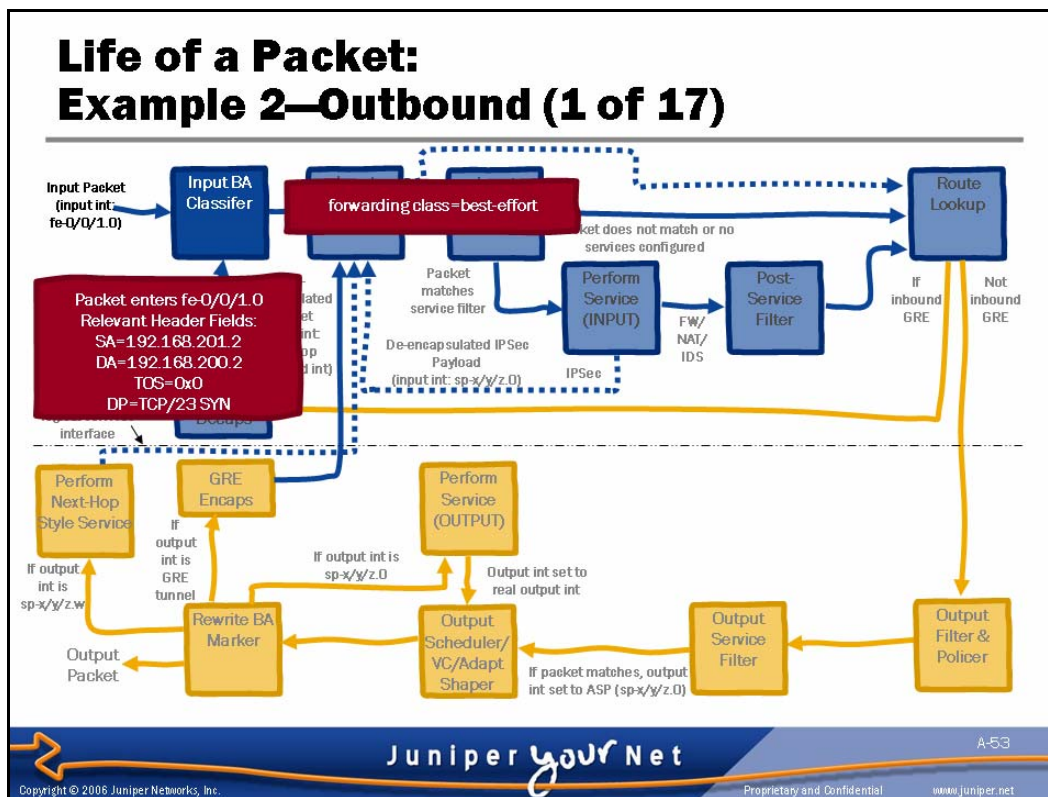
*Continued on next page.*

**Example 2: Overview and Configuration (contd.)**

```

        authentication-algorithm hmac-sha1-96;
        encryption-algorithm 3des-cbc;
    }
    policy dynamic_ipsec_policy {
        perfect-forward-secrecy {
            keys group2;
        }
        proposals esp_sha1_3des_ipsec_proposal;
    }
}
ike {
    proposal psk_sha1_3des_ike_proposal {
        authentication-method pre-shared-keys;
        authentication-algorithm sha1;
        encryption-algorithm 3des-cbc;
    }
    policy main_mode_ike_policy {
        mode main;
        proposals psk_sha1_3des_ike_proposal;
        pre-shared-key ascii-text "$9$KqT3AtORcl0BlMLNY2UjH"; ##
SECRET-DATA
    }
}
traceoptions {
    flag ike;
}
establish-tunnels immediately;
}

```



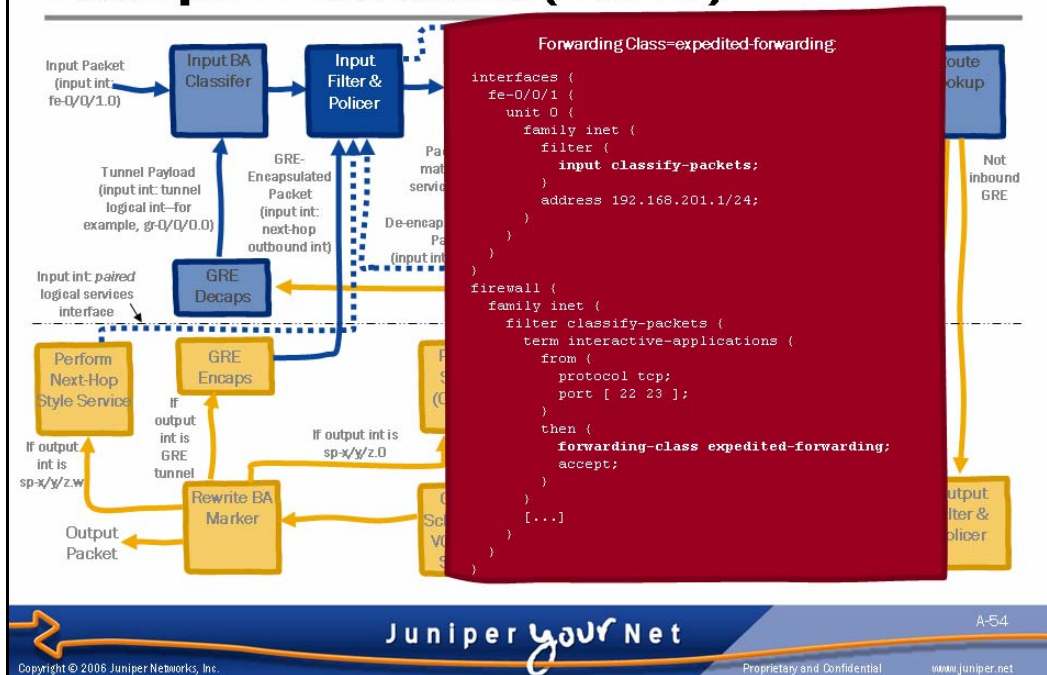
### Original Packet

Like Example 1, this example shows the first two packets in a Telnet session opened from 192.168.201.2 to 192.168.200.2. Because this user is attempting to begin a new TCP session, the SYN bit is set in the packet's TCP header. The packet is destined for TCP port 23 (the Telnet port). The ToS bits in the IP header are set to 0x0 when the packet arrives.

Because the packet arrives on the fe-0/0/1.0 interface, the router sets the input interface to fe-0/0/1.0, and the router processes the packet through the BA classifier configured for this interface. (Be aware of the input interface because this interface changes as the router continues processing the packet.)

Even though no BA classifier is configured for fe-0/0/1.0, the router still uses a default classifier to assign each packet to a forwarding class. In this case, the router assigns this packet to the best-effort forwarding class (see page A-16).

## Life of a Packet: Example 2—Outbound (2 of 17)

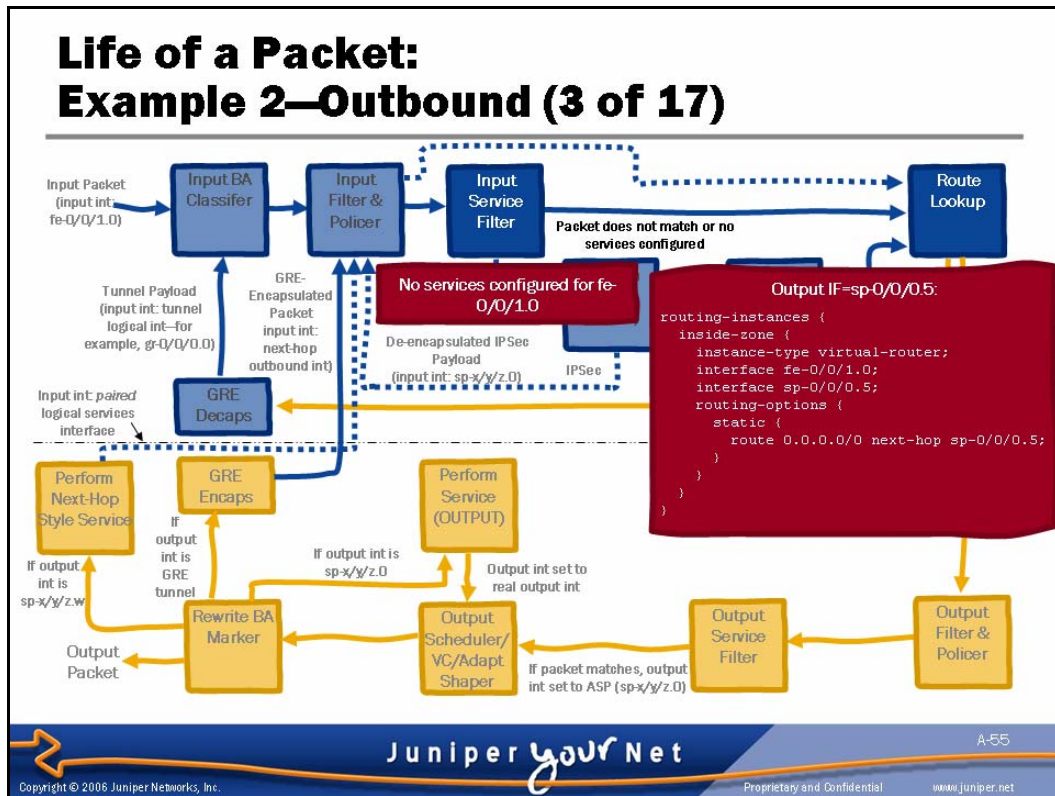


### Input Filter and Policer

The PFE next processes the input policer and input filter. (If the filter references a policer, it is processed at the same time.)

In this case, the `fe-0/0/1.0` configuration references an input filter called `classify-packets`. The packet matches the first term of the filter and is assigned to the expedited-forwarding forwarding class. The router maintains this forwarding class association throughout the remainder of packet processing unless it is changed by a later step in the packet processing.

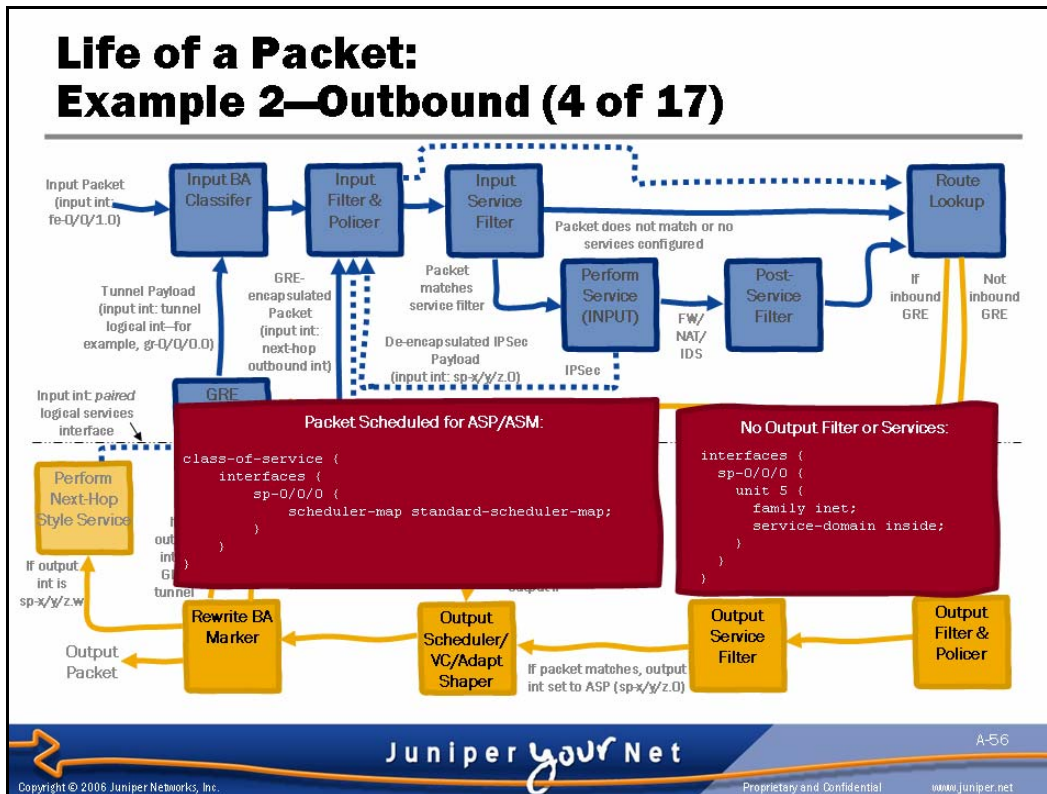




### Input Services and Route Lookup

Because no input service is configured for fe-0/0/1.0, the PFE does not need to check service filters.

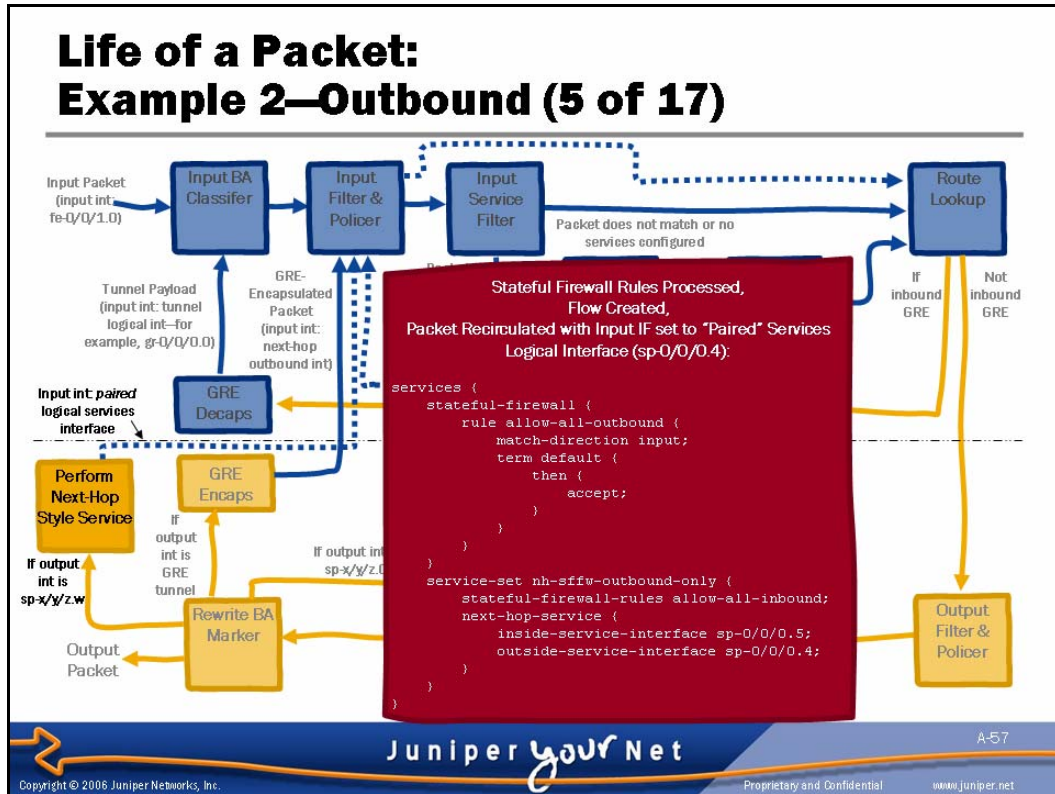
The PFE next performs a route lookup for the destination (192.168.200.2) in the inside-zone.inet forwarding table. This lookup returns the default route to sp-0/0/0.5.



### Output Processing (for `sp-0/0/0.5`)

The PFE next checks for an output policer and output filter for `sp-0/0/0.5`. Because none exists, the packet proceeds to CoS processing.

The router processes the packet through the configuration for the `sp-0/0/0.5` interface under `[edit class-of-service interfaces]`. The router will use `standard-scheduler-map` to schedule the traffic for transmission from the FPC to the PIC (see page A-21).

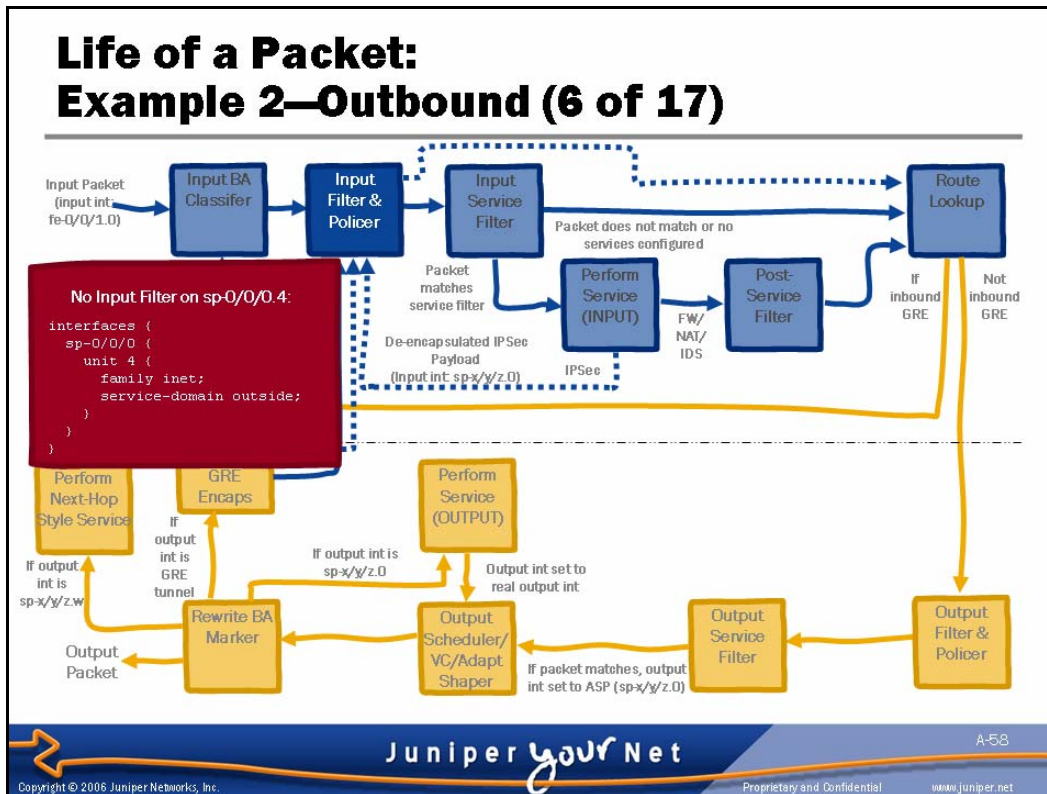


### Perform Next-Hop-Style Service

The router sends the entire packet to the ASP, where the next-hop-style service is processed. In this case, because `sp-0/0/0.5` is the inside-service-interface of the `nh-sffw-outbound-only` service set, that service set is performed.

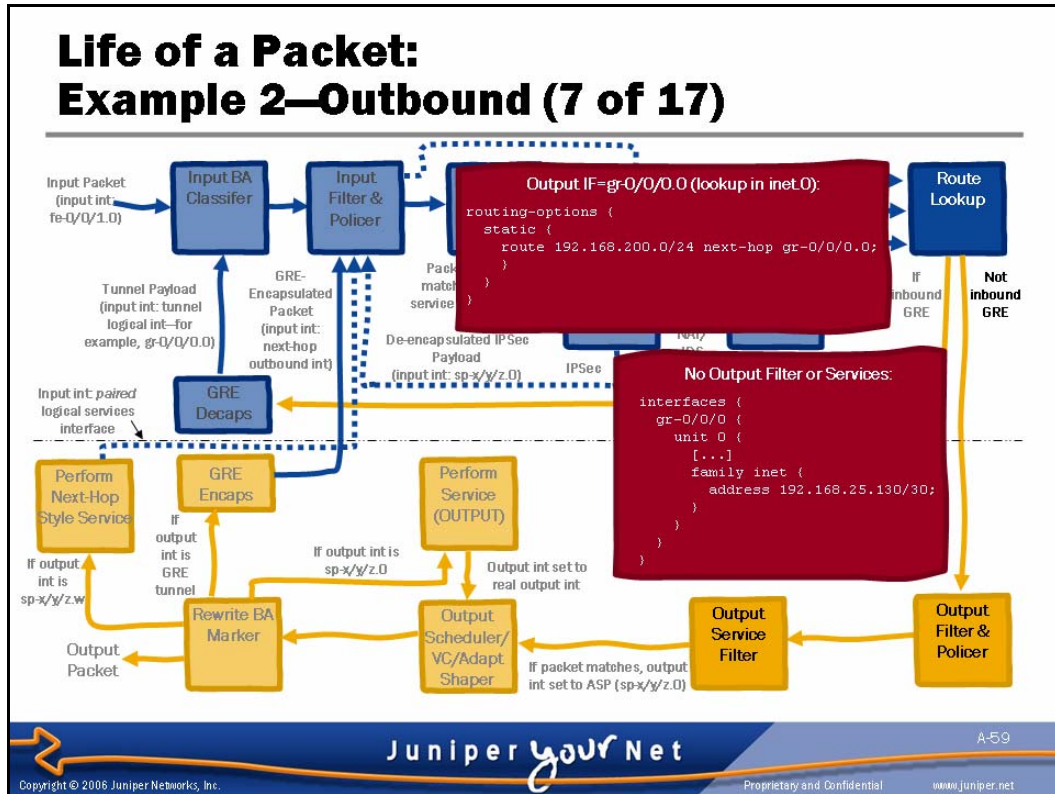
The only rule within that service set is a stateful firewall rule that permits all traffic outbound (from the inside interface to the outside interface). Note, however, that the `match-direction` of this rule is `input`. The match direction you choose has to do with a difference in the way that next-hop-style and interface-style services are performed within the ASP.

When the ASP finishes processing the packet, it transmits the entire packet back to the FPC with the input interface set to the *paired* logical service interface. In this case, because the ASP received the packet on the inside-service-interface (`sp-0/0/0.5`), it sends the packet to the FPC with the input interface set to the outside-service-interface (`sp-0/0/0.4`).



### Input Filter and Policer (for sp-0/0/0.4)

The PFE next processes the packet through the input policer and input filter for sp-0/0/0.4. However, because none is configured, the PFE proceeds directly to a route lookup. (The dotted line that skips the input service filter represents it not being possible to configure services on service interfaces.)



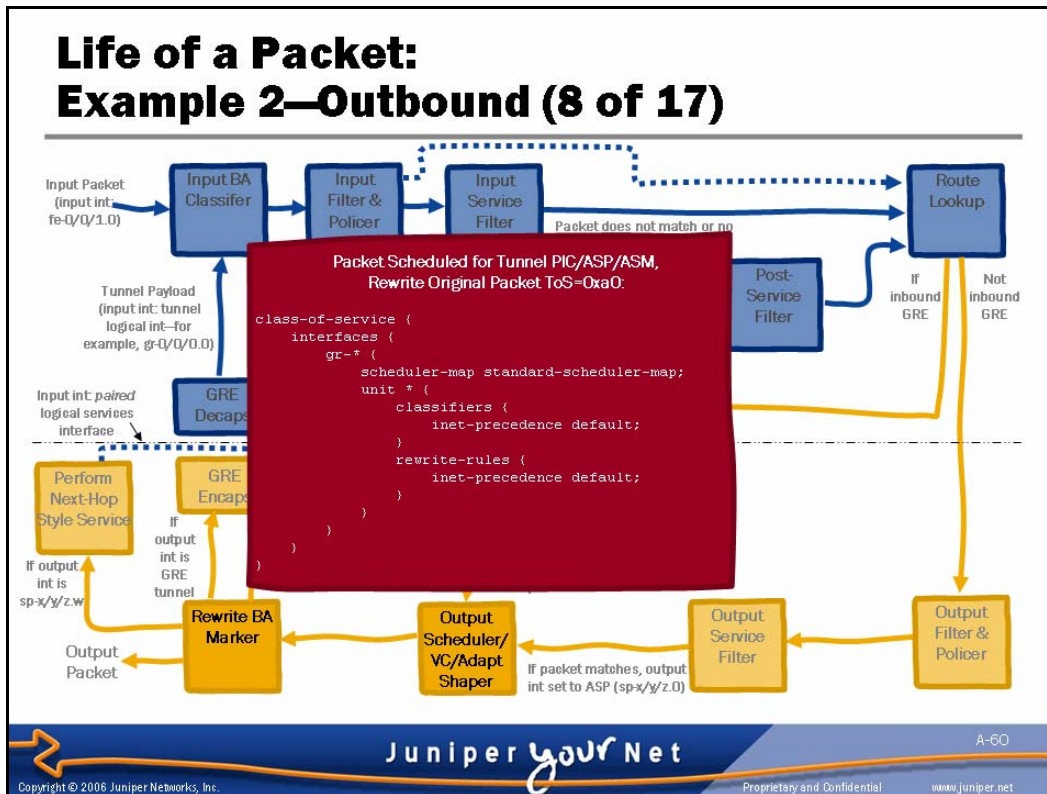
### Route Lookup and Output Filters (for gr-0/0/0.0)

The PFE next looks up the destination (still 192.168.200.2, because no NAT rules existed to change the destination) in the `inet` forwarding table and finds a static route for 192.168.200.0/24 with a next hop of `gr-0/0/0.0`. Therefore, it sets the output interface to `gr-0/0/0.0`.

It performs this lookup in the `inet` forwarding table because the input interface (now `sp-0/0/0.4`) is in the master routing instance. To perform the stateful firewall service using a next-hop-style service set and avoid a routing loop, we placed the inside interface in a different routing instance. This placement allowed us to have a route to the `inside-service-interface` in that routing instance, while we placed the actual route to the destination in the master routing instance. If we had placed both the inside and outside interfaces in the same routing instance and simply configured a static route for the destination with a next-hop of the `inside-service-interface`, we would have had a routing loop within the router itself (between the ASP and the PFE), also called a *services loop*. To avoid this issue, we use different forwarding tables for the lookup that sends the traffic to the `inside-service-interface` and the lookup that occurs when the traffic leaves the `outside-service-interface`.

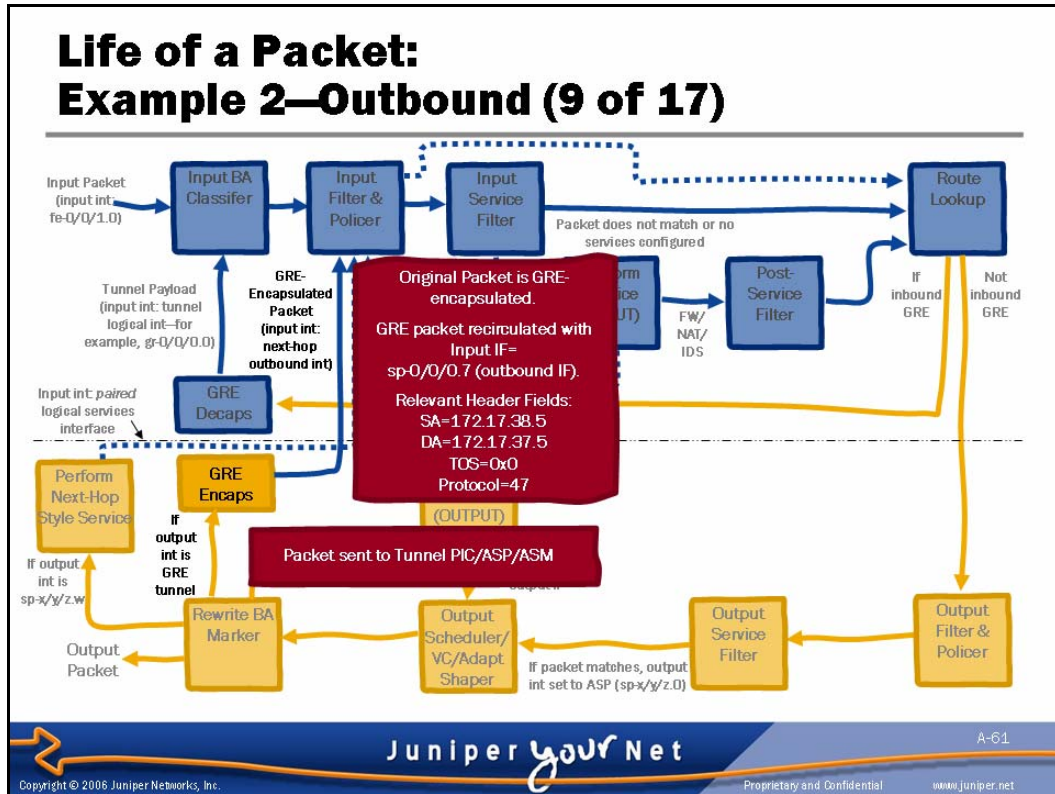
After the route lookup, the PFE checks the output policer, output filter, and output service filters for `gr-0/0/0.0`. Because none is configured, the packet proceeds to CoS processing.





### CoS Processing (for `gr-0/0/0.0`)

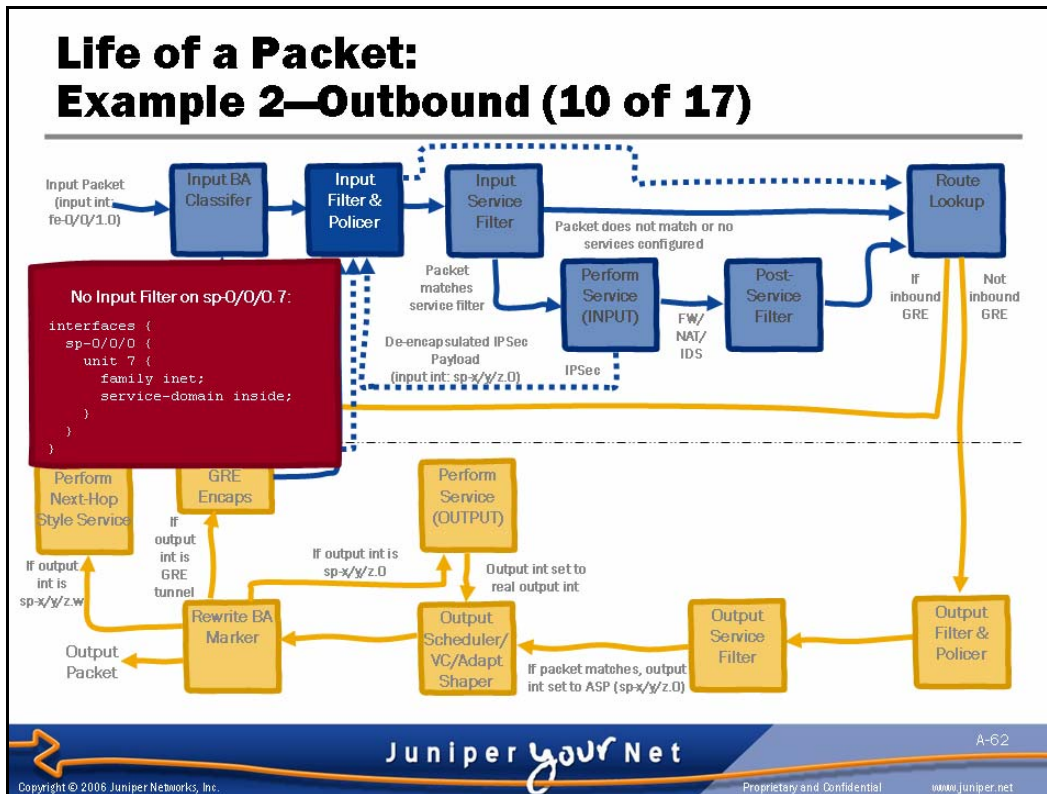
The packet is scheduled for transmission to the ASP according to the configured scheduler. Because the packet is still in the expedited-forwarding forwarding class (it has been maintained since being set on page A-54), the ToS bits in the IP header are rewritten to 0xa0 (the appropriate value for the expedited-forwarding forwarding class and the configured rewrite rule).



## GRE Encapsulation

After CoS processing for `gr-0/0/0.0` completes, the router sends the entire packet to the Tunnel PIC or ASP for encapsulation. The PIC performs the GRE encapsulation. It does maintain the forwarding class of the packet, but it does *not* copy the ToS bits from the IP header of the original packet to the IP header of the GRE packet. Therefore, the ToS field of the IP header of the GRE packet is set to 0x0.

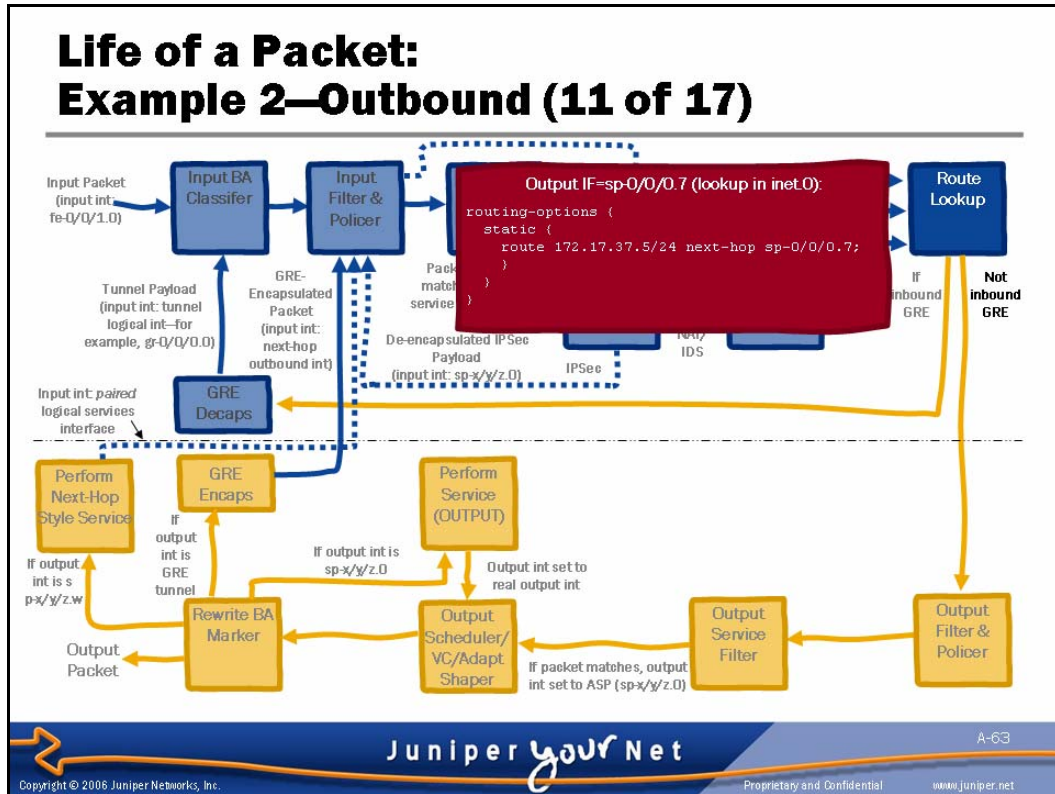
Once the GRE packet is ready, it is sent to the PFE for processing with the *input interface set to the next-hop outbound interface for the destination of the tunnel*. In this case, because the next hop to 172.17.37.5 (the destination of the tunnel) is `sp-0/0/0.7`, the GRE packet's input interface will be set to `sp-0/0/0.7`.



### Input Filter and Policer (for sp-0/0/0.7)

The PFE next processes the packet through the input policer and input filter for sp-0/0/0.7. However, because none is configured, the PFE proceeds directly to a route lookup. (The dotted line that skips the input service filter represents it not being possible to configure services on service interfaces.)

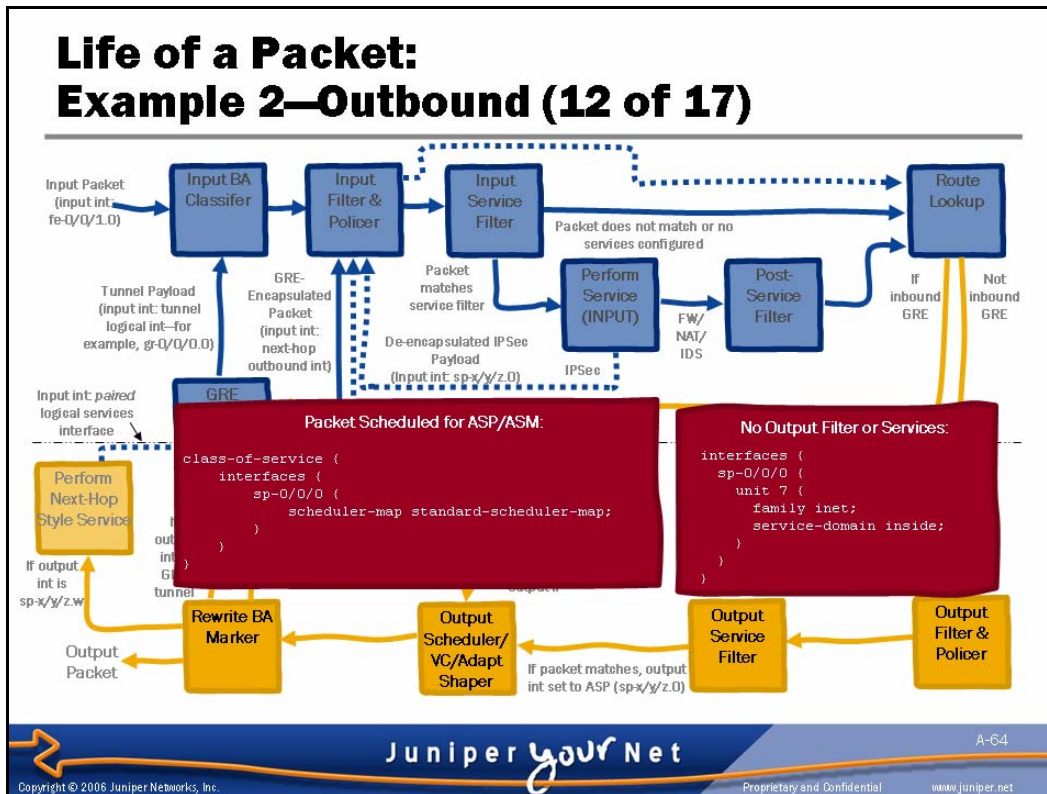




## Route Lookup

Because `sp-0/0/0.7` is in the master routing instance, the router looks up the destination (172.17.37.5) in the `inet` forwarding table and finds the static route for 172.17.37.5/32 with a next-hop of `sp-0/0/0.7`.

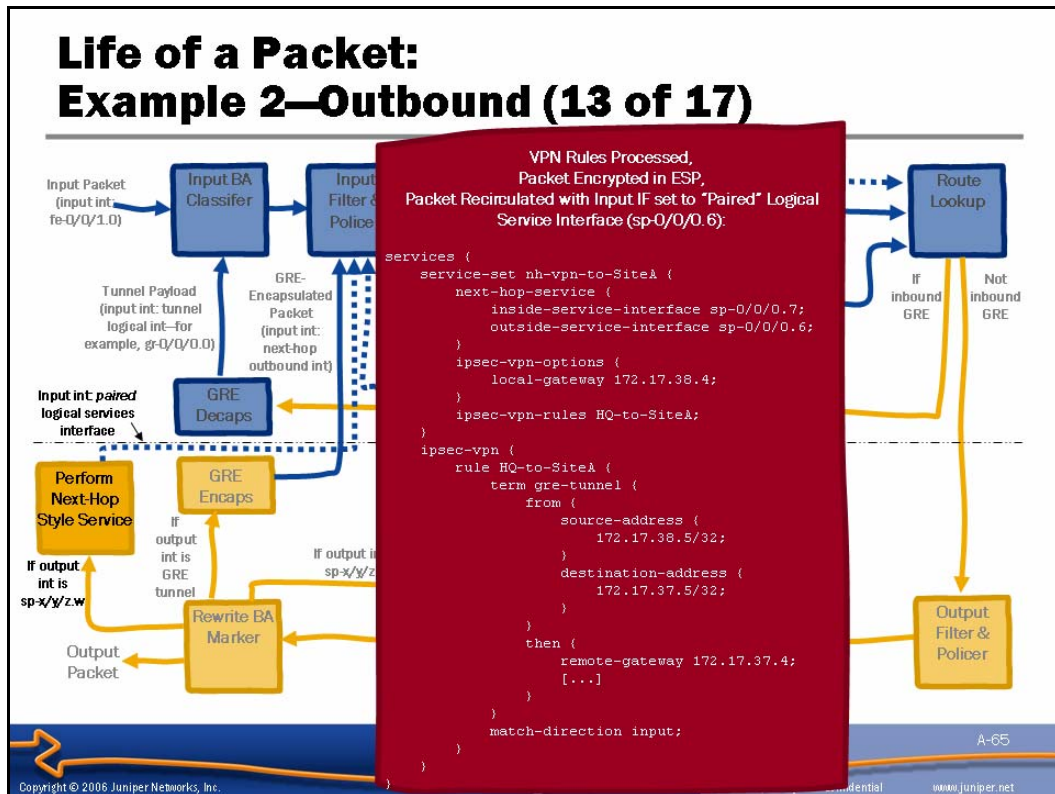
We used different endpoint addresses for the GRE tunnel and VPN security association because the route lookups for both are done in the `inet` forwarding table. Therefore, if the same address were used for both, the route lookup would return the same next hop for both. The router using the same next hop for both pre-encrypted and post-encrypted packets results in either all the GRE traffic leaving the router unencrypted or in a routing loop between the PFE and ASP (see page A-59). To send the GRE traffic to the ASP for encryption and send the VPN traffic to the destination, we chose to use different endpoint addresses.



### Output Processing (for `sp-0/0/0.7`)

The PFE next checks for an output policer and output filter for `sp-0/0/0.7`. Because there is none, the packet proceeds to CoS processing.

The router processes the packet through the configuration for the `sp-0/0/0.7` interface under `[edit class-of-service interfaces]`. The router will use `standard-scheduler-map` to schedule the traffic for transmission from the FPC to the PIC.

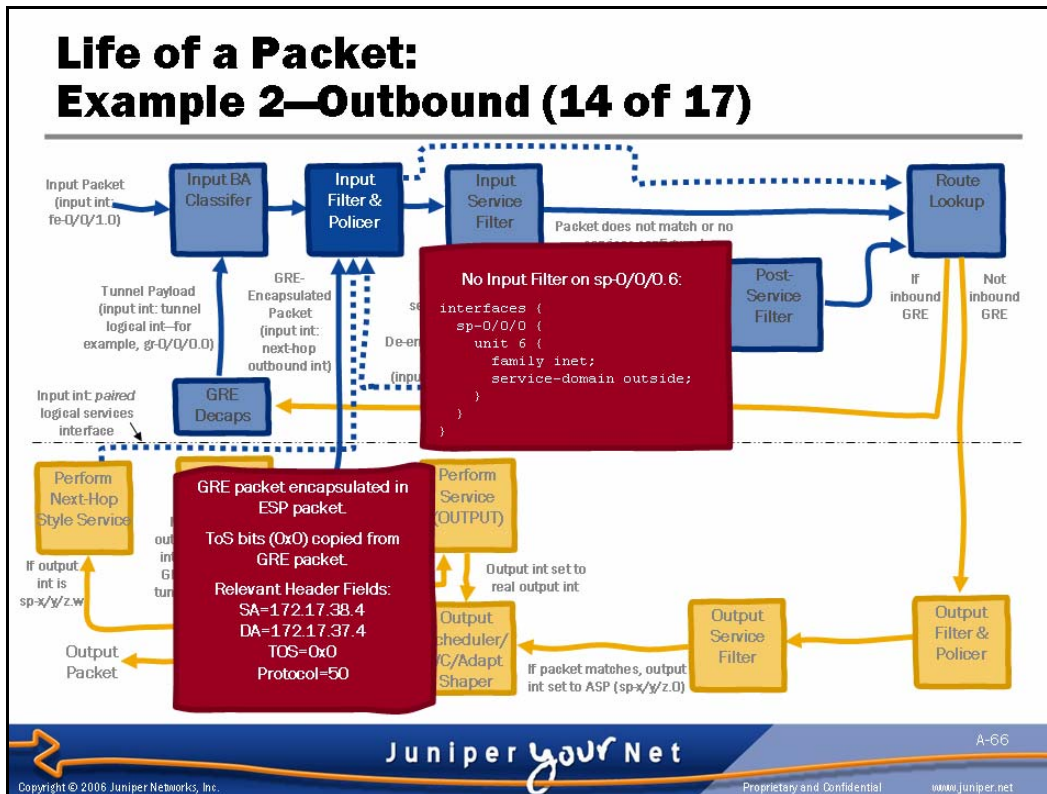


### Perform Next-Hop-Style Service

The entire packet is next sent to the ASP, where the next-hop-style service is processed. In this case, because `sp-0/0/0.7` is the `inside-service-interface` of the `nh-vpn-to-SiteA` service set, that service set is performed.

The only rule within that service set is a VPN rule that matches traffic from `172.17.38.5/32` to `172.17.37.5/32` (the endpoints of the tunnel). Note that, although the match is meant to occur on outbound traffic (traffic from the inside interface to the outside interface), the `match-direction` of this rule is `input`. This match direction has to do with a difference in the way that next-hop-style and interface-style services are performed within the ASP.

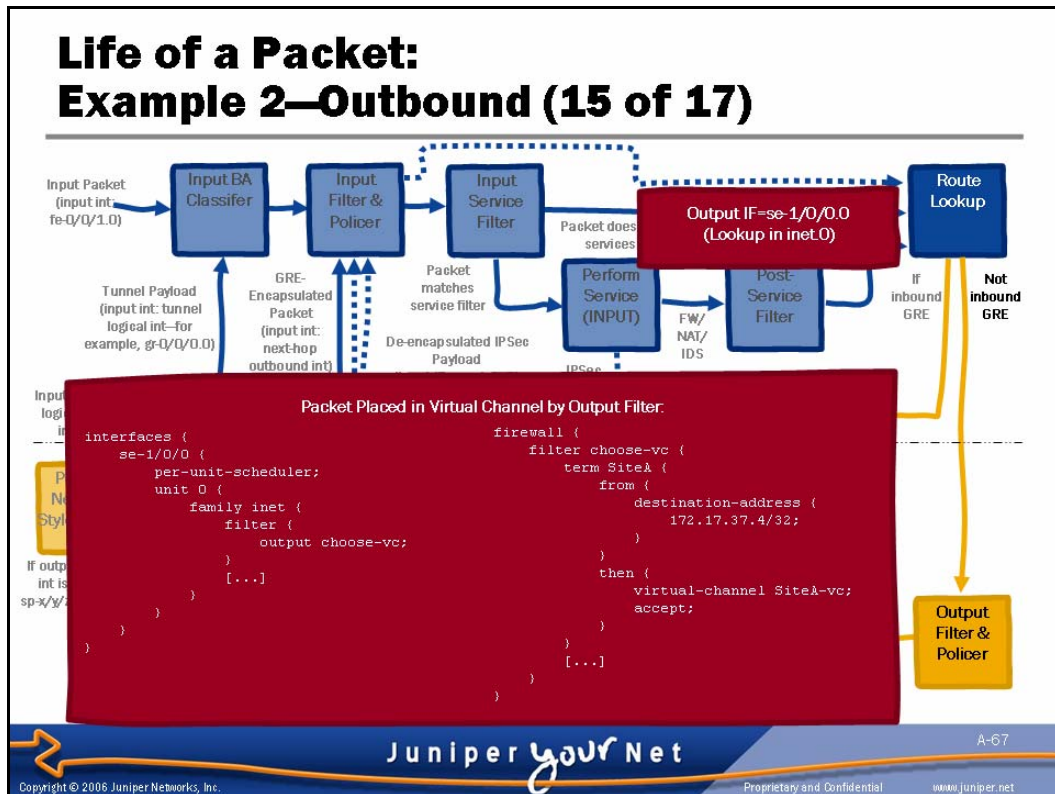
When the ASP is finished processing the packet, it sends the entire packet back to the FPC with the input interface set to the *paired* logical service interface. In this case, because the ASP received the packet on the `inside-service-interface` (`sp-0/0/0.7`), it sends the packet to the FPC with the input interface set to the `outside-service-interface` (`sp-0/0/0.6`).



### Input Filter and Policer (for sp-0/0/0.6)

The ASP encrypts the GRE packet and encapsulates it within an ESP packet. The ASP copies the ToS bits (0x0) from the IP header of the GRE packet to the IP header of the ESP packet. The IP header of the ESP packet has a source address of the configured local-gateway and a destination address of the configured remote-gateway.

When the PFE receives the packet from the ASP, it processes the packet through the input policer and input filter for sp-0/0/0.6. However, because none is configured, the PFE proceeds directly to a route lookup. (The dotted line that skips the input service filter represents it not being possible to configure services on service interfaces.)



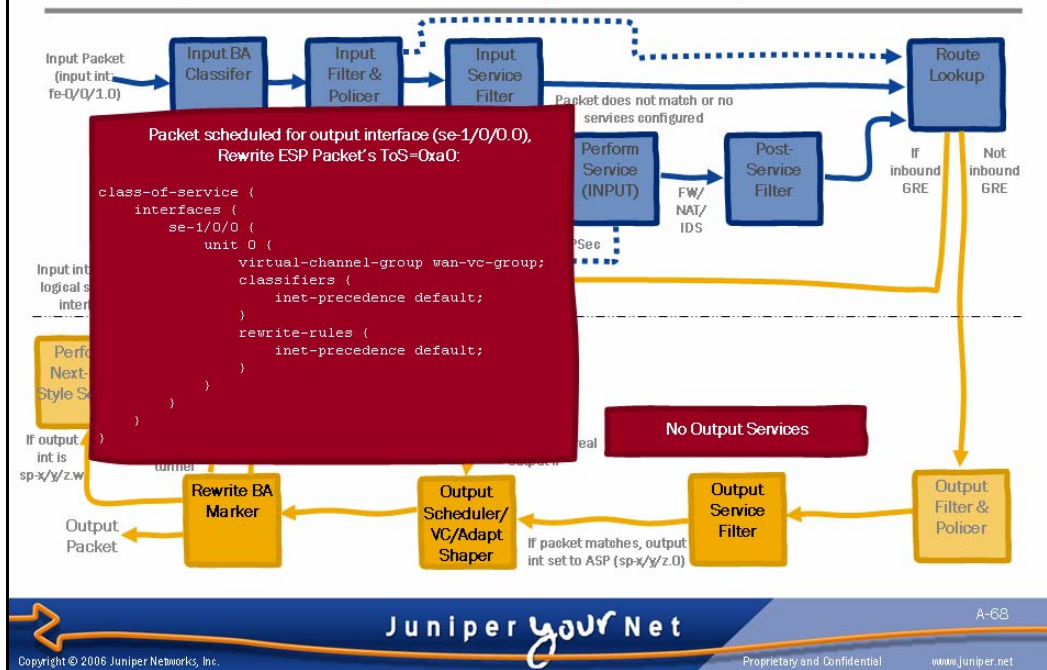
### Route Lookup and Output Filter Processing

Because `se-0/0/0.6` is in the master routing instance, the router looks up the destination (`172.17.37.4`) in the `inet` forwarding table and finds the default route with a next hop of `se-1/0/0.0`.

The PFE then processes the output filter for `se-1/0/0.0`. The output filter places traffic destined to the remote site in a virtual channel called `SiteA-vc`. Because the ESP packets (rather than the original packets or GRE-encapsulated packets) are processed through this filter, this filter must only match on the ESP tunnel endpoint to match all traffic destined for SiteA via the VPN.



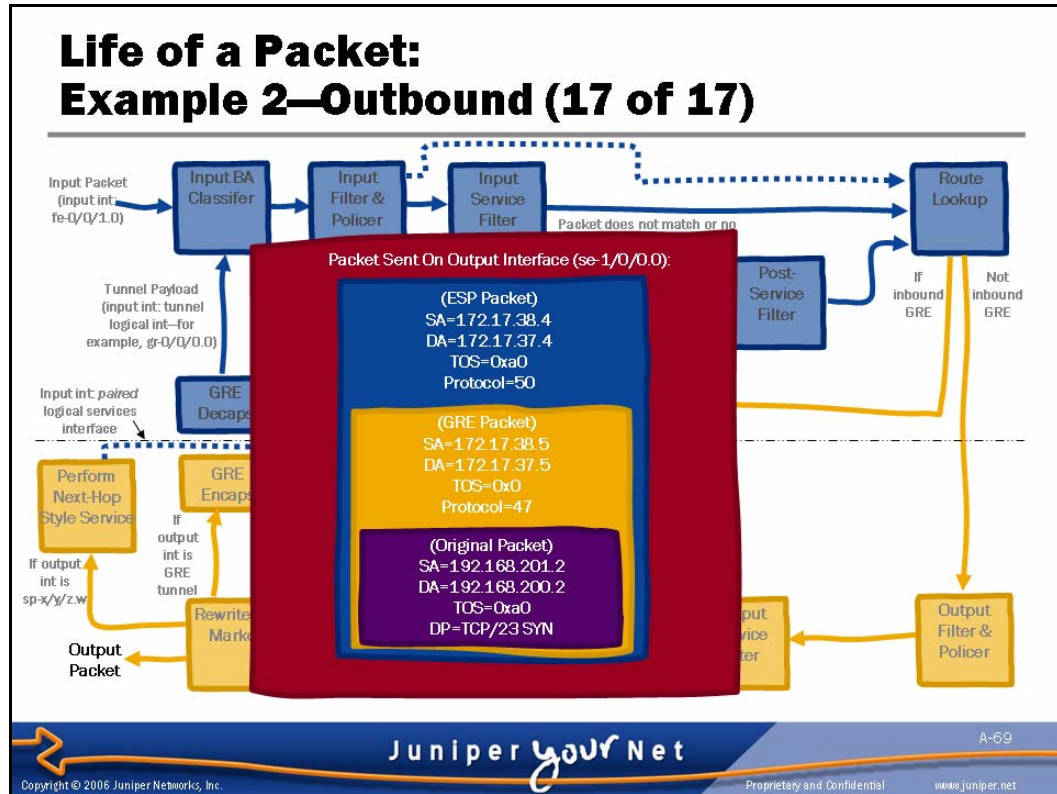
## Life of a Packet: Example 2—Outbound (16 of 17)



### Output Services and CoS Processing (for se-1/0/0.0)

After processing the output filter, the PFE processes service filters; however, because no output service is configured on `se-1/0/0.0`, the packet proceeds to CoS processing.

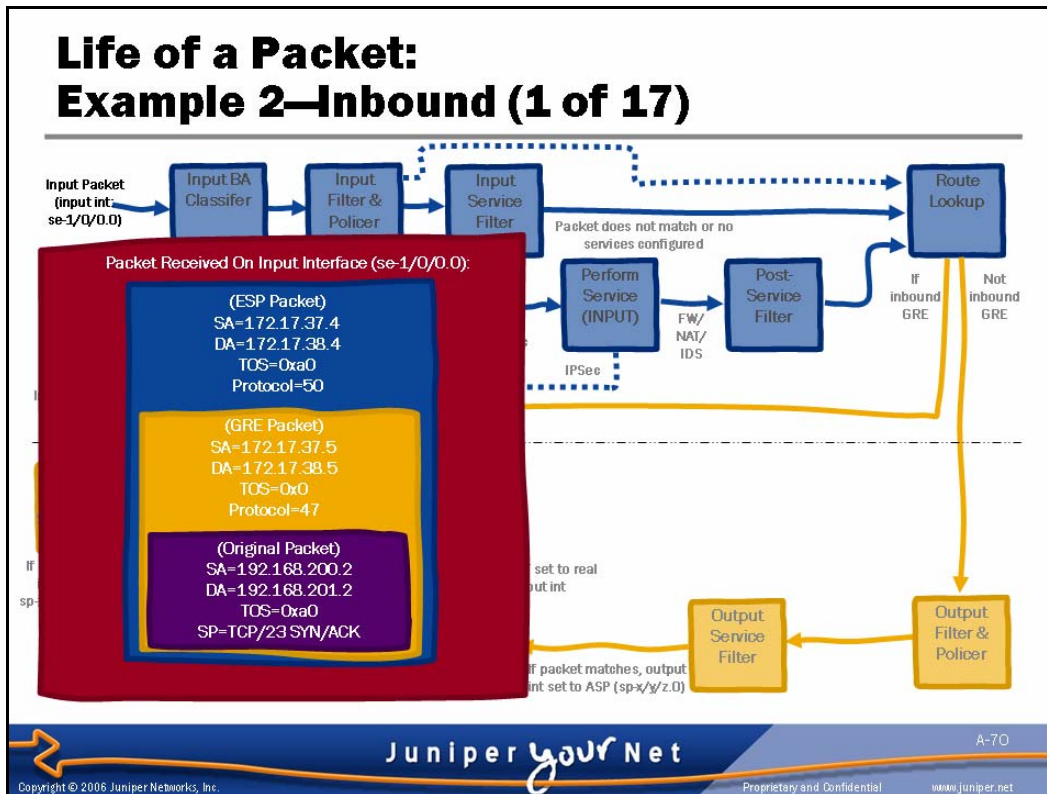
The packet is scheduled according to the configured virtual-channel group configuration. Because the packet is still in the expedited-forwarding forwarding class (it has been maintained since being set on page A-54), the ToS bits in the IP header are rewritten to 0x0 (the appropriate value for the expedited-forwarding forwarding class and the configured rewrite rule).



### Packet Transmitted

After this entire process is completed, the PIC finally transmits the packet onto the wire.

The graphic on the slide represents the transmitted packet. The original packet (changed only by having its ToS bits set to 0x0) is encapsulated within a GRE packet. The GRE packet is encrypted and encapsulated within an ESP packet. The ESP packet is sent on the wire with its ToS bits set to 0x0 by a rewrite rule.

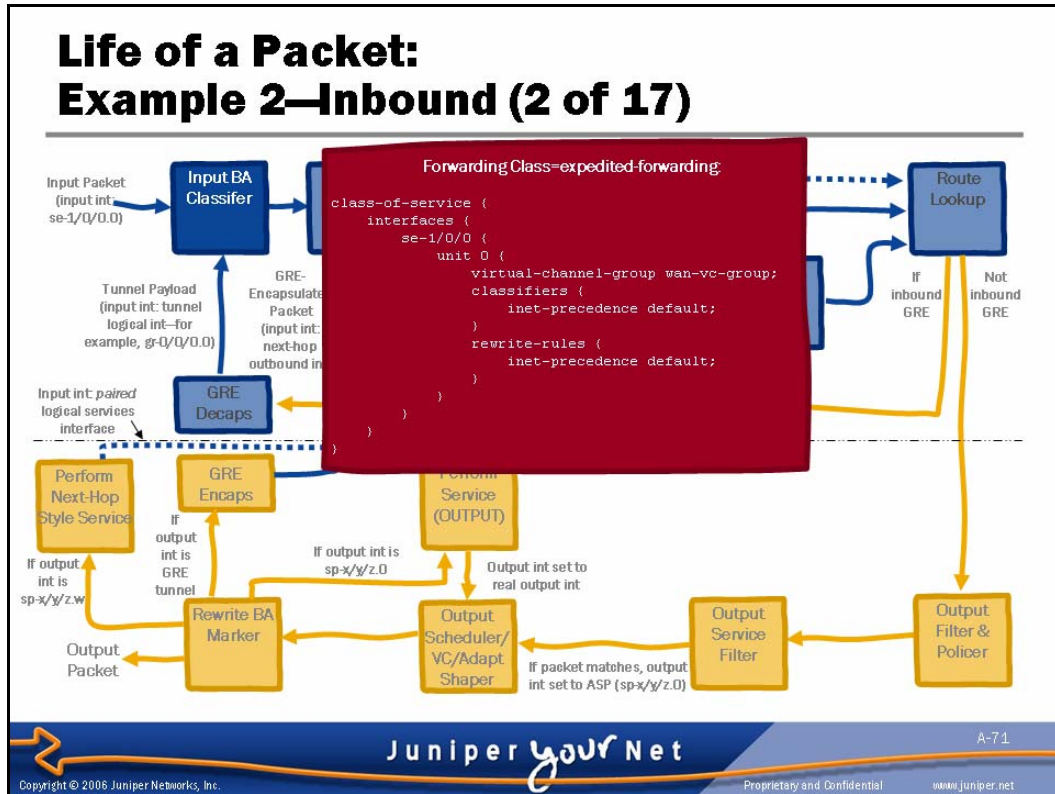


### Packet Received

Once the remote Telnet server (192.168.200.2) receives the initial TCP packet (with the SYN bit set) from 192.168.201.2, it responds with a TCP packet with the SYN and ACK flags set. When that packet is received by HQ-1, it is encapsulated within a GRE packet, which is itself encrypted and encapsulated within an ESP packet (as shown on the slide).

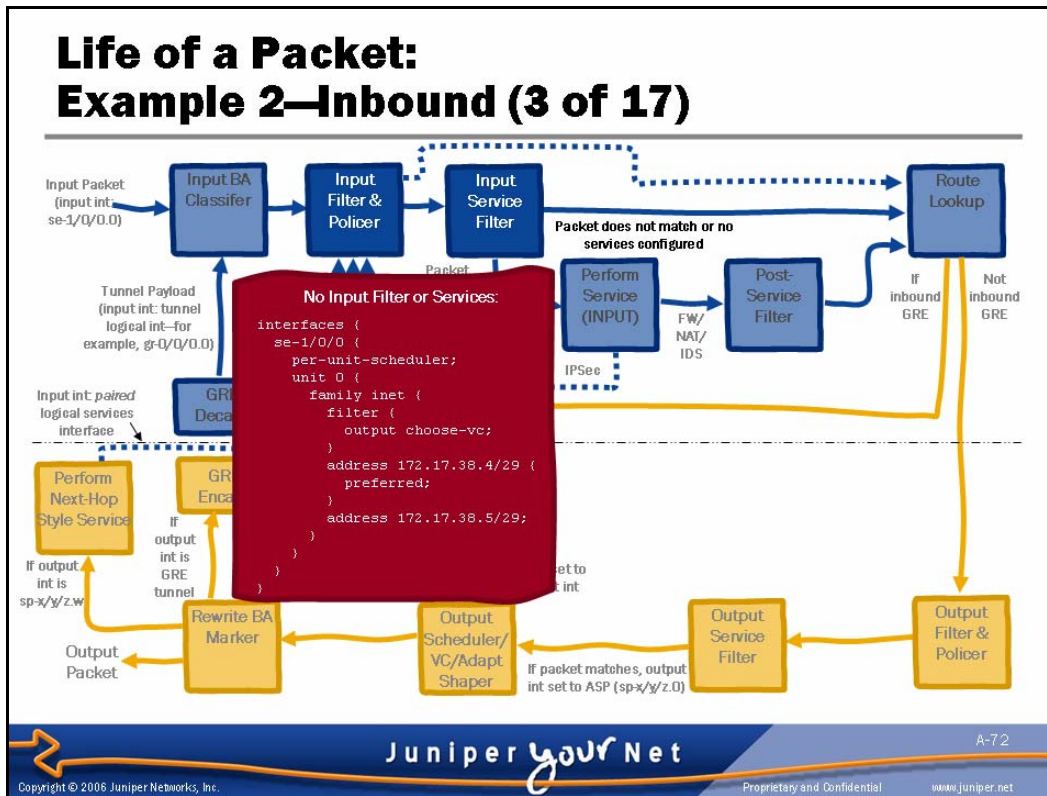
We now examine how the router processes this packet. We must examine both directions because some differences exist between the inbound and outbound packet processing.





### BA Classifier (se-1/0/0.0)

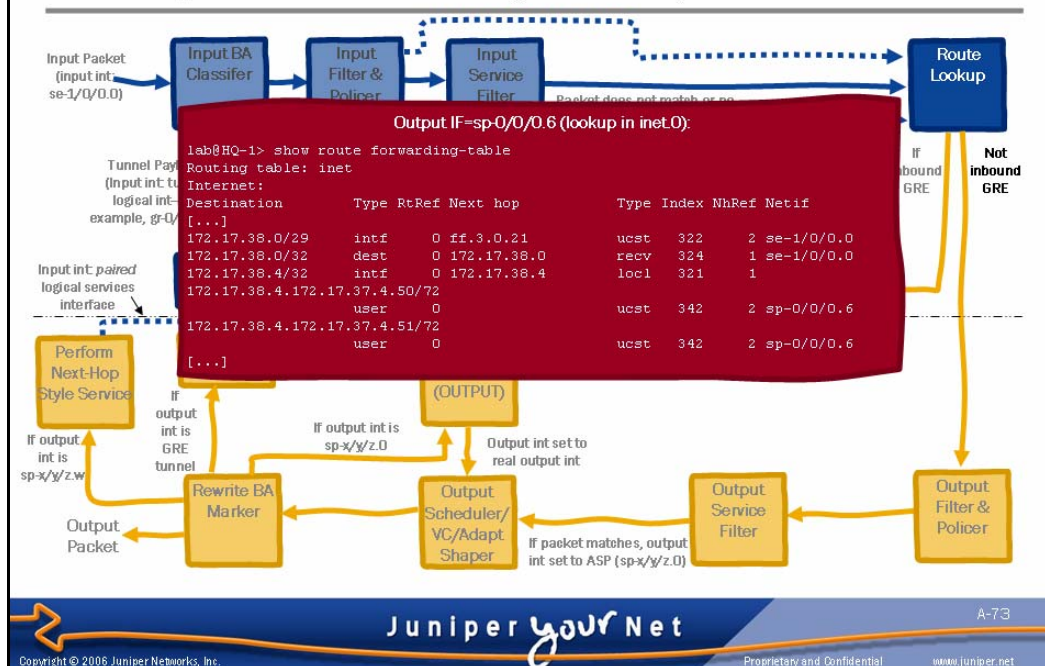
The router begins by processing the BA classifier for se-1/0/0.0 (the input interface). The classifier causes this packet to be assigned the expedited-forwarding class.



### Input Filter and Service Filters

No input policer, input filter, or input service is configured on the `se-1/0/0.0` interface; therefore, the PFE can proceed directly to a route lookup.

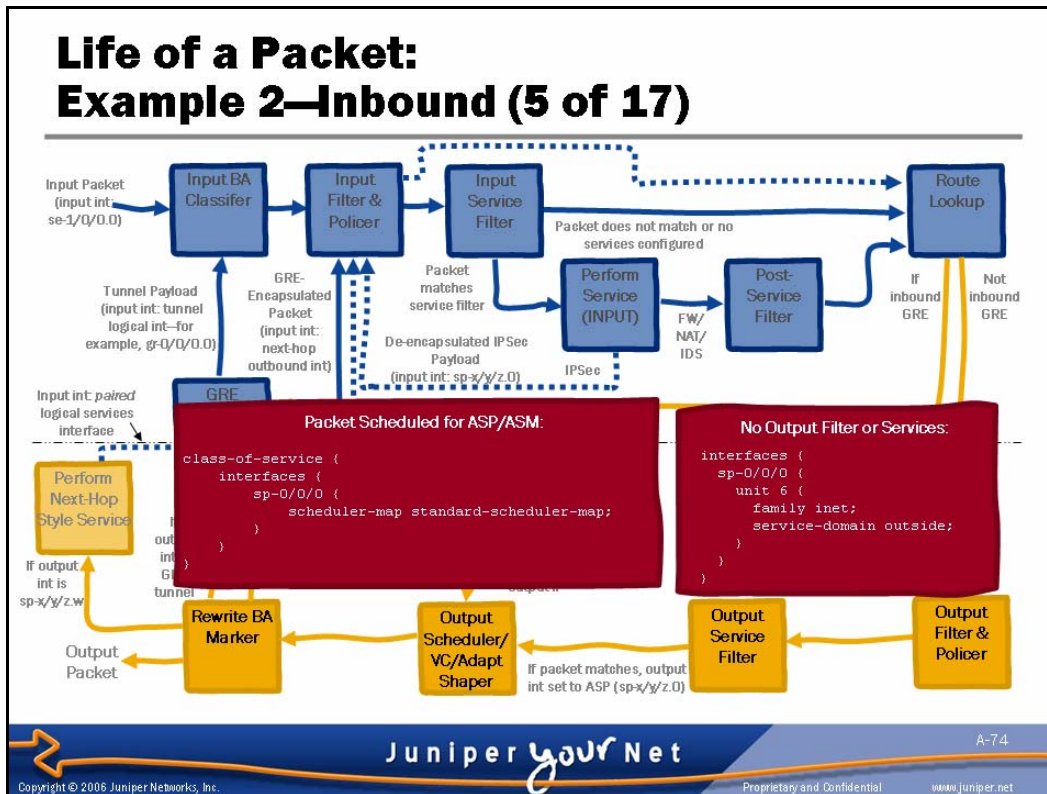
## Life of a Packet: Example 2—Inbound (4 of 17)



### Route Lookup

When the PFE performs the route lookup, it finds that the input packet matches one of two /72 entries installed in the forwarding table for this security association. The router installs entries in the forwarding table for the source address of the remote-peer, the destination address of the local-peer, and IP protocols 50 and 51 (ESP and AH, respectively) with a next hop of the appropriate service interface. This forwarding table entry causes the PFE to set the output interface to be the outside-service-interface of the appropriate VPN service-set (in this case, the next hop is sp-0/0/0.6):

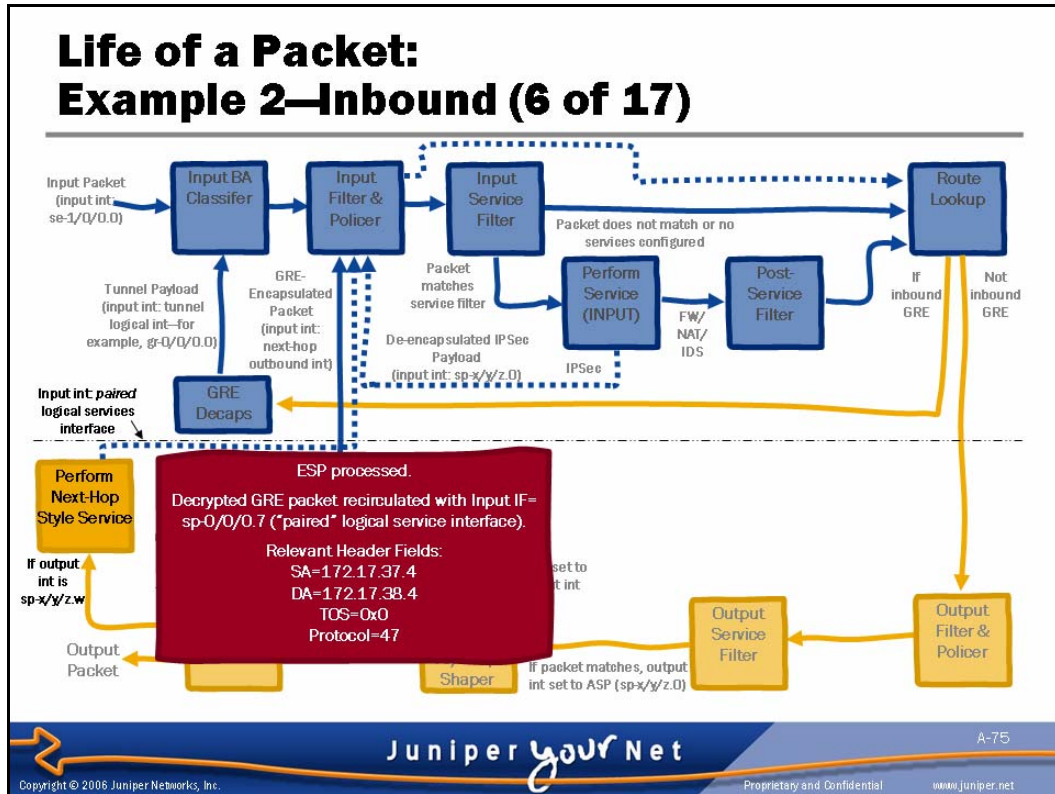
```
lab@HQ-1> show route forwarding-table
Routing table: inet
Internet:
Destination      Type RtRef Next hop      Type Index NhRef Netif
[...]
172.17.38.0/29    intf  0 ff.3.0.21    ucst  322  2 se-1/0/0.0
172.17.38.0/32    dest  0 172.17.38.0  recv  324  1 se-1/0/0.0
172.17.38.4/32    intf  0 172.17.38.4  locl  321  1
172.17.38.4.172.17.37.4.50/72
user             0                                ucst  342  2 sp-0/0/0.6
172.17.38.4.172.17.37.4.51/72
user             0                                ucst  342  2 sp-0/0/0.6
[...]
```



### Output Processing (for sp-0/0/0.6)

The PFE next checks for an output policer and output filter for sp-0/0/0.6. Because none exists, the packet proceeds to CoS processing.

The router processes the packet through the configuration for the sp-0/0/0.6 interface under [edit class-of-service interfaces]. The router uses standard-scheduler-map to schedule the traffic for transmission from the FPC to the PIC.



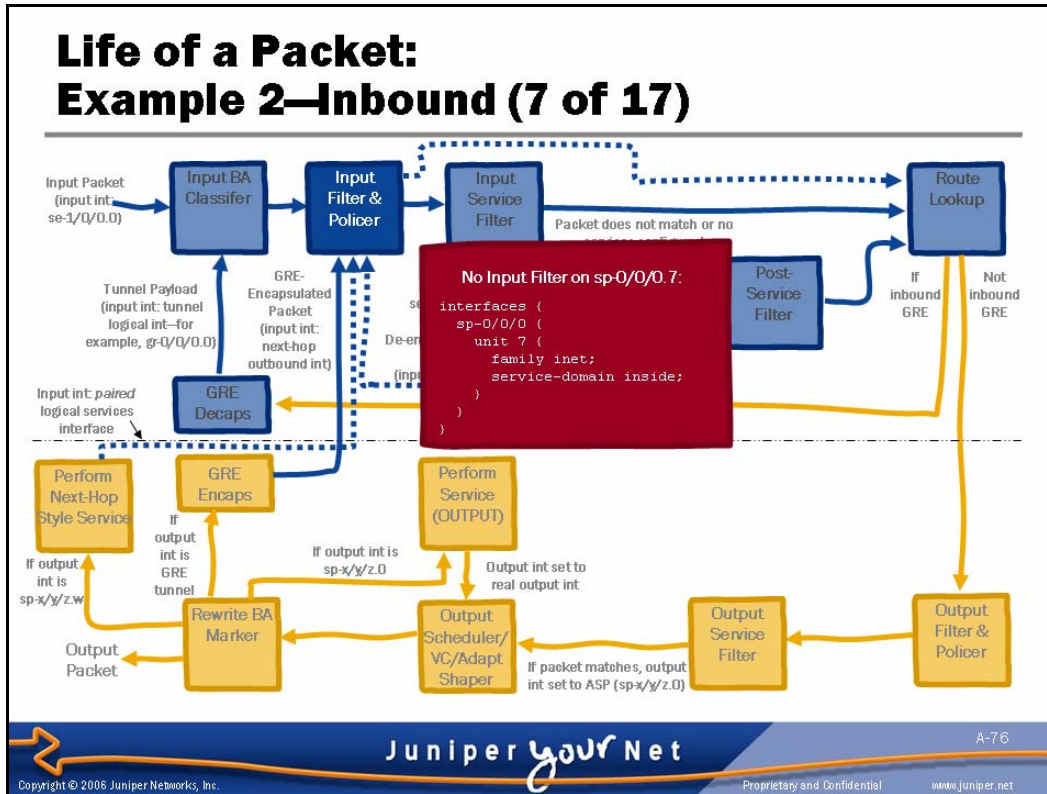
### Perform Next-Hop-Style Service

The entire packet is next sent to the ASP, where the next-hop-style service is processed. In this case, because `sp-0/0/0.6` is the outside-service-interface of the `nh-vpn-to-SiteA` service set, that service set is performed.

The ASP processes the ESP packet and decrypts the encapsulated GRE packet. The ToS bits from the IP header of the ESP packet are *not* copied to the IP header of the decrypted GRE packet. However, the ASP does maintain the packet's forwarding class association.

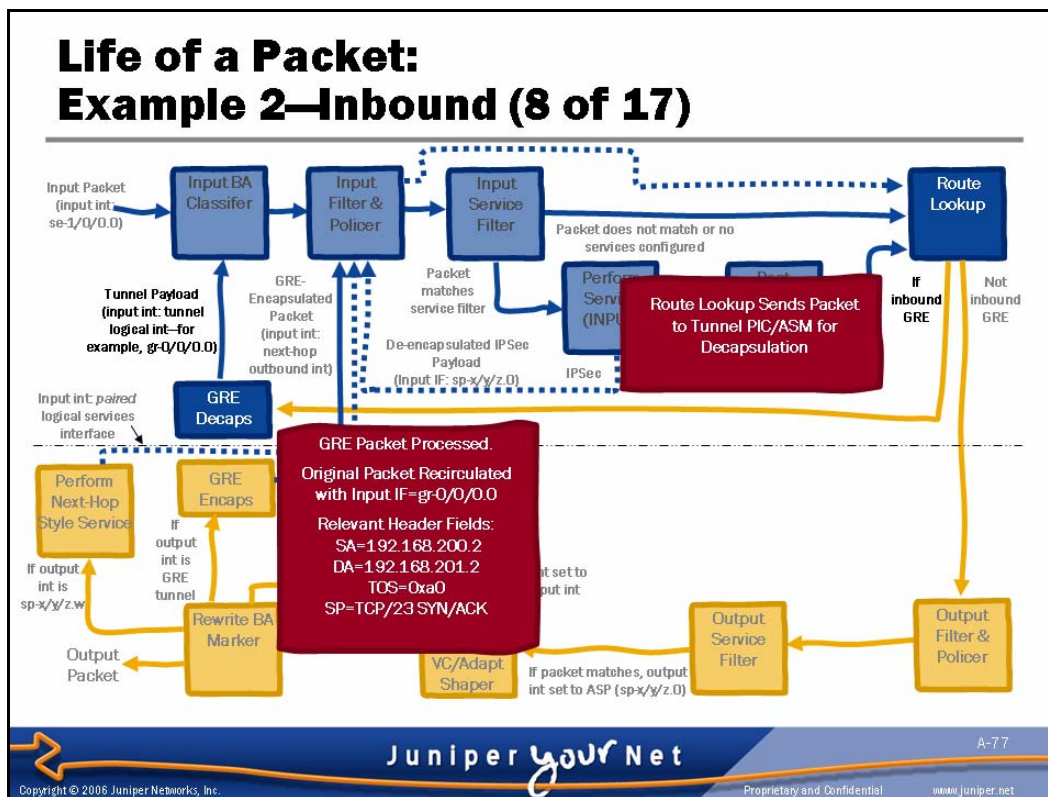
When the ASP finishes processing the packet, it sends the entire packet back to the FPC with the input interface set to the *paired* logical service interface. In this case, because the ASP received the packet on the outside-service-interface (`sp-0/0/0.6`), it will send the packet to the FPC with the input interface set to the inside-service-interface (`sp-0/0/0.7`).





### Input Filter and Policer (for sp-0/0/0.6)

When the PFE receives the packet from the ASP, it processes the packet through the input policer and input filter for sp-0/0/0.6. However, because none is configured, the PFE proceeds directly to a route lookup. (The dotted line that skips the input service filter represents it not being possible to configure services on service interfaces.)

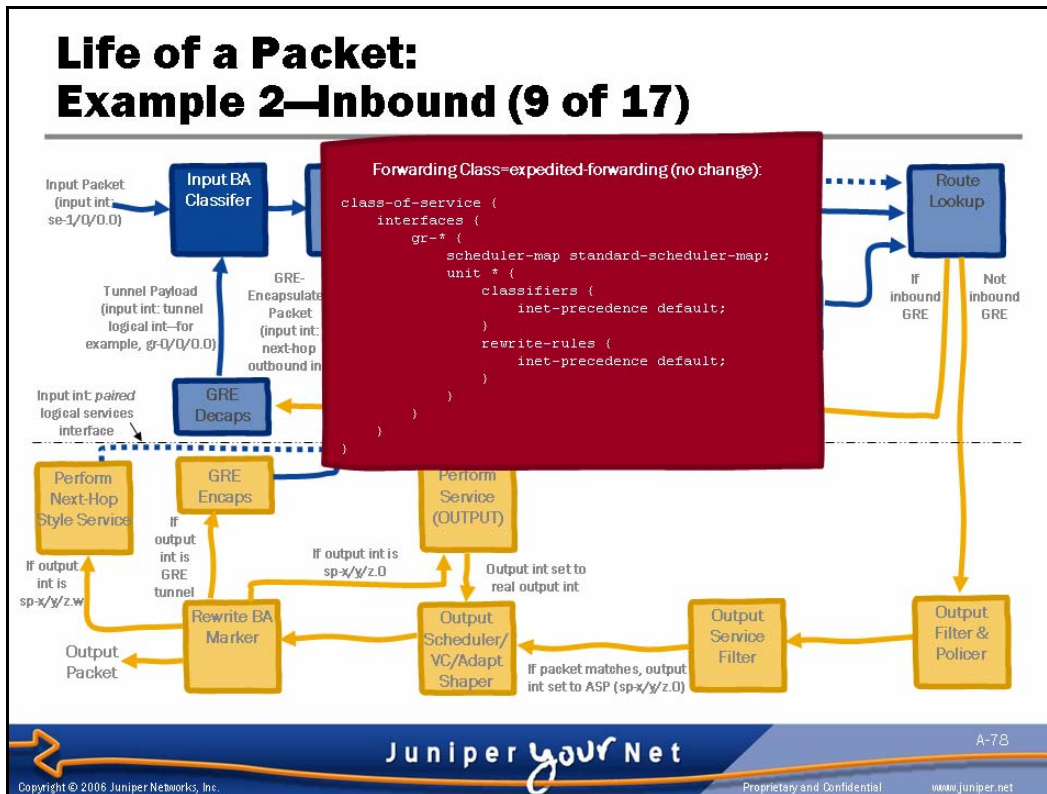


## GRE Decapsulation

The PFE performs a route lookup in the forwarding table and finds that the packet matches a /72 entry for a source address of the remote tunnel endpoint, a destination address of the local tunnel endpoint, and a protocol number of 47 (the protocol number used for GRE). This forwarding table entry causes the entire packet to be sent to the Tunnel PIC or ASP for processing:

```
lab@Montreal# run show route forwarding-table
Routing table: inet
Internet:
Destination          Type RtRef Next hop          Type Index NhRef Netif
[...]
172.17.38.4/32        intf      0 172.17.38.4          locl   321    1
172.17.38.4.172.17.37.4.50/72
                        user       0                      ucst   331    2 sp-1/2/0.6
172.17.38.4.172.17.37.4.51/72
                        user       0                      ucst   331    2 sp-1/2/0.6
172.17.38.5.172.17.37.5.47/72
                        dest       0                      locl   332    1
172.17.38.7/32        dest      0 172.17.38.7          bcst   323    1 so-0/1/2.0
[...]
```

When the packet is decapsulated, it is sent to the FPC with a source interface of the tunnel's logical interface (gr-0/0/0.0). The IP header of the encapsulated packet is maintained (including the original ToS bits), and the forwarding class is maintained.

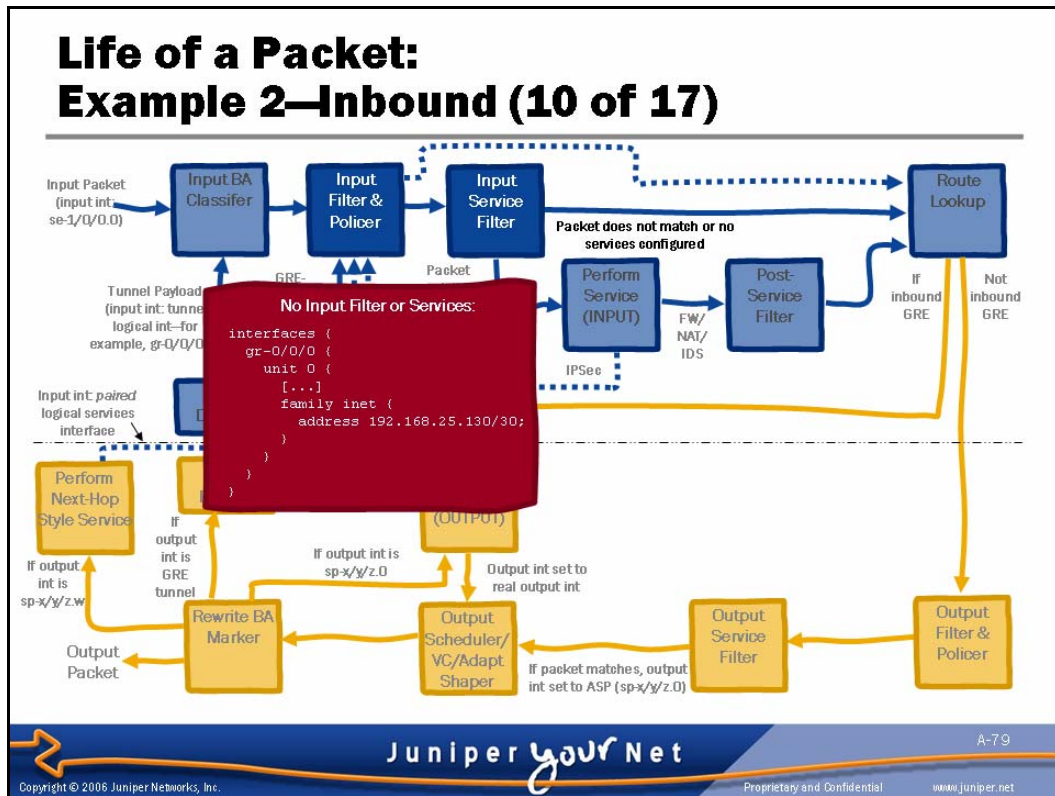


### BA Classifier (for gr-0/0/0.0)

At this point, the router processes the de-encapsulated packet through the BA classifier configured for `gr-0/0/0.0`. Because the IP header of the encapsulated packet had the ToS field set to 0x0, the `inet-precedence` default classifier sets the forwarding class to expedited-forwarding, which is the same as the previous forwarding class of the packet.

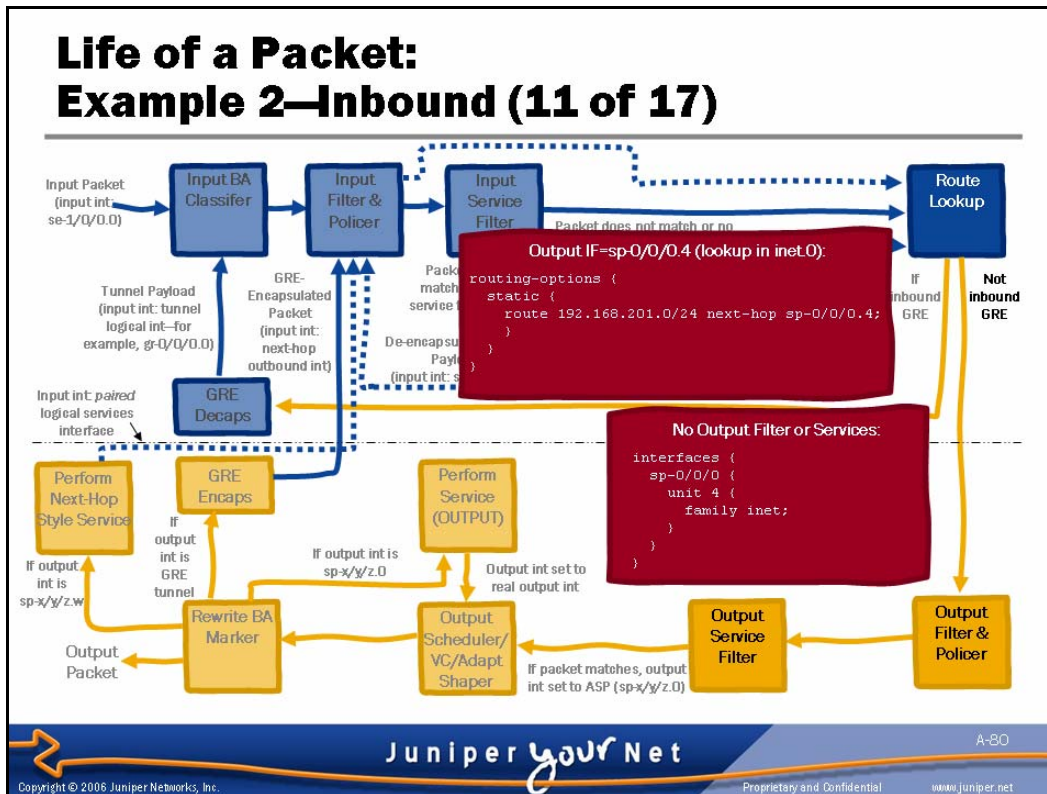
However, as was explained on page A-16, if no classifier were configured on `gr-0/0/0.0`, a default classifier would still have been used by the router. This default classifier would have set this packet in the best-effort forwarding class because that classifier maps a ToS field of 0x0 (IP precedence bits 101) to the best-effort forwarding class. Also, if the de-encapsulated packet had had a different value in the ToS field of its IP header (or the classifier configured on `gr-0/0/0.0` had returned a different forwarding class for any reason), the router would have changed the forwarding class of the packet to the value returned by the BA classifier.





### Inbound Filter and Services (for gr-0/0/0.0)

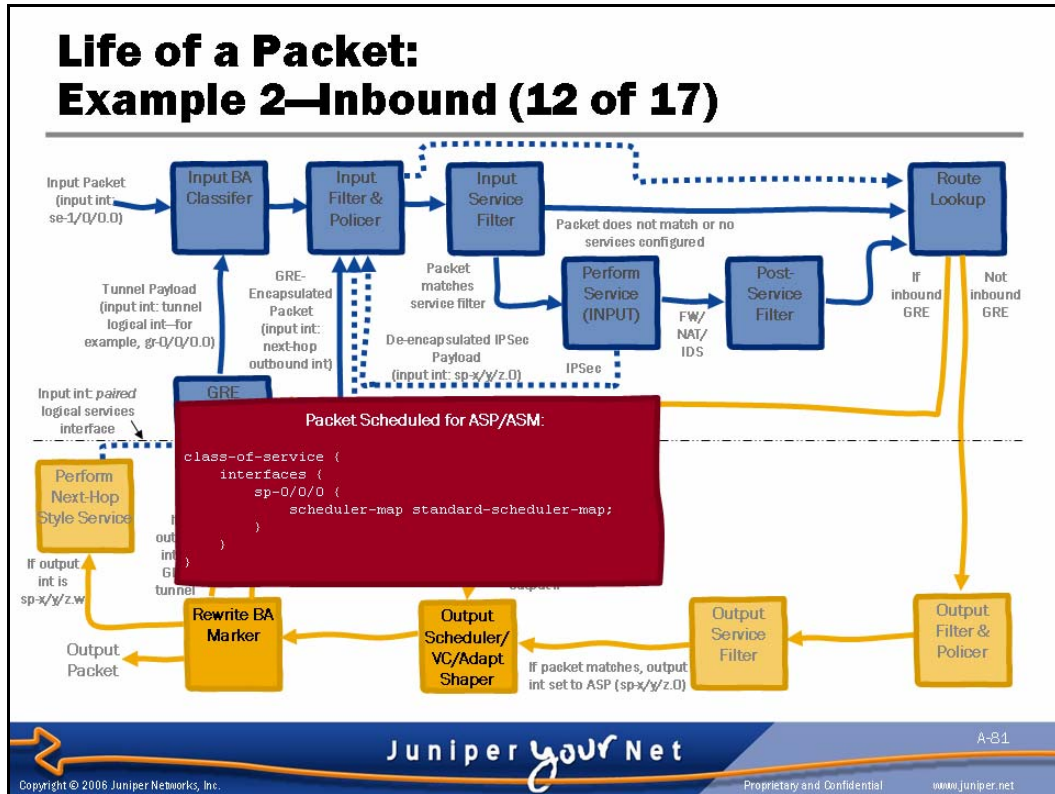
Because no input policer, input filter, or input service is configured for gr-0/0/0.0, the PFE proceeds to a route lookup.



### Route Lookup

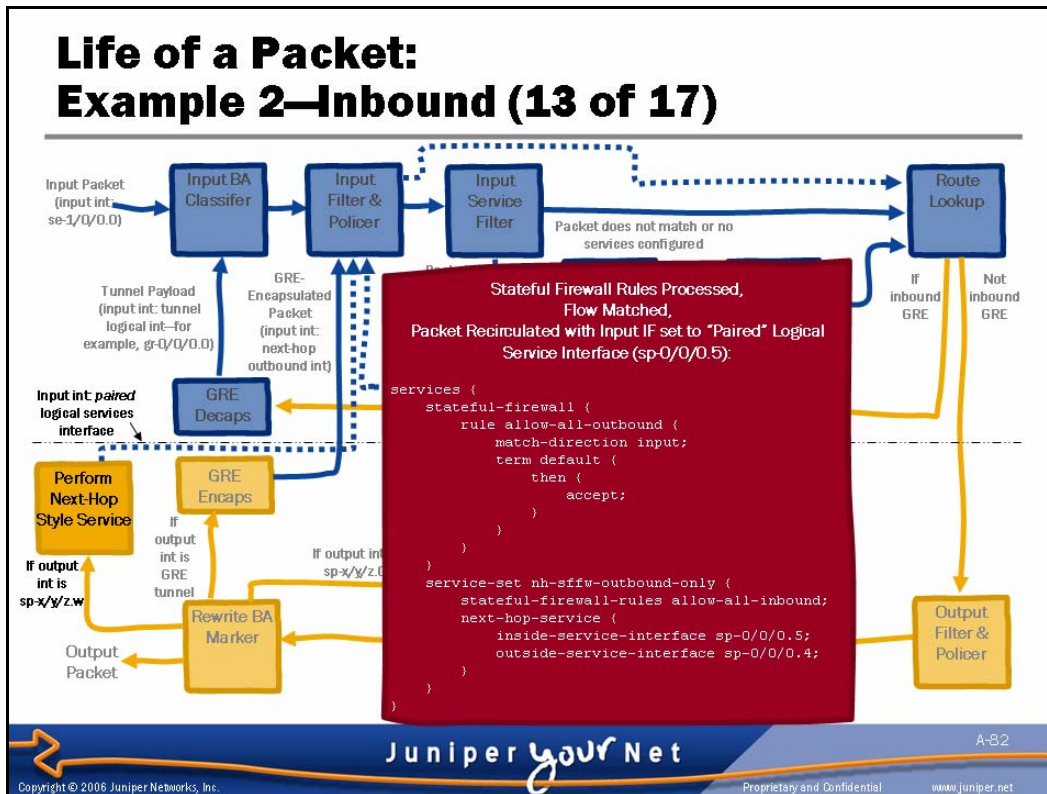
The router next looks up the destination (192.168.201.2) in the `inet` forwarding table and finds the next hop of `sp-0/0/0.4` (due to the static route for 192.168.201.0/24 with that interface as a next-hop).

Because no output policer or output filter is configured on the `sp-0/0/0.4` interface, the packet proceeds to CoS processing.



### Output CoS Processing (for sp-0/0/0.4)

The router processes the packet through the configuration for the sp-0/0/0.4 interface under [edit class-of-service interfaces]. The router uses *standard-scheduler-map* to schedule the traffic for transmission from the FPC to the PIC.

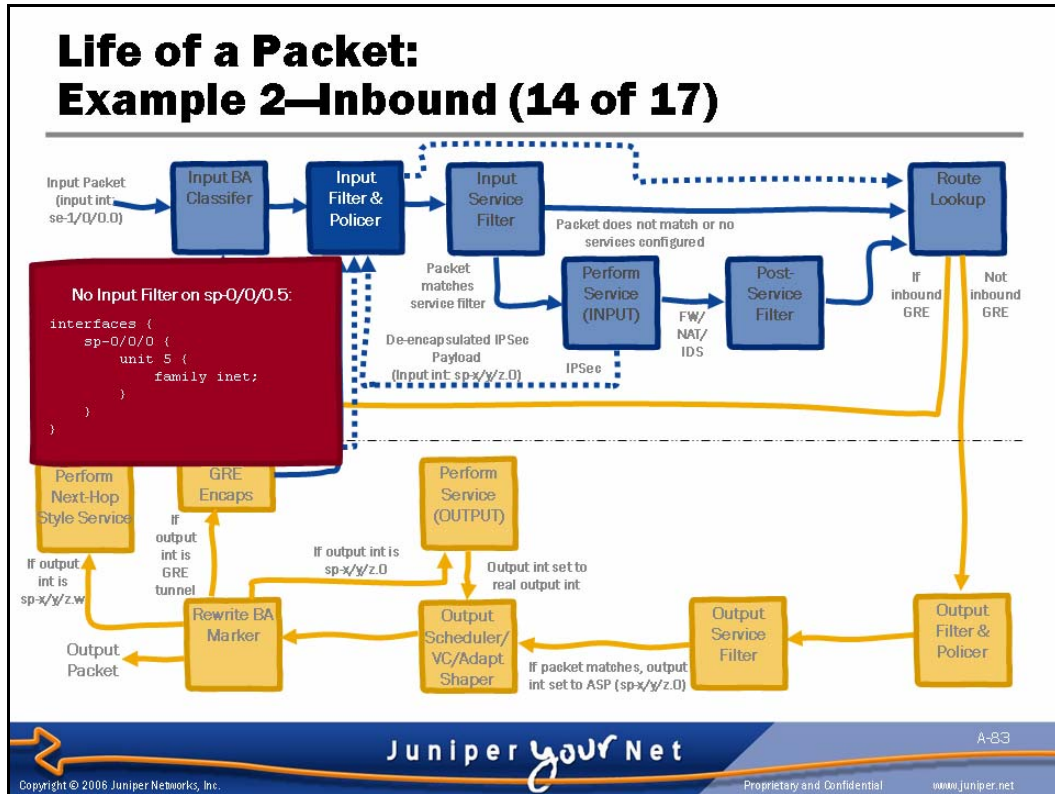


### Perform Next-Hop-Style Service

The entire packet is next sent to the ASP, where the next-hop-style service is processed. In this case, because `sp-0/0/0.4` is the `outside-service-interface` of the `nh-sffw-outbound-only` service set, that service set is performed.

The ASP processes the rules for the service set and finds a matching flow in its stateful firewall flow table. Therefore, the packet is accepted. Because this service set does not include NAT or IDS configuration, those services are not performed.

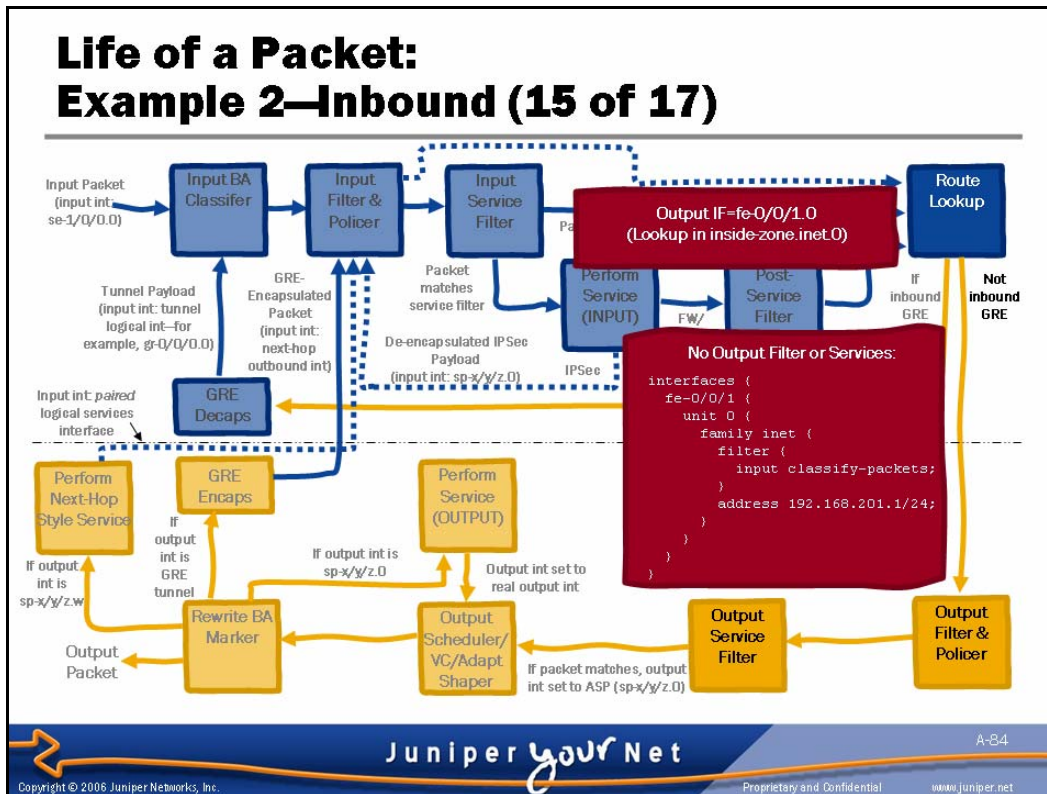
When the ASP finishes processing the packet, it sends the entire packet back to the FPC with the input interface set to the *paired* logical service interface. In this case, because the ASP received the packet on the `outside-service-interface` (`sp-0/0/0.4`), it sends the packet to the FPC with the input interface set to the `inside-service-interface` (`sp-0/0/0.5`).



### Input Filter and Policer (for sp-0/0/0.5)

The PFE next processes the packet through the input policer and input filter for sp-0/0/0.5. However, because none is configured, the PFE proceeds directly to a route lookup. (The dotted line that skips the input service filter represents it not being possible to configure services on service interfaces.)

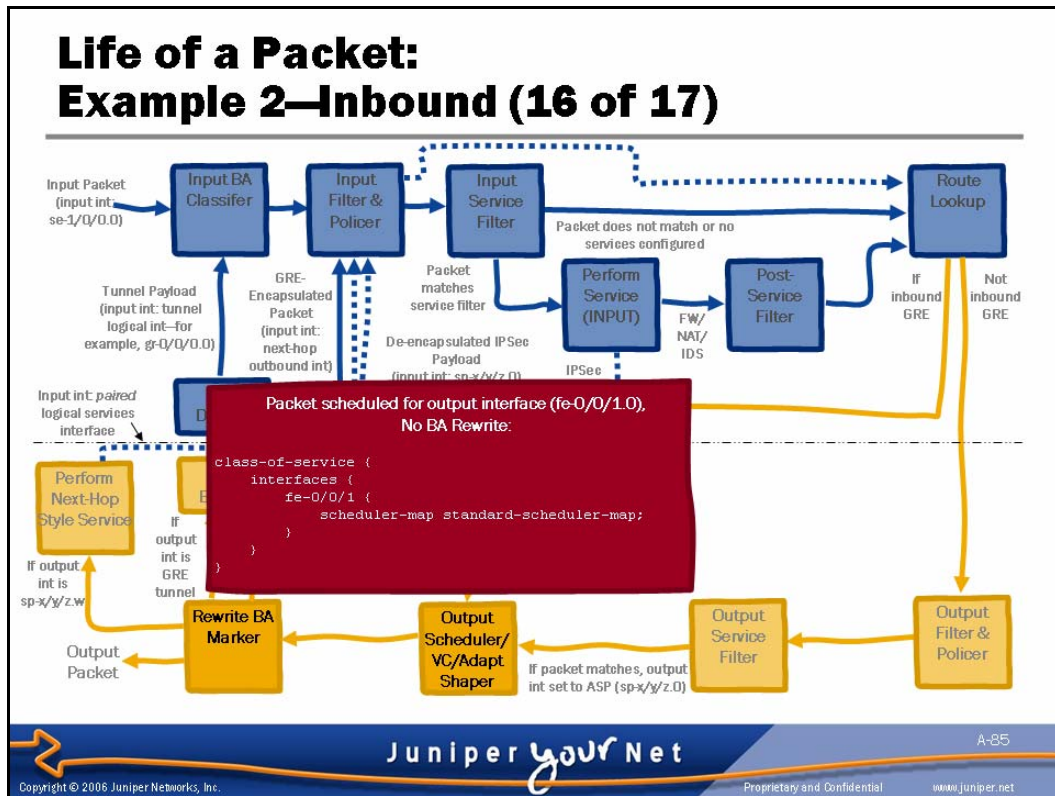




### Route Lookup and Output Processing (for fe-0/0/1.0)

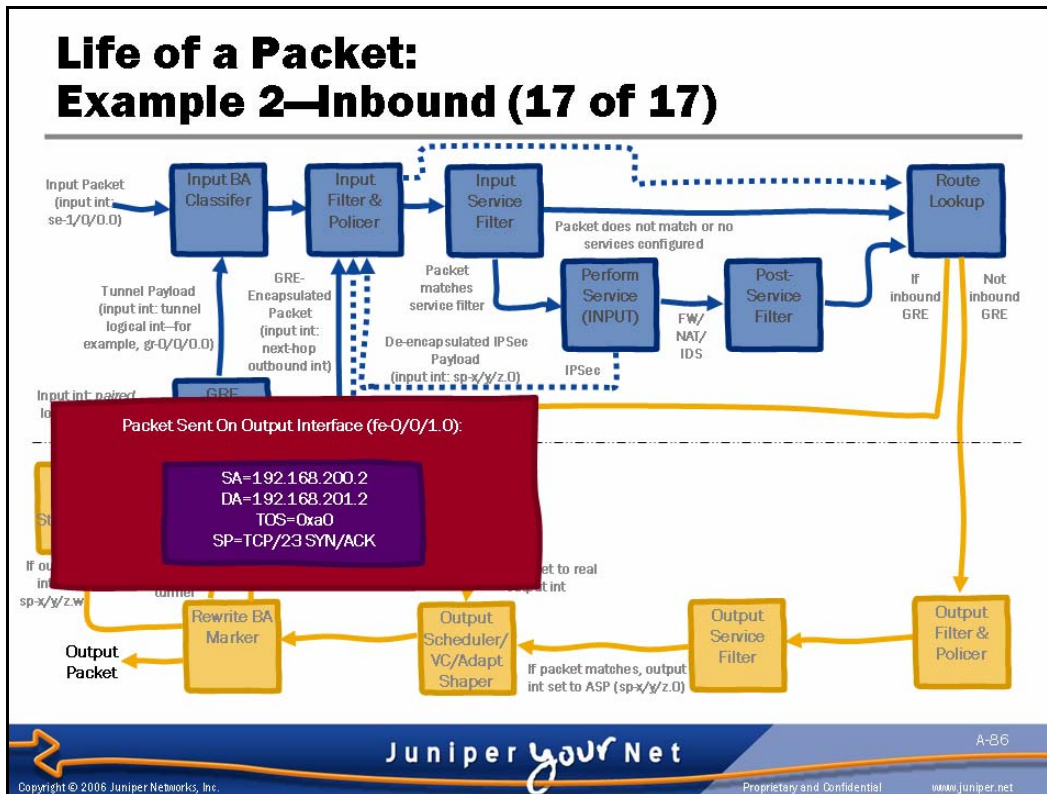
Because the sp-0/0/0.5 interface is in the `inside-zone` routing instance, the PFE looks up the route for the destination address (192.168.201.2) in the `inside-zone.inet` forwarding table. Because the fe-0/0/1.0 interface is also in the `inside-zone` routing instance, it finds the directly connected entry with an output interface of fe-0/0/1.0.

Because no output policer, output filter, or output service is configured for fe-0/0/1.0, the packet proceeds to CoS processing.



### CoS Output Processing

The packet is scheduled for transmission according to the scheduler map configured for `fe-0/0/1.0`. Because no rewrite rule is configured for `fe-0/0/1.0`, the original ToS field of the IP header is maintained.



### Packet Transmitted

After this entire process is completed, the PIC transmits the packet onto the wire.

The ESP packet was processed. The enclosed GRE packet was decrypted and processed. The de-encapsulated original packet was processed and transmitted to its destination.



## Summary

- In this chapter, we:
  - Discussed the way packets flow through a JUNOS router



## Summary

This chapter is designed to show the way that various services interact with each other as J-series, M-series, or T-series routers process packets. As you can see, many policy application points are available to influence packet processing.

The following list provides some important things to note:

- The de-encapsulation of GRE packets can change the forwarding class of a packet as the de-encapsulated packet is processed by the BA classifier (whether by the default BA classifier or an explicitly configured one) (see page A-16).
- When they are encapsulated, GRE packets are *recirculated* through the router with the inbound interface set to the next-hop output interface of the tunnel destination (see page A-24).
- The particular packet available for inspection by filters changes as the packet moves through the router. (For example, see the difference in the information available for the virtual-channel classification on page A-26 and page A-67.)
- To avoid service loops and to ensure that packets are routed correctly, you must understand how the route lookups occur when using the next-hop-style service sets (for example, see page A-59 and page A-63).

Understanding how a JUNOS router processes packets will enable you to configure the router wisely and understand the effects of any piece of configuration.

## Review Questions

1. When packets are encapsulated within GRE, how do the GRE packets get *recirculated* through the router?
2. What are *service loops* and how do you avoid them?
3. How can the forwarding class of GRE-encapsulated packets change when they are de-encapsulated, even if no BA classifier is configured?



### This Chapter Discussed:

- How a JUNOS router processes packets and the way various configuration options will impact packet processing.



# **Advanced Juniper Networks Routing in the Enterprise**

## **Appendix C: Sample Routing Engine Filter**

## Sample Routing Engine Filter

The following filter is a sample firewall filter you could use to protect your Routing Engine. You would apply this to the lo0 logical interface in the input direction:

```
policy-options {
  prefix-list mgmt-hosts {
    10.14.10.0/24;
    192.168.200.0/24;
  }
  prefix-list bgp-peers {
    apply-path "protocols bgp group <*> neighbor <*>";
  }
  prefix-list radius-servers {
    apply-path "system radius-server <*>";
  }
}
firewall {
  family inet {
    filter protect-re {
      term allow-established {
        from {
          protocol tcp;
          tcp-established;
        }
        then accept;
      }
      term allow-telnet-ssh-ftp {
        from {
          source-prefix-list {
            internal-routes;
          }
          protocol tcp;
          destination-port [ ssh telnet ftp ];
        }
        then accept;
      }
      term allow-bgp {
        from {
          source-prefix-list {
            bgp-peers;
          }
          protocol tcp;
          destination-port 179;
        }
        then accept;
      }
      term allow-ospf {
        from {
          protocol ospf;
        }
        then accept;
      }
    }
  }
}
```

```

    }
    term allow-vrrp {
        from {
            protocol vrrp;
        }
        then accept;
    }
    term permit-some-icmp {
        from {
            protocol icmp;
            icmp-type [ echo-request echo-reply time-exceeded
unreachable ];
        }
        then accept;
    }
    term permit-dns-replies {
        from {
            protocol udp;
            source-port 53;
        }
        then accept;
    }
    term permit-ntp {
        /* If you choose to limit by source address, ensure you include
the lo0 address and the router's configured NTP source address */
        from {
            protocol udp;
            port 123;
        }
        then accept;
    }
    term snmp {
        from {
            protocol udp;
            port [ snmp snmptrap ];
        }
        then accept;
    }
    term radius {
        from {
            source-prefix-list {
                radius-servers;
            }
            protocol udp;
            source-port [ 1645 1646 1812 1813 ];
        }
        then accept;
    }
    term permit-traceroute {
        from {
            fragment-offset 0;
            fragment-flags " !more-fragments ";
            protocol udp;
            ttl 1;

```

```
        }  
        then accept;  
    }  
}  
}
```



# **Advanced Juniper Networks Routing in the Enterprise**

## **Appendix D: Sample IPSec-over-GRE VPN Configuration**

## Introduction

This appendix provides a sample configuration for connecting a Juniper Networks router and Cisco router with an IPSec-over-GRE VPN. The Juniper Networks router's physical interface is `fe-2/0/1`, which is configured with IP address 172.17.37.4. The Cisco router's physical interface is `FastEthernet0`, with IP address 172.17.38.4.

## Sample Juniper Networks Configuration

Relevant portions of the Juniper Networks configuration follow:

```
interfaces {
  gr-0/0/0 {
    unit 0 {
      tunnel {
        source 172.17.37.4;
        destination 172.17.38.4;
      }
      family inet {
        address 192.168.25.129/30;
      }
    }
  }
  fe-2/0/1 {
    unit 0 {
      family inet {
        service {
          input {
            service-set gre-vpn service-filter match-vpn-no-gre;
          }
          output {
            service-set gre-vpn service-filter match-vpn;
          }
        }
        address 172.17.37.4/29;
      }
    }
  }
}
firewall {
  family inet {
    service-filter match-vpn {
      term vpn {
        from {
          source-address {
            172.17.37.4/32;
          }
          destination-address {
            172.17.38.4/32;
          }
        }
      }
    }
    then service;
  }
}
```



```

    }
    term default {
        then skip;
    }
}
service-filter no-gre {
    term ignore-gre {
        from {
            protocol gre;
        }
        then skip;
    }
    term default {
        then service;
    }
}
service-filter match-vpn-no-gre {
    term ignore-gre {
        from {
            protocol gre;
        }
        then skip;
    }
    term vpn {
        from {
            source-address {
                172.17.38.4/32;
            }
            destination-address {
                172.17.37.4/32;
            }
        }
        then service;
    }
    term default {
        then skip;
    }
}
}
}
services {
    service-set gre-vpn {
        interface-service {
            service-interface sp-0/0/0;
        }
        ipsec-vpn-options {
            local-gateway 172.17.37.4;
        }
        ipsec-vpn-rules vpn-to-cisco;
    }
    ipsec-vpn {
        rule vpn-to-cisco {
            term gre-tunnel {
                from {

```

```

        source-address {
            172.17.37.4/32;
        }
        destination-address {
            172.17.38.4/32;
        }
    }
    then {
        remote-gateway 172.17.38.4;
        dynamic {
            ike-policy main_mode_ike_policy;
            ipsec-policy dynamic_ipsec_policy;
        }
    }
}
match-direction output;
}
ipsec {
    proposal cisco_compat {
        protocol esp;
        authentication-algorithm hmac-md5-96;
        encryption-algorithm des-cbc;
    }
    policy dynamic_ipsec_policy {
        perfect-forward-secrecy {
            keys group1;
        }
        proposals cisco_compat;
    }
}
ike {
    proposal cisco-compat {
        authentication-method pre-shared-keys;
        authentication-algorithm md5;
        dh-group group1;
        encryption-algorithm des-cbc;
    }
    policy main_mode_ike_policy {
        proposals cisco-compat;
        pre-shared-key ascii-text "$9$hEycK8-ds4JDwY"; ## SECRET-DATA
    }
}
establish-tunnels immediately;
}
}

```

### Sample Cisco Configuration

Relevant portions of the Cisco configuration follow:

```

crypto isakmp policy 1
  hash md5
  authentication pre-share
crypto isakmp key test address 172.17.37.4

```

```
crypto isakmp keepalive 10 2 periodic

!
!
crypto ipsec transform-set esp_des_set esp-des esp-md5-hmac
!
!
crypto map gre-to-juniper 1 ipsec-isakmp
  set peer 172.17.37.4
  set transform-set esp_des_set
  set pfs group1
  match address 110

access-list 110 permit ip host 172.17.38.4 host 172.17.37.4

interface tunnell
  ip address 192.168.25.130 255.255.255.252
  tunnel mode gre ip
  tunnel destination 172.17.37.4
  tunnel source 172.17.38.4

interface fast0
  crypto map gre-to-juniper
  ip address 172.17.38.4 255.255.255.0
```

