Junos Routing Essentials

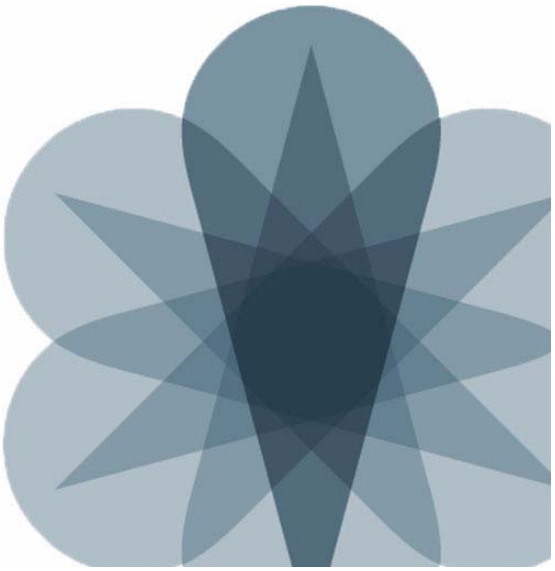
V-12.b

Student Guide

Worldwide Education Services

1133 Innovation Way
Sunnyvale, CA 94089
USA
408-745-2000
www.juniper.net

Course Number: EDU-JUN-JRE



This document is produced by Juniper Networks, Inc.

Junos Routing Essentials Student Guide, Revision V-12.b

Revision History:

Revision 9.a—July 2009

Revision 9.b—October 2009

Revision 10.a—May 2010

Revision 10.b—December 2010

Revision 11.a—June 2011

Revision 12.a—June 2012

Revision V-12.b—September 2016

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 12.1X47-D30.4. 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 operating system 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 Juniper Networks software are described in the software license provided with the software, or to the extent applicable, in an agreement executed between you and Juniper Networks, or Juniper Networks agent. By using Juniper Networks software, you indicate that you understand and agree to be bound by its license terms and conditions. Generally speaking, the software license restricts the manner in which you are permitted to use the Juniper Networks software, may contain prohibitions against certain uses, and may state conditions under which the license is automatically terminated. You should consult the software license for further details.

Contents

Course Overview

This one-day course provides students with foundational routing knowledge and configuration examples and includes an overview of general routing concepts, routing policy, and firewall filters. This course is based on Junos operating system Release 12.1X47-D30.4.

Through demonstrations and hands-on labs, students will gain experience in configuring and monitoring the Junos operating system and monitoring basic device operations.

Intended Audience

This course benefits individuals responsible for configuring and monitoring devices running the Junos OS.

Course Level

The *Junos Routing Essentials* (JRE) course is a one-day introductory course.

Prerequisites

Students should have basic networking knowledge and an understanding of the Open Systems Interconnection (OSI) reference model and the TCP/IP protocol suite. Students should also attend the *Introduction to the Junos Operating System* (IJOS) course prior to attending this class.

Objectives

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

- Explain basic routing operations and concepts.
- View and describe routing and forwarding tables.
- Configure and monitor static routing.
- Configure and monitor OSPF.
- Describe the framework for routing policy.
- Explain the evaluation of routing policy.
- Identify situations where you might use routing policy.
- Write and apply a routing policy.
- Describe the framework for firewall filters.
- Explain the evaluation of firewall filters.
- Identify instances where you might use firewall filters.
- Write and apply a firewall filter.
- Describe the operation and configuration for unicast reverse path forwarding (RPF).

Course Agenda

Day 1

Chapter 1: Course Introduction

Chapter 2: Routing Fundamentals

Lab 1: Routing Fundamentals

Chapter 3: Routing Policy

Lab 2: Routing Policy

Chapter 4: Firewall Filters

Lab 3: Firewall Filters

Appendix A: Class of Service

Lab 4: Class of Service (Optional)

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: • Screen captures • Noncommand-related syntax	`commit complete` `Exiting configuration mode`
	GUI text elements: • Menu names • Text field entry	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 these instances will be shown in the context of where you must enter them. We use bold style to distinguish text that is input versus text that is simply displayed.

Style	Description	Usage Example
`Normal CLI` `Normal GUI`	No distinguishing variant.	`Physical interface:fxp0, Enabled` View configuration history by clicking `Configuration > History`.
`CLI Input` **`GUI Input`**	Text that you must enter.	`lab@San_Jose>` **`show route`** Select `File > Save`, and type **`config.ini`** 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
`CLI Variable` *`GUI Variable`*	Text where variable value is already assigned.	`policy` *`my-peers`* Click *`my-peers`* in the dialog.
`CLI Undefined` *`GUI Undefined`*	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 user must input.	Type **`set policy`** ***`policy-name`***. **`ping 10.0.`*`x.y`* Select `File > Save`, and type ***`filename`*** 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 *Junos Routing Essentials Student Guide* was developed and tested using software Release 12.1X47-D30.4. Previous and later versions of software might 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.

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.

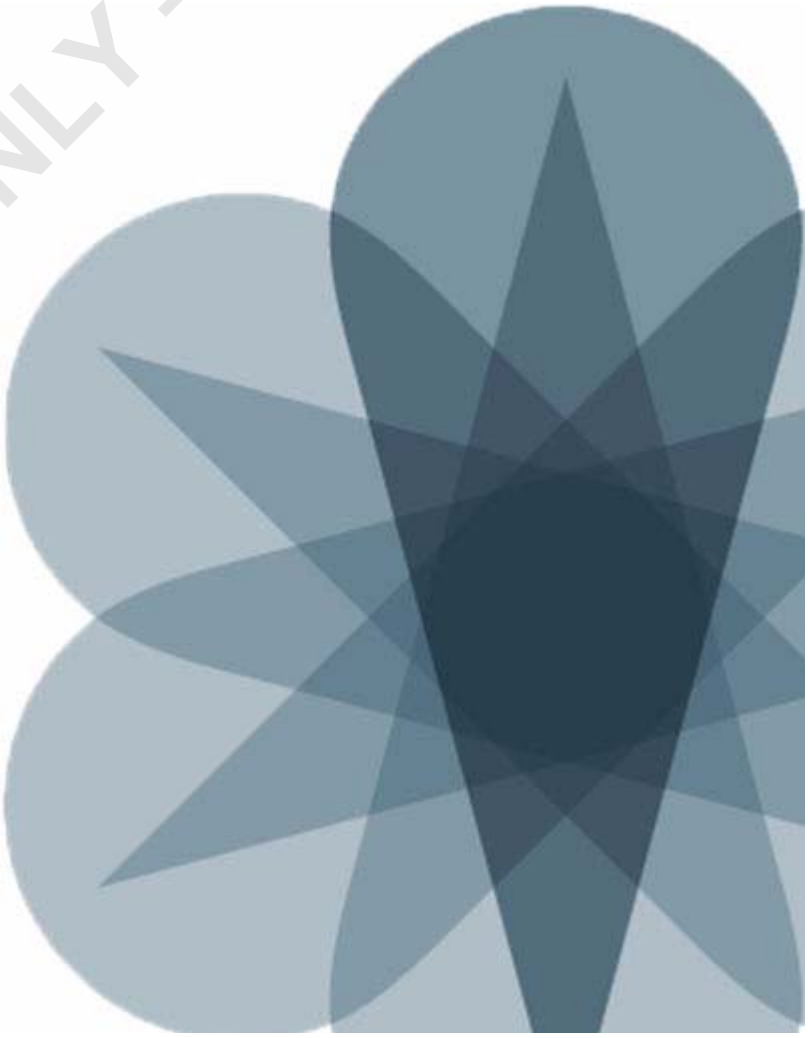
Documentation sets and CDs are available through your local Juniper Networks sales office or account representative.

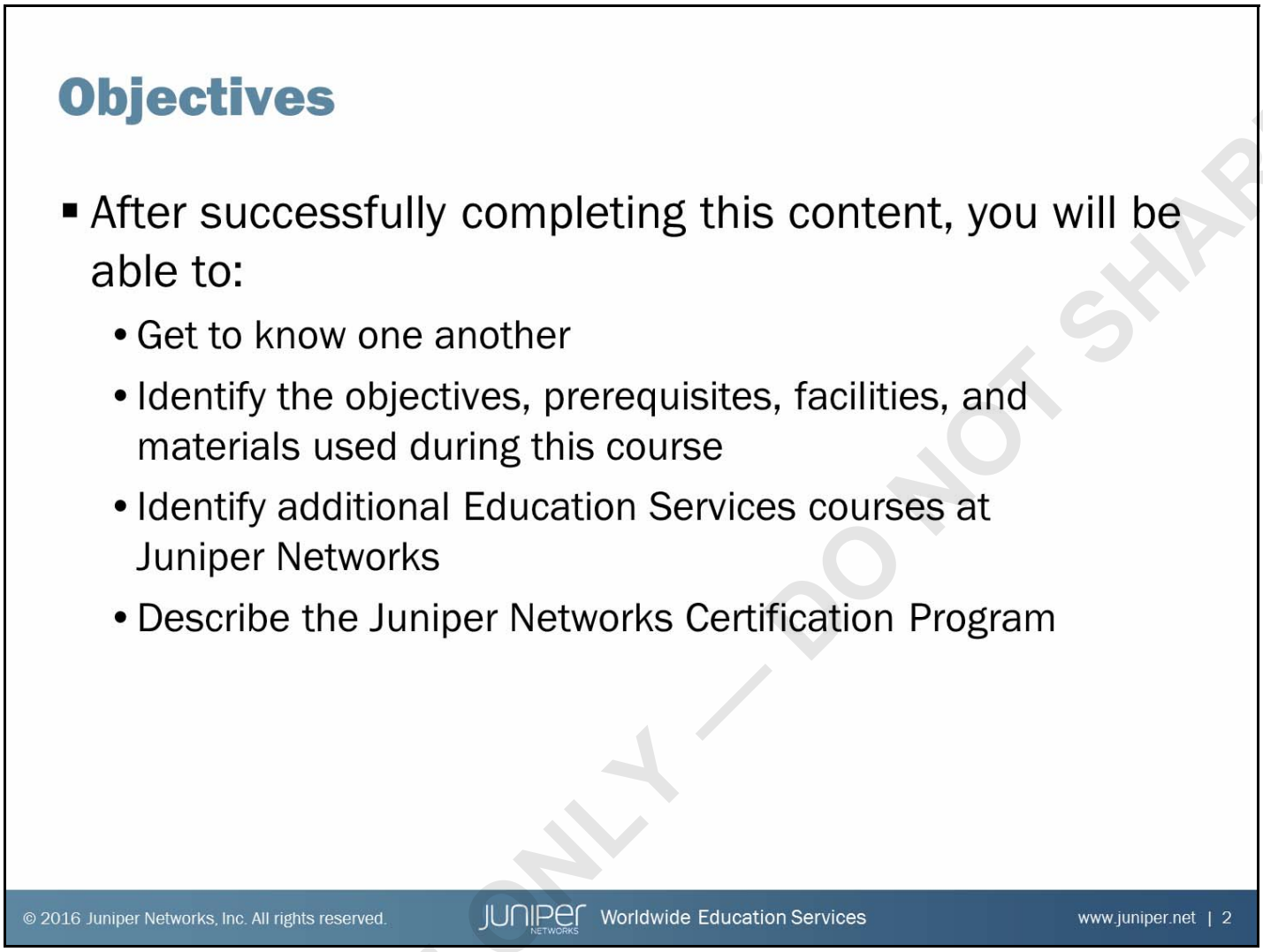
Juniper Networks Support

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).

Junos Routing Essentials

Chapter 1: Course Introduction





We Will Discuss:

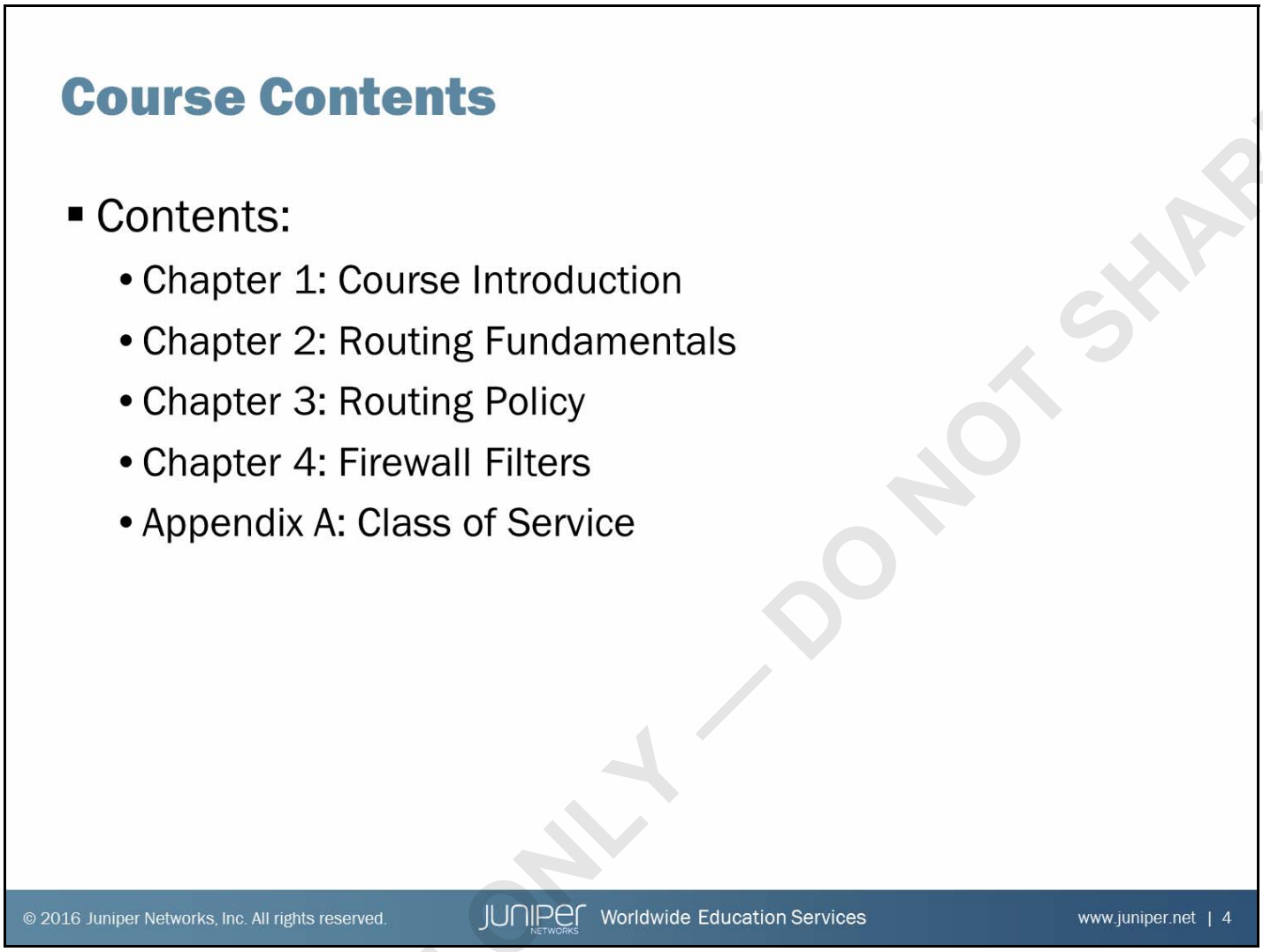
- Objectives and course content information;
- Additional Juniper Networks, Inc. courses; and
- The Juniper Networks Certification Program.

Introductions

- Before we get started...
  - What is your name?
  - Where do you work?
  - What is your primary role in your organization?
  - What kind of network experience do you have?
  - Are you certified on Juniper Networks?
  - What is the most important thing for you to learn in this training session?

Introductions

The slide asks several questions for you to answer during class introductions.



Course Contents

The slide lists the topics we discuss in this course.

Prerequisites

- The prerequisites for this course are the following:
  - Basic networking knowledge
  - Understanding of the Open Systems Interconnection reference model and TCP/IP
  - The *Introduction to the Junos Operating System* (IJOS) course or equivalent knowledge

Prerequisites

The slide lists the prerequisites for this course.

Course Administration

- The basics:
  - Sign-in sheet
  - Schedule
    - Class times
    - Breaks
    - Lunch
  - Break and restroom facilities
  - Fire and safety procedures
  - Communications
    - Telephones and wireless devices
    - Internet access

General Course Administration

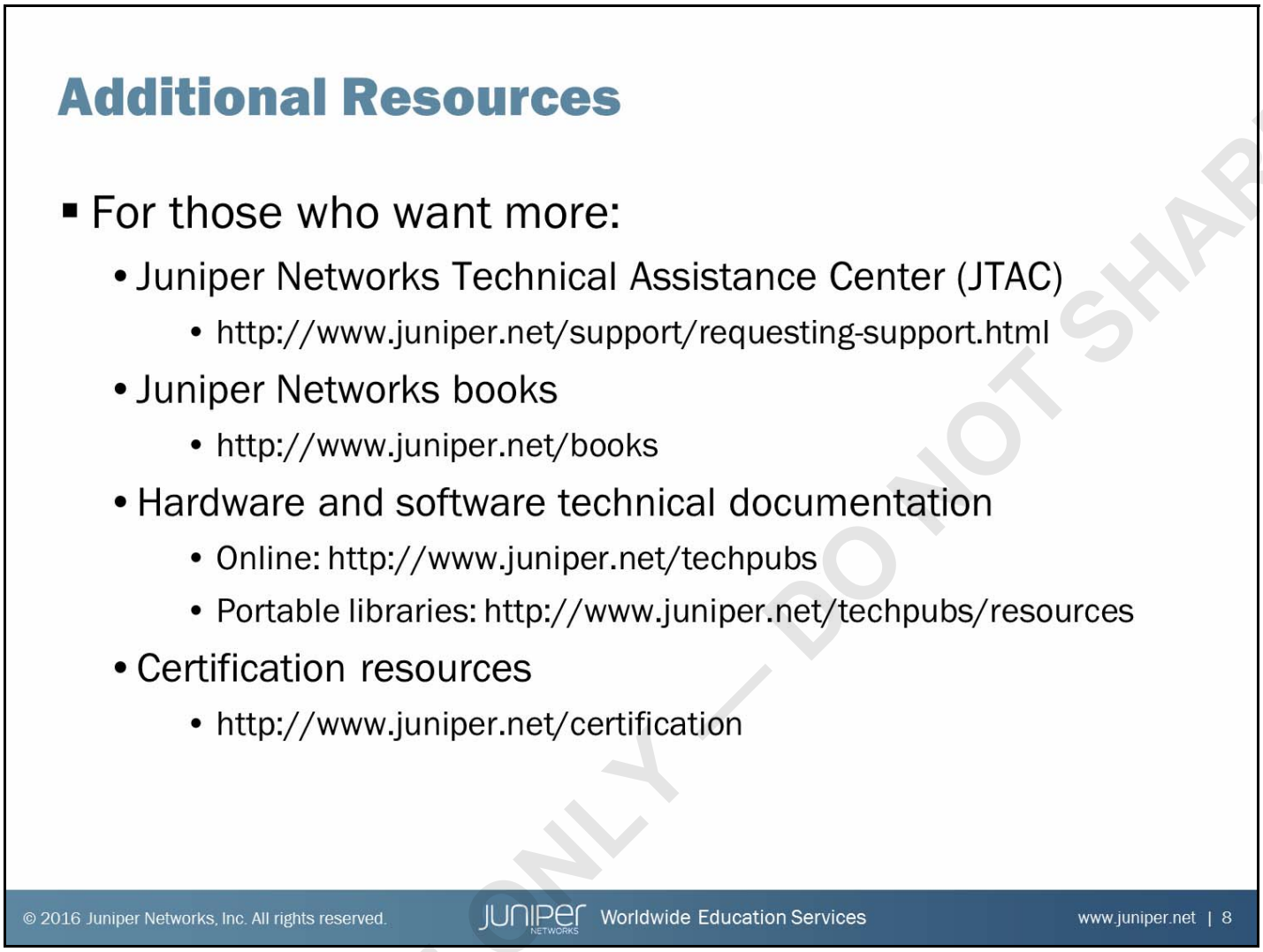
The slide documents general aspects of classroom administration.

Education Materials

- Available materials for classroom-based and instructor-led online classes:
  - Lecture material
  - Lab guide
  - Lab equipment
- Self-paced online courses also available
  - http://www.juniper.net/courses

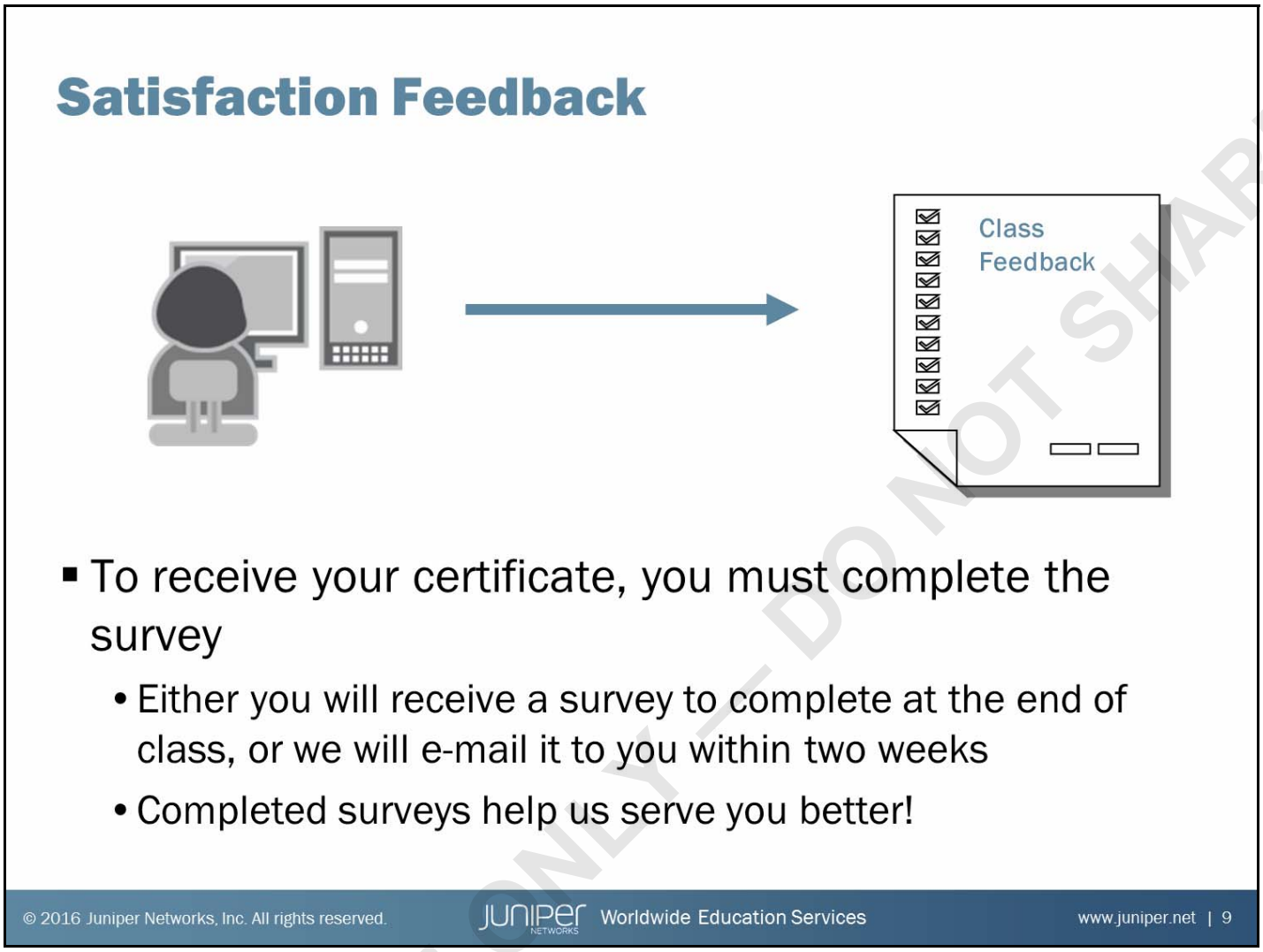
Training and Study Materials

The slide describes Education Services materials that are available for reference both in the classroom and online.



Additional Resources

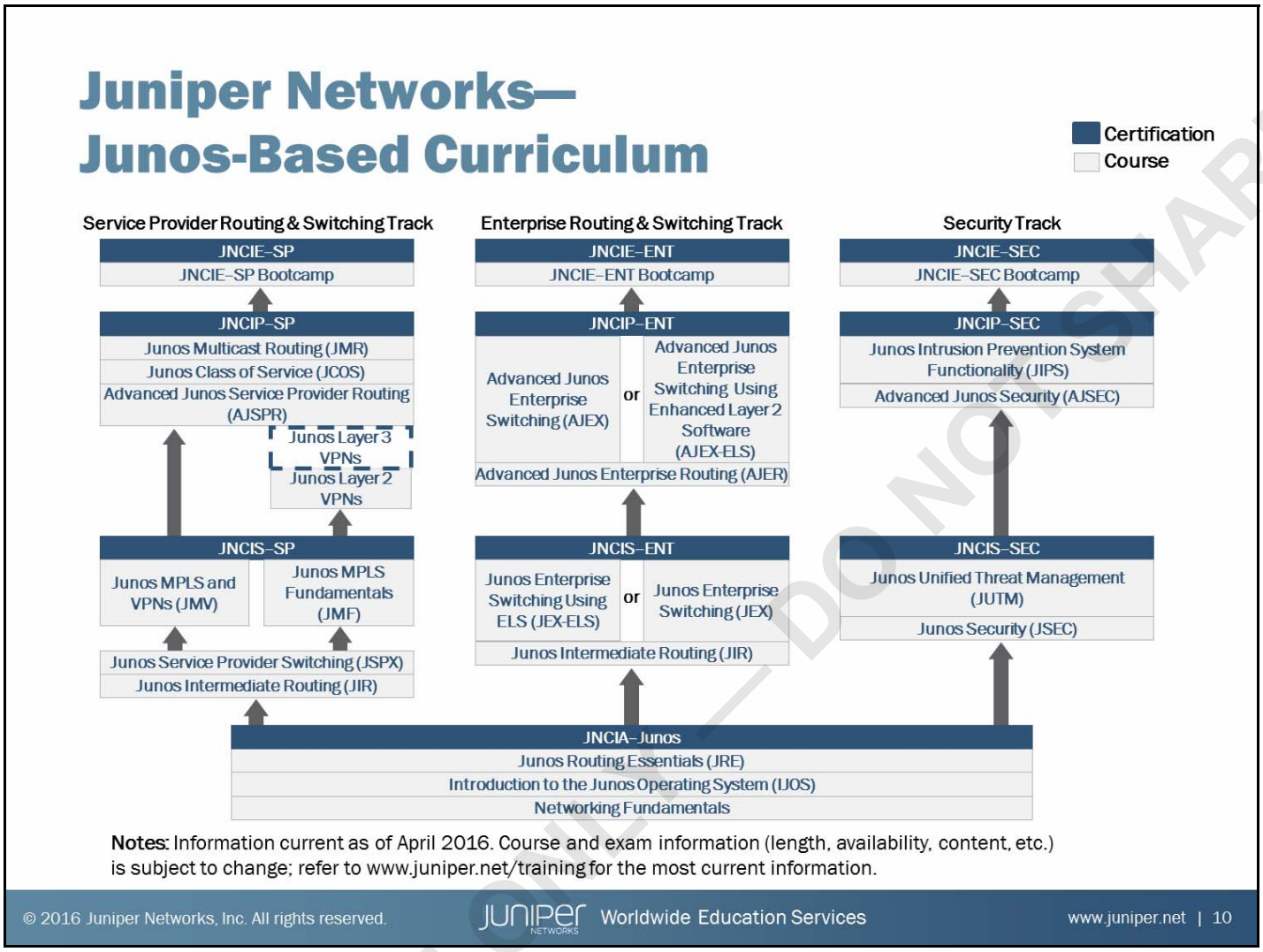
The slide provides links to additional resources available to assist you in the installation, configuration, and operation of Juniper Networks products.



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.



Juniper Networks Education Services Curriculum

Juniper Networks Education Services can help ensure that you have the knowledge and skills to deploy and maintain cost-effective, high-performance networks for both enterprise and service provider environments. We have expert training staff with deep technical and industry knowledge, providing you with instructor-led hands-on courses in the classroom and online, as well as convenient, self-paced eLearning courses. In addition to the courses shown on the slide, Education Services offers training in automation, E-Series, firewall/VPN, IDP, network design, QFabric, support, and wireless LAN.

Courses

Juniper Networks courses are available in the following formats:

- Classroom-based instructor-led technical courses
- Online instructor-led technical courses
- Hardware installation eLearning courses as well as technical eLearning courses
- Learning bytes: Short, topic-specific, video-based lessons covering Juniper products and technologies

Find the latest Education Services offerings covering a wide range of platforms at http://www.juniper.net/training/technical_education/.

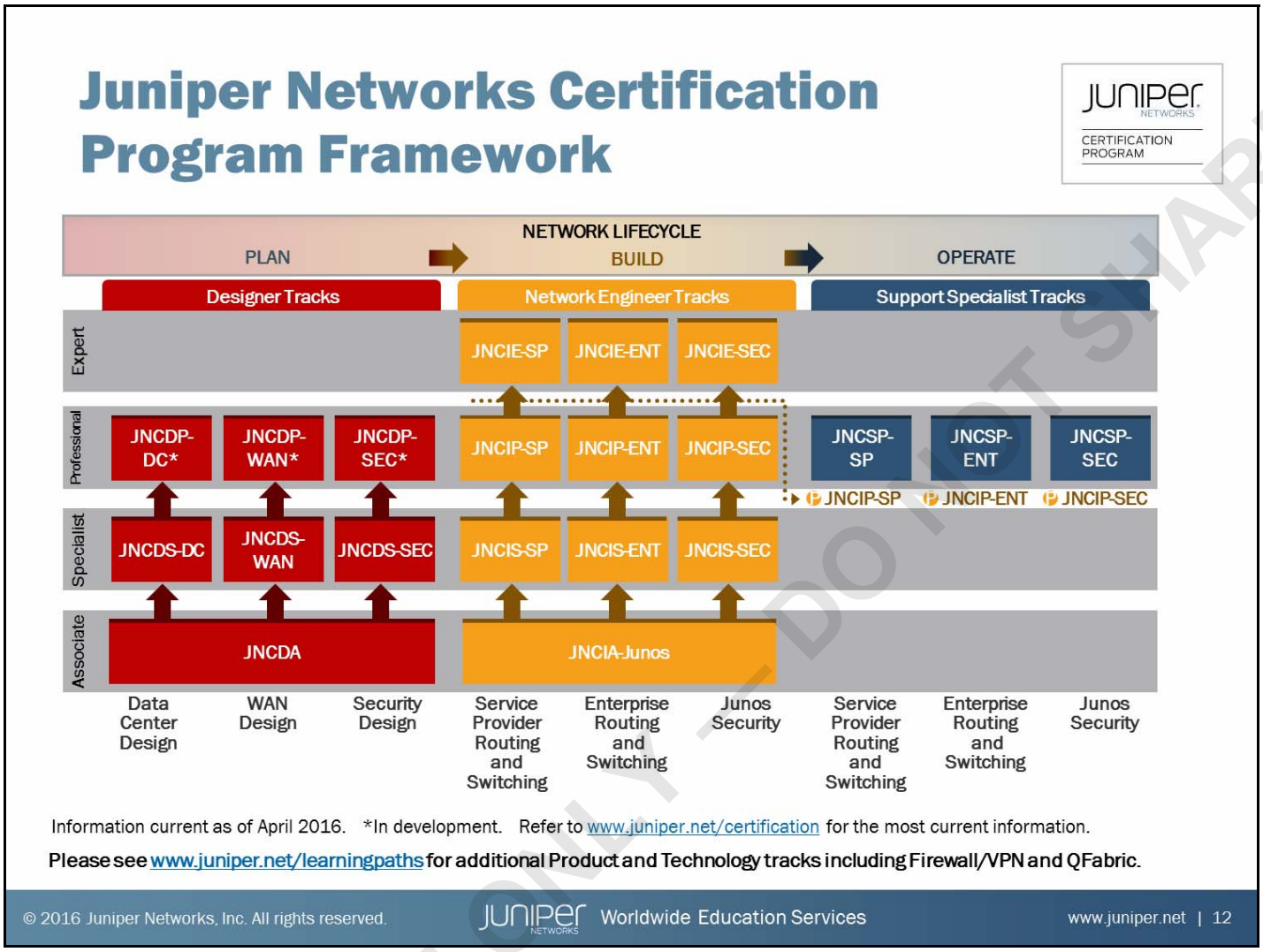
Juniper Networks Certification Program

- Why earn a Juniper Networks certification?
  - Juniper Networks certification makes you stand out
    - Demonstrate your solid understanding of networking technologies
    - Distinguish yourself and grow your career
  - Capitalize on the promise of the New Network
    - Develop and deploy the services you need
    - Lead the way and increase your value
  - Unique benefits for certified individuals

JUNIPER NETWORKS
CERTIFICATION PROGRAM

Juniper Networks Certification Program

A Juniper Networks certification is the benchmark of skills and competence on Juniper Networks technologies.



Juniper Networks Certification Program Overview

The Juniper Networks Certification Program (JNCP) consists of platform-specific, multitiered tracks that enable participants to demonstrate competence with Juniper Networks technology through a combination of written proficiency exams and hands-on configuration and troubleshooting exams. Successful candidates demonstrate a thorough understanding of Internet and security technologies and Juniper Networks platform configuration and troubleshooting skills.

The JNCP offers the following features:

- Multiple tracks;
- Multiple certification levels;
- Written proficiency exams; and
- Hands-on configuration and troubleshooting exams.

Each JNCP track has one to four certification levels—Associate-level, Specialist-level, Professional-level, and Expert-level. The Associate-level, Specialist-level, and Professional-level exams are computer-based exams composed of multiple choice questions administered at Pearson VUE testing centers worldwide.

Expert-level exams are composed of hands-on lab exercises administered at select Juniper Networks testing centers. Please visit the JNCP website at http://www.juniper.net/certification for detailed exam information, exam pricing, and exam registration.

Certification Preparation

- Training and study resources:
  - Juniper Networks Certification Program website: www.juniper.net/certification
  - Education Services training classes: www.juniper.net/training
  - Juniper Networks documentation and white papers: www.juniper.net/techpubs
- Community:
  - J-Net: http://forums.juniper.net/t5/Training-Certification-and/bd-p/Training_and_Certification
  - Twitter: @JuniperCertify

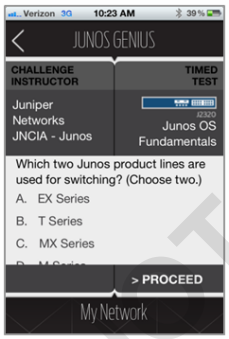
Preparing and Studying

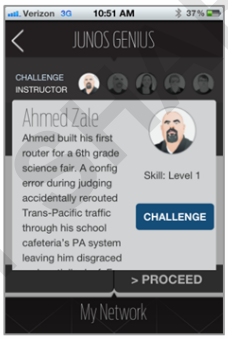
The slide lists some options for those interested in preparing for Juniper Networks certification.

Junos Genius: Certification Preparation App

Unlock your Genius...

- Practice for multiple exams in *Study Mode*
  - Hundreds of multiple choice questions and answer explanations, many with CLI snapshots
- Simulate an exam in *Time Challenge Mode*
- Earn device achievements by winning in *Instructor Challenge Mode*
- Build a virtual network with device achievements
- Track your results in the app and Game Center; share your network through Facebook and Twitter





ANDROID APP ON Google play

Available on the App Store

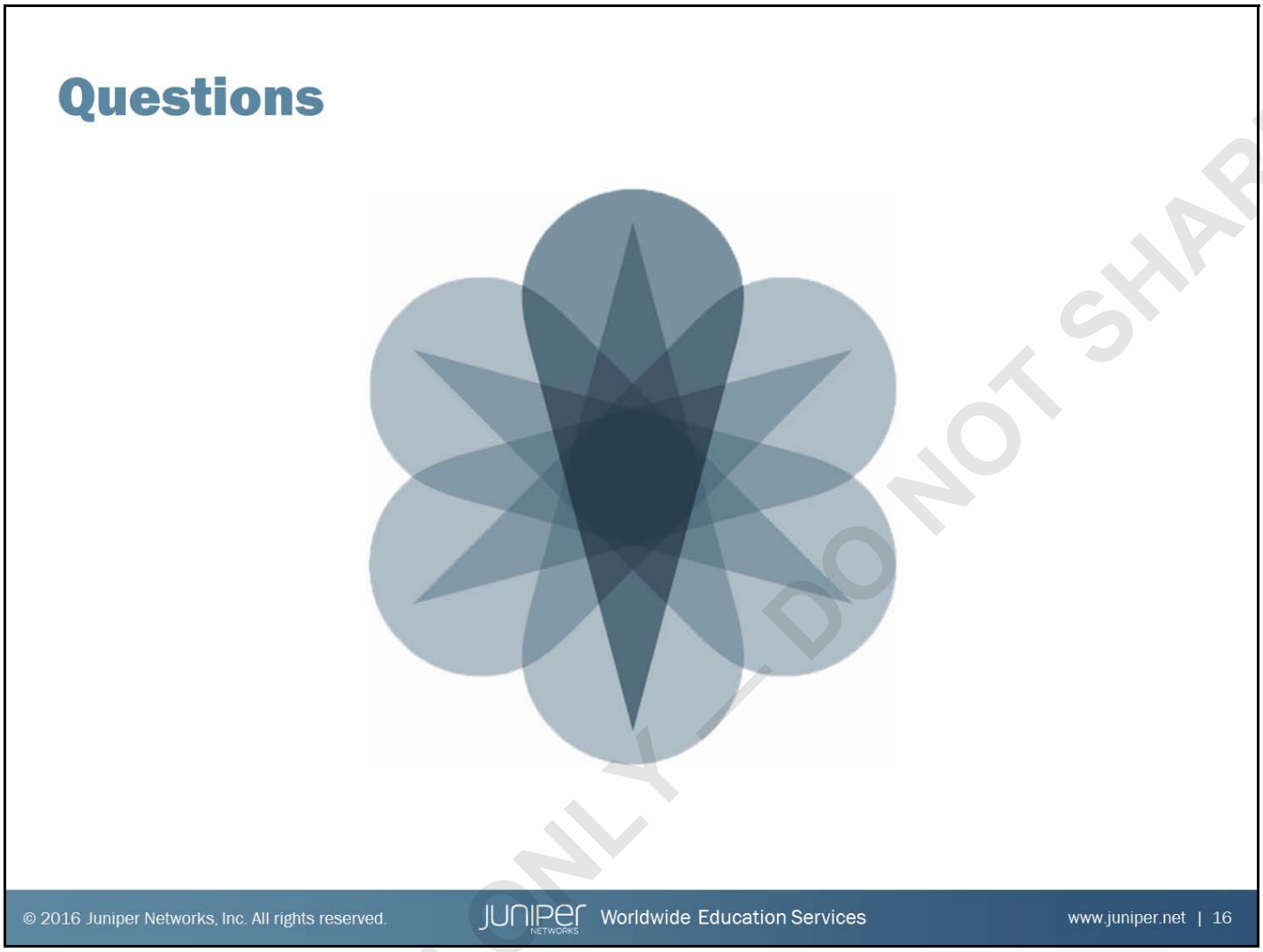
www.juniper.net/junosgenius

Junos Genius

The Junos Genius application takes certification exam preparation to a new level. With Junos Genius you can practice for your exam with flashcards, simulate a live exam in a timed challenge, and even build a virtual network with device achievements earned by challenging Juniper instructors. Download the app now and *Unlock your Genius today!*

Find Us Online

The slide lists some online resources to learn and share information about Juniper Networks.

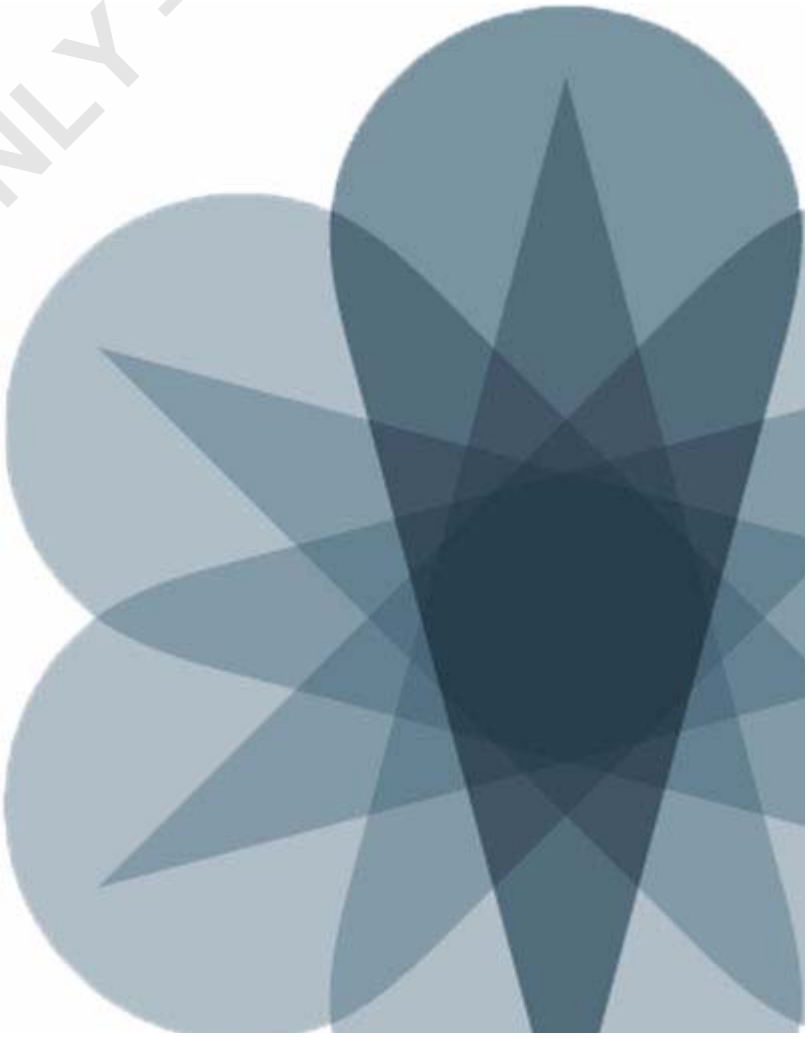


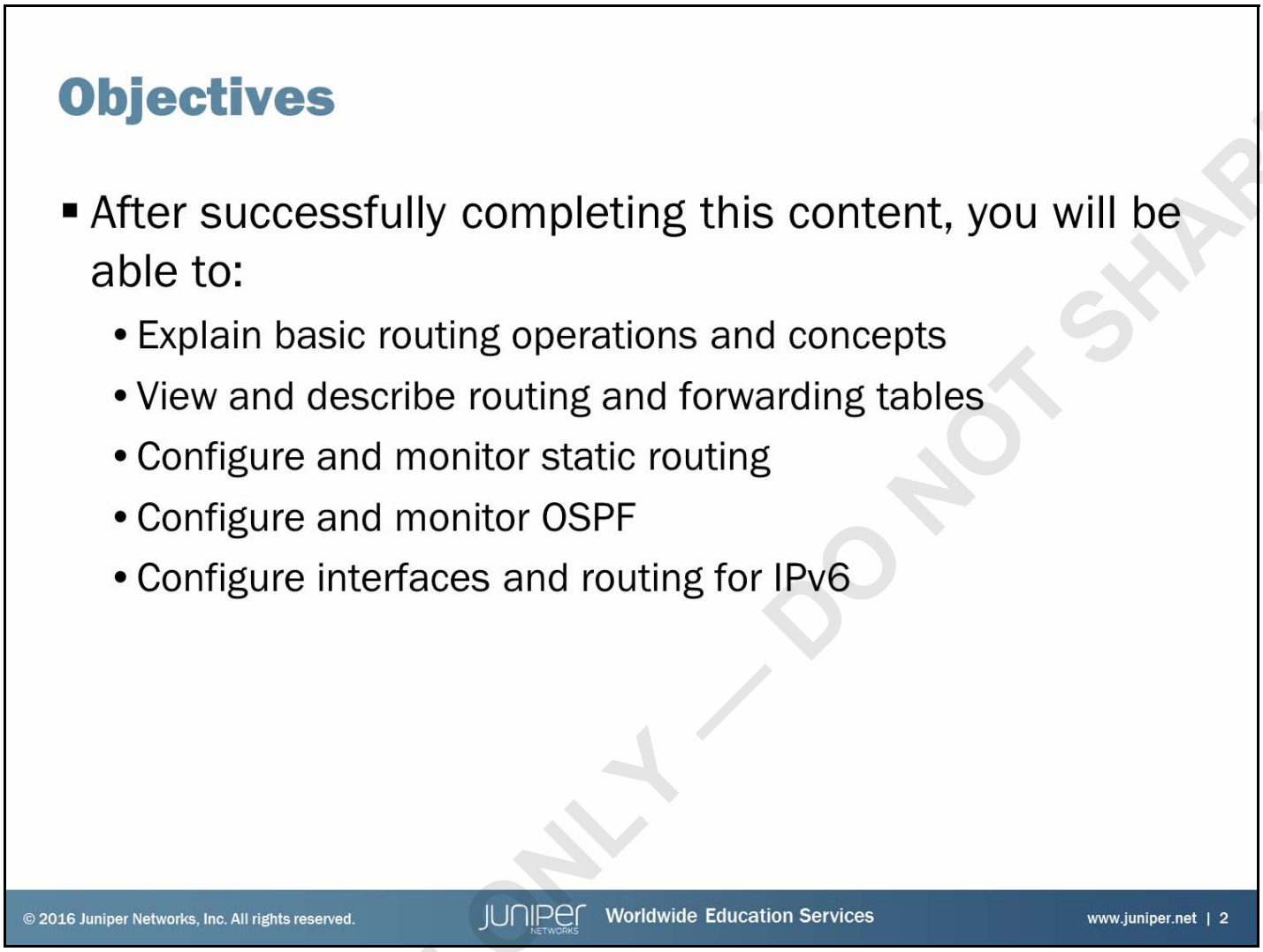
Any Questions?

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.

Junos Routing Essentials

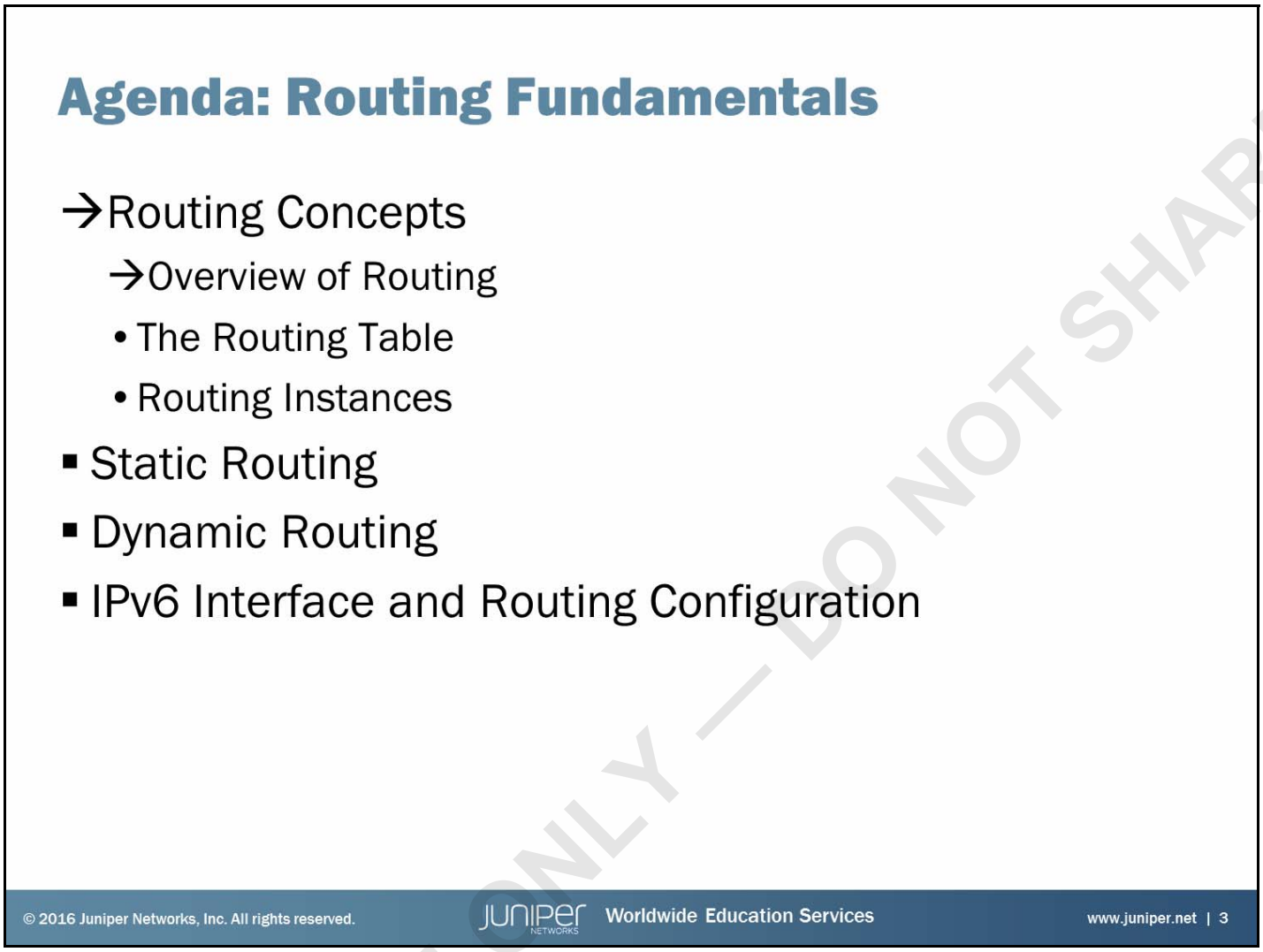
Chapter 2: Routing Fundamentals





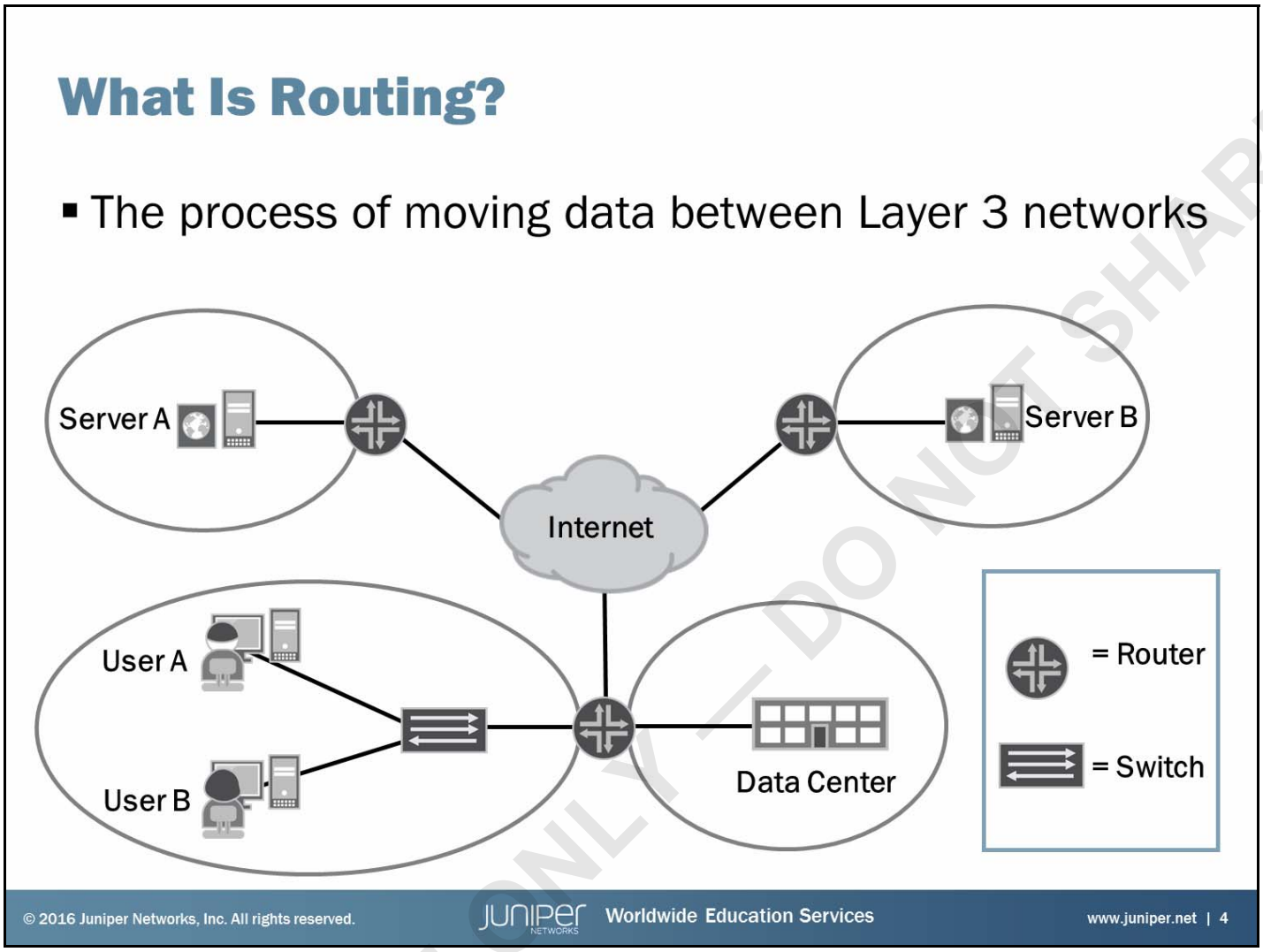
We Will Discuss:

- Basic routing operations and concepts;
- Routing and forwarding tables;
- Configuration and monitoring of static routing;
- Configuration and monitoring of basic OSPF. and
- Basic Interface and routing configuration for IPv6.



Routing Concepts: Overview of Routing

The slide lists the topics we will discuss. We discuss the highlighted topic first.



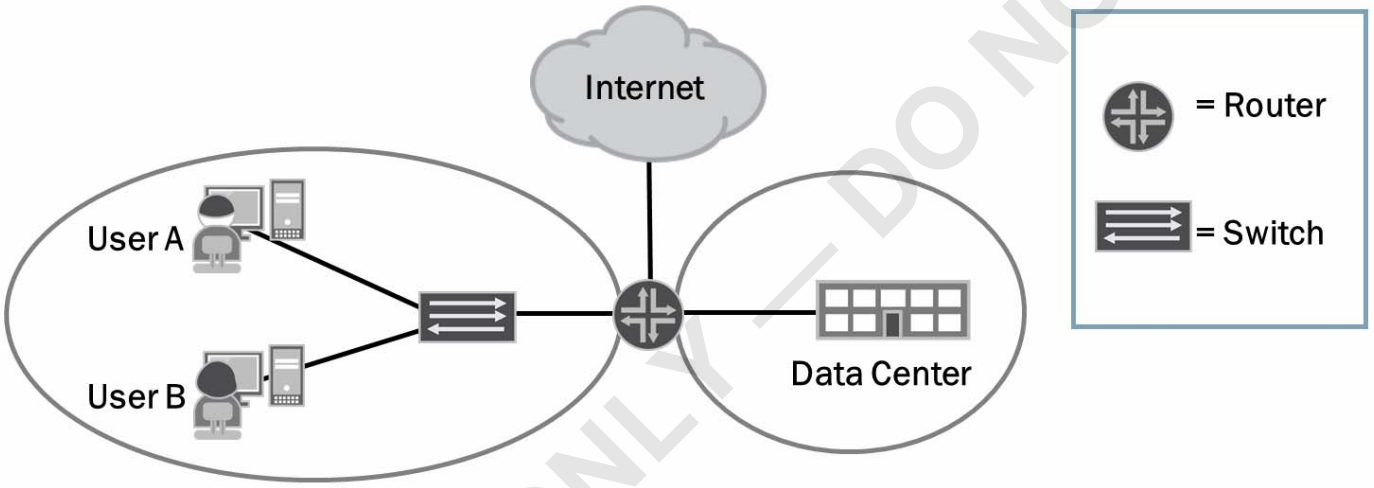
A Basic Definition of Routing

Routing, in its most basic form, is the process of moving data between Layer 3 networks. The sample topology on the slide consists of several Layer 3 networks, all connected to routers. Although routers are the most common devices for performing routing operations, note that many switches and security devices also perform routing operations. Note also that the Internet is actually a collection of many networks rather than a single network.

We look at the required components of routing and how devices running the Junos operating system make routing decisions on subsequent slides within this section.

Components of Routing

- For a device to communicate with another device in a remote network, the following requirements exist:
  - End-to-end communications path
  - Routing information on participating Layer 3 devices



Routing Components

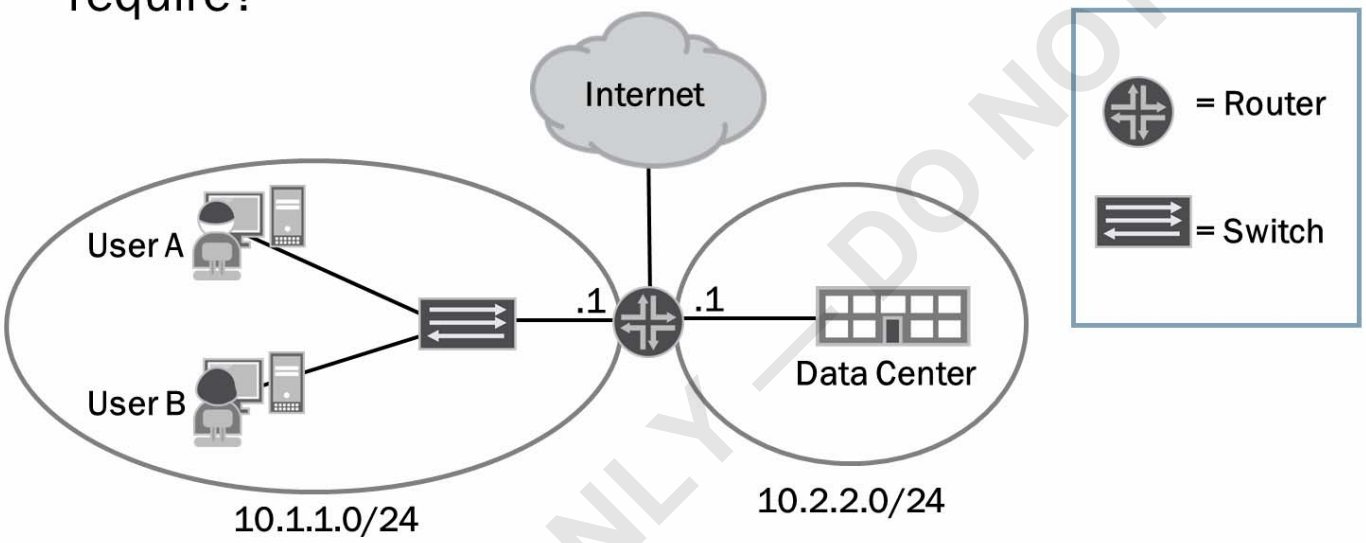
You must consider several components and other aspects to effectively implement routing between remote networks. However, you can classify the various components and considerations into two primary requirements—having an end-to-end communications path and ensuring all Layer 3 devices within the communications path have the required routing information.

In the example shown on the slide, you can see that a physical path exists between the highlighted networks and the Internet. As long as the physical path is configured and functioning correctly, the first requirement is satisfied.

For the second requirement, all Layer 3 devices participating in the communications path must have the necessary routing information. The devices within the user and data center networks must have the proper gateway configured (the router that connects to those networks as well as to the Internet). The gateway device must determine the proper next hop for each destination prefix for transit traffic it receives. Devices running the Junos OS use the forwarding table, which is a subset of information found in the route table, to make this determination. We discuss the route and forwarding tables in the next section.

Test Your Knowledge

- If User A needs to communicate with a device within the data center, what routing information do the Layer 3 devices within the communications path require?

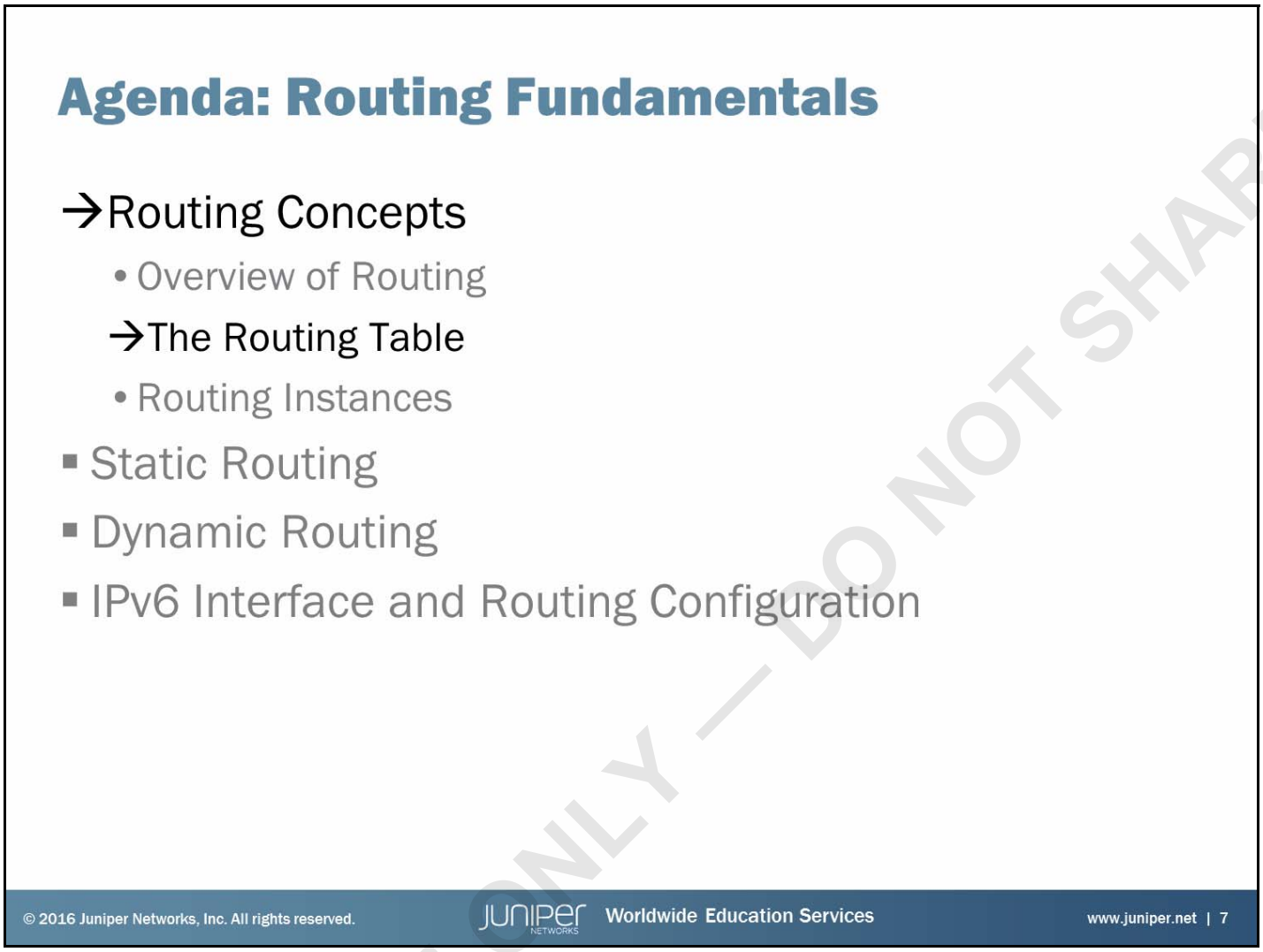


Test Your Knowledge

The slide presents a simple routing scenario and asks what routing information is required for User A to communicate with a device in the data center network.

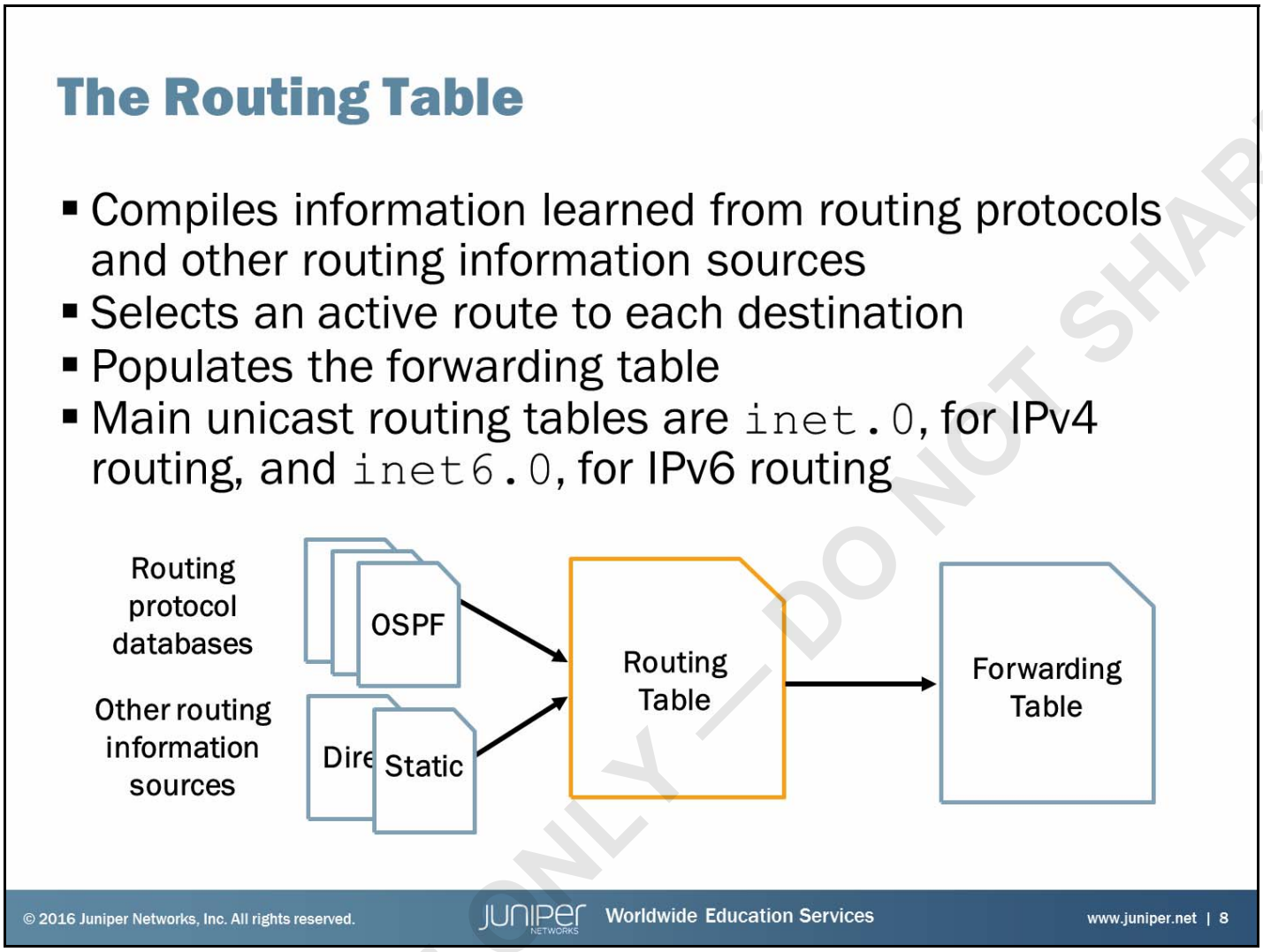
For any device to communicate with another device outside its directly connected subnet, a properly configured gateway is required. In the scenario illustrated on the slide, the device associated with User A must have its gateway set to the router's IP address (10.1.1.1). Likewise, the devices within the data center network need a properly configured gateway (10.2.2.1).

The router, which functions as the gateway device for the user and data center networks, requires sufficient routing information to determine the proper next hop for the traffic sent between the connected networks. In this example, the router learns the required information by way of the interface configuration. The router adds the networks, in which the interfaces are participating, to the route and forwarding tables. The router consults its forwarding table to determine the actual next hop for received traffic.



Routing Concepts: The Routing Table

The slide highlights the topic we discuss next.



Routing Information Sources

The Junos OS routing table consolidates prefixes from multiple routing information sources including various routing protocols, static routes, and directly connected routes.

Active Route Selection

When a device running the Junos OS receives multiple routes for a given prefix, it selects a single route as the active route. With additional configuration, the Junos OS supports multiple, equal-cost routes.

Forwarding Table

The router uses the active route for each destination prefix to populate the forwarding table. The forwarding table determines the outgoing interface and Layer 2 rewrite information for each packet forwarded by a device running the Junos OS.

Continued on the next page.

Multiple Routing Tables

Devices running the Junos OS can accommodate multiple routing tables. The primary routing table, `inet.0`, stores IPv4 unicast routes. Additional predefined routing tables exist, such as `inet6.0`, which the Junos OS creates when the configuration requires it. An administrator can create custom routing tables to be used in addition to these routing tables.

The following is a summary of the common predefined routing tables you might see on a device running the Junos OS:

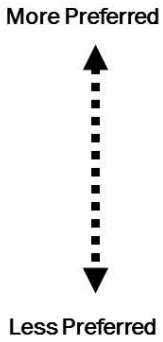
- `inet.0`: Used for IPv4 unicast routes;
- `inet.1`: Used for the multicast forwarding cache;
- `inet.2`: Used for Multicast Border Gateway Protocol (MBGP) routes to provide reverse path forwarding (RPF) checks;
- `inet.3`: Used for MPLS path information;
- `inet.4`: Used for Multicast Source Discovery Protocol (MSDP) route entries;
- `inet6.0`: Used for IPv6 unicast routes; and
- `mpls.0`: Used for MPLS next hops.

Route Preference

- Ranks routes received from different sources
- Primary criterion for selecting the active route
  - Used as a tiebreaker when the same destination prefix is available through multiple sources

Route Preference Values

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



Preferred Routing Information Sources

The Junos OS uses route preference to differentiate routes received from different routing protocols or routing information sources. Route preference is equivalent to administrative distance on equipment from other vendors.

Selecting the Active Route

The Junos OS uses route preference to rank routes received through the various route information sources and as the primary criterion for selecting the active route.

The following table shows the default preference values for a selected set of routing information sources. The complete list of default route preference assignments is shown on the following page.

Continued on the next page.

Selecting the Active Route (contd.)

Default Route Preferences

Direct	0	SNMP	50
Local	0	Router discovery	55
System routes 4	4	RIP	100
Static and Static LSPs	5	RIPng	100
RSVP-signaled LSPs	7	DVMRP	110
LDP-signaled LSPs	9	Aggregate	130
OSPF internal	10	OSPF AS external	150
IS-IS Level 1 internal	15	IS-IS Level 1 external	160
IS-IS Level 2 internal	18	IS-IS Level 2 external	165
Redirects	30	BGP (internal and external)	170
Kernel	40	MSDP	175

Routing preference values can range from 0 to 4,294,967,295. Lower preference values are preferred over higher preference values. The following command output demonstrates that a static route with a preference of five is preferred over an OSPF internal route with a preference of ten:

```
user@router> show route 192.168.36.1 exact

inet.0: 5 destinations, 6 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.36.1/32    *[Static/5] 00:00:31
                    > to 10.1.1.2 via ge-0/0/10.0
                    [OSPF/10] 00:02:21, metric 1
                    > to 10.1.1.2 via ge-0/0/10.0
```

You can modify the default preference value for most routing information sources to make them more or less desirable. The exception is with direct and local routes, which are always preferred regardless of the modified route preference value associated with other routing information sources.

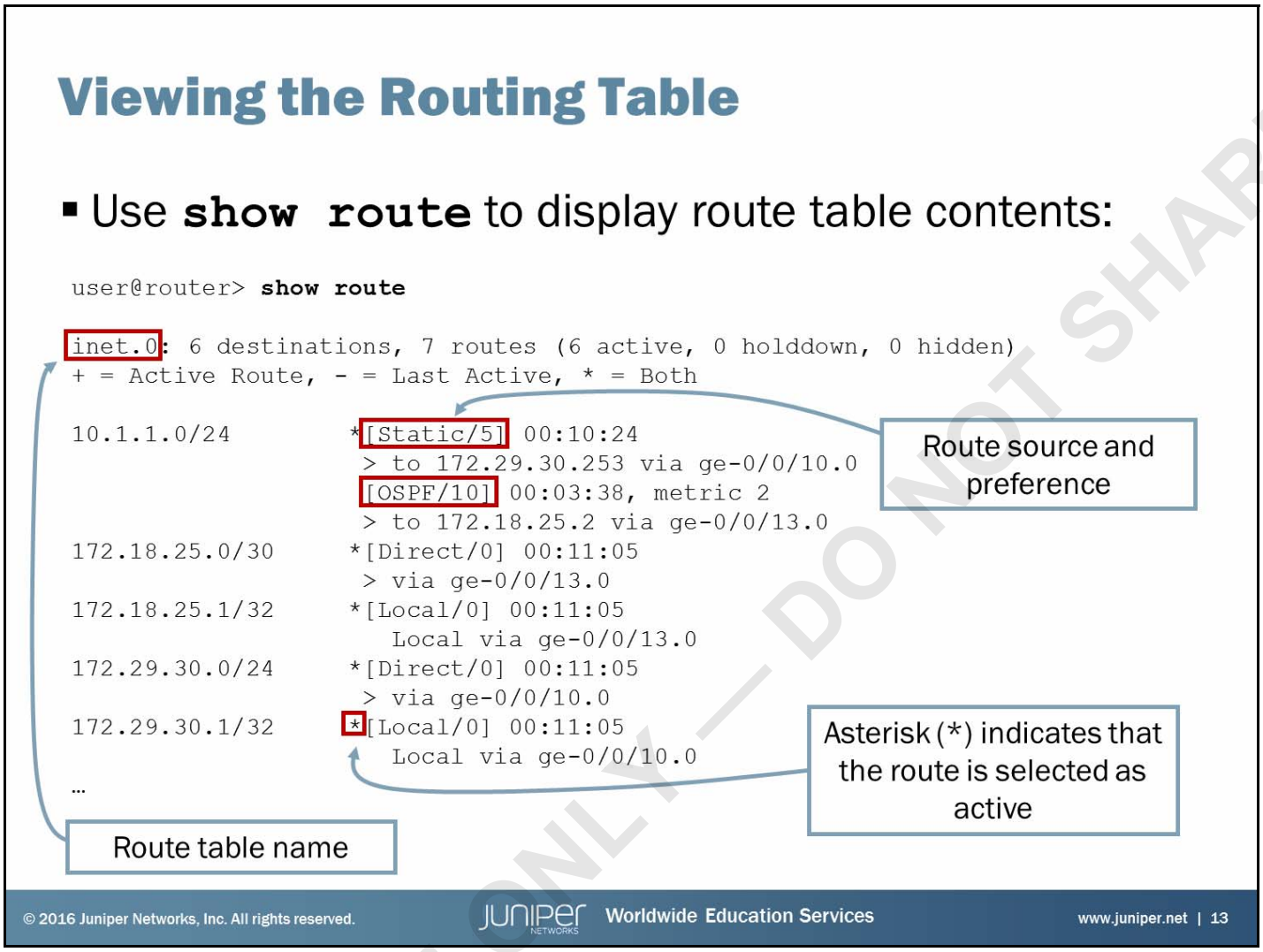
Continued on the next page.

Selecting the Active Route (contd.)

If equal-cost paths exist for the same destination, the routing protocol daemon (rpd) randomly selects one of the available paths. This approach provides load distribution among the paths while maintaining packet ordering per destination. The following output illustrates this point:

```
user@router> show route 10.1.0.0/16

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

10.1.1.0/24        *[Static/5] 00:00:25
                      to 172.20.66.2 via ge-0/0/2.0
                    > to 172.20.77.2 via ge-0/0/3.0
10.1.2.0/24        *[Static/5] 00:00:25
                    > to 172.20.66.2 via ge-0/0/2.0
                      to 172.20.77.2 via ge-0/0/3.0
10.1.3.0/24        *[Static/5] 00:00:25
                      to 172.20.66.2 via ge-0/0/2.0
                    > to 172.20.77.2 via ge-0/0/3.0
10.1.4.0/24        *[Static/5] 00:00:25
                    > to 172.20.66.2 via ge-0/0/2.0
                      to 172.20.77.2 via ge-0/0/3.0
```

If desired, you can enable per-flow load balancing over multiple equal-cost paths through routing policy. Load balancing is outside the scope of this class.



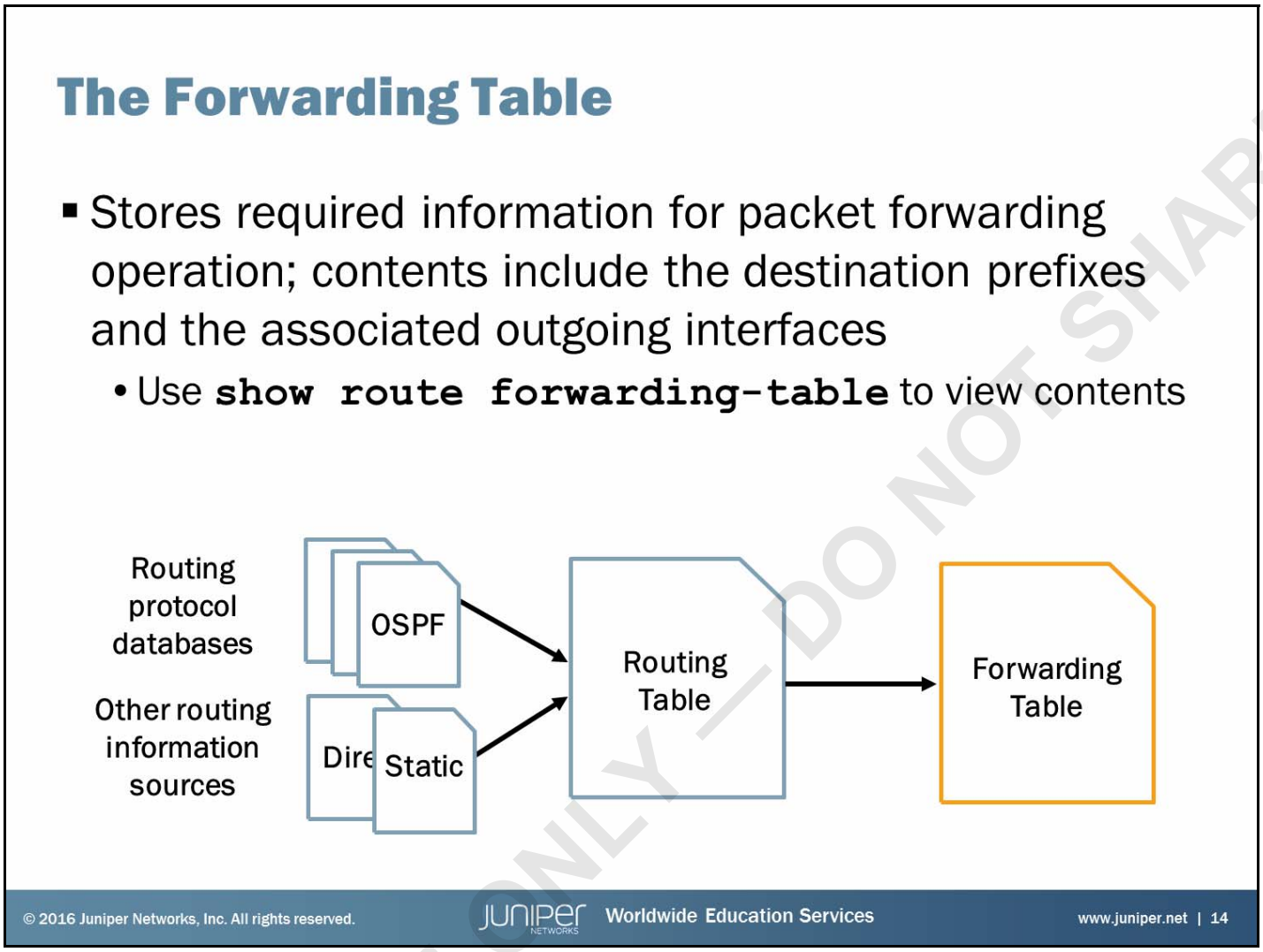
Viewing the Route Table

The slide shows the use of the **show route** command, which displays all route entries in the routing table. As identified on the slide, all active routes are marked with an asterisk (*) next to the selected entry. Each route entry displays the source from which the device learned the route, along with the route preference for that source.

The **show route** command displays a summary of active, holddown, and hidden routes. Active routes are the routes the system uses to forward traffic. Holddown routes are routes that are in a pending state before the system declares them as inactive. Hidden routes are routes that the system cannot use for reasons such as an invalid next hop and route policy.

You can filter the generated output by destination prefix, protocol type, and other distinguishing attributes. The following sample capture illustrates the use of the protocol filtering option:

```
user@router> show route protocol ospf

inet.0: 6 destinations, 7 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.1.1.0/24          [OSPF/10] 04:57:41, metric 2
                     > to 172.18.25.2 via ge-0/0/13.0
224.0.0.5/32        *[OSPF/10] 05:00:58, metric 1
                       MultiRecv
```



The Forwarding Table

The forwarding table stores a subset of information from the routing table. Within the forwarding table, you can find the details used by a device running the Junos OS to forward packets such as the learned destination prefixes and the outgoing interfaces associated with each destination prefix.

You use the **show route forwarding-table** CLI command to view the forwarding table contents:

```
user@router> show route forwarding-table
Routing table: inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            user     0 0:17:cb:4e:ae:81   ucst   520     3 ge-0/0/0.0
default            perm     0                    rjct    36     1
0.0.0.0/32         perm     0                    dscd    34     1
172.19.0.0/16      user     0 200.1.4.100        ucst   535     3 ge-0/0/3.0
172.19.52.0/24     user     0 200.1.2.100        ucst   529     3 ge-0/0/1.0
172.19.52.16/28    user     0 200.1.3.100        ucst   534     3 ge-0/0/2.0
...
```

Continued on the next page.

The Forwarding Table (contd.)

Note that the Junos kernel adds some forwarding entries and considers them permanent in nature. One such example is the `default` forwarding entry, which matches all packets when no other matching entry exists. When a packet matches this `default` forwarding entry, the router discards the packet and it sends an Internet Control Message Protocol (ICMP) destination unreachable message back to the sender. If you configured a user-defined default route, the router uses it instead of the permanent `default` forwarding entry.

The following list displays some common route types associated with forwarding entries:

- `dest`: Remote addresses directly reachable through an interface;
- `intf`: Installed as a result of configuring an interface;
- `perm`: Routes installed by the kernel when the routing table initializes; and
- `user`: Routes installed by the routing protocol process or as a result of the configuration.

The following list displays some common next-hop types associated with forwarding entries:

- `bcst`: Broadcast;
- `dscd`: Discard silently without sending an ICMP unreachable message;
- `hold`: Next hop is waiting to be resolved into a unicast or multicast type;
- `locl`: The local address on an interface;
- `mcst`: Wire multicast next hop (limited to the LAN);
- `mdsc`: Multicast discard;
- `recv`: Receive;
- `rjct`: Discard and send an ICMP unreachable message;
- `ucst`: Unicast; and
- `ulst`: A list of unicast next hops used when you configure load balancing.

Determining the Next Hop

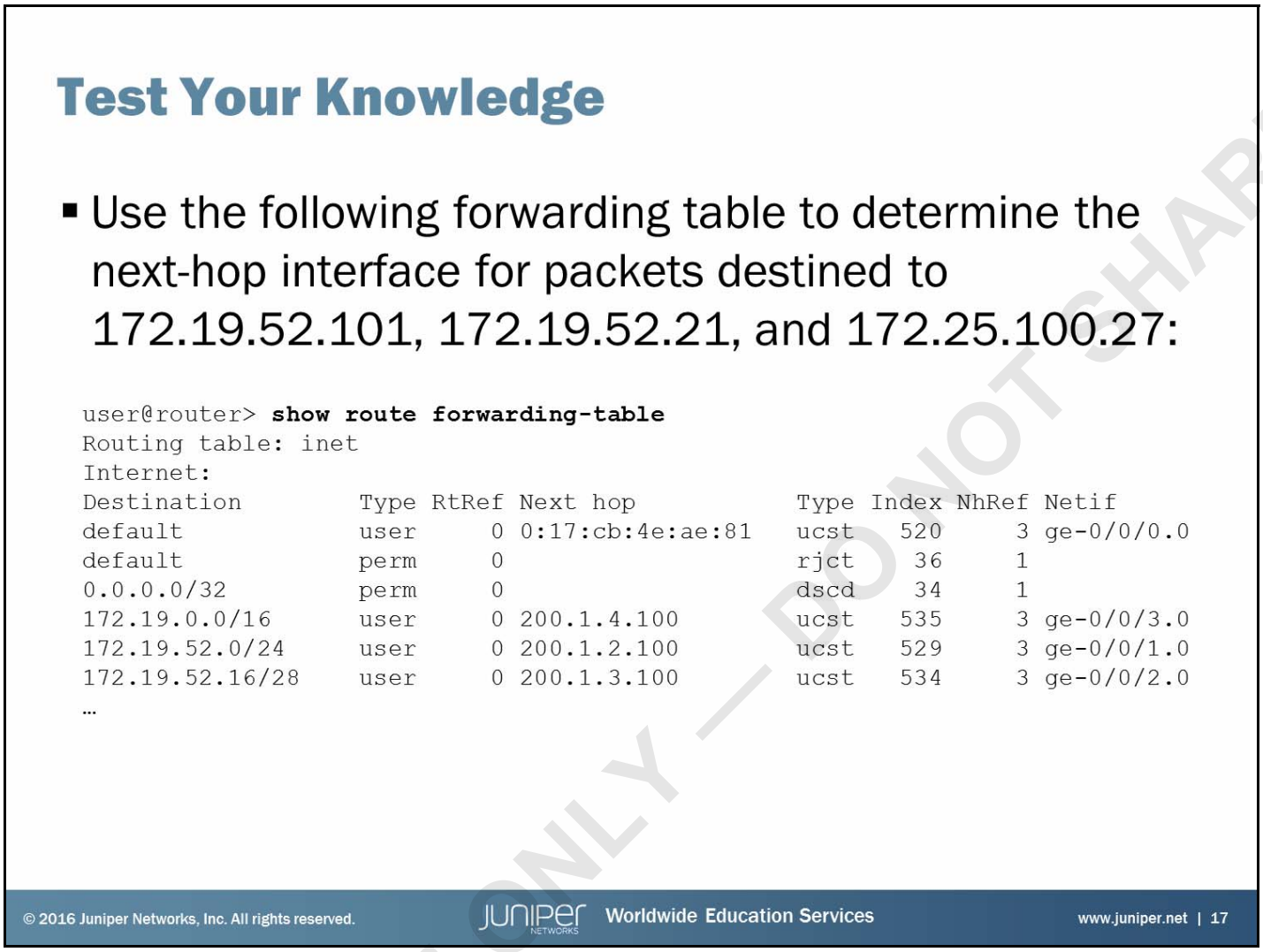
- Device compares incoming packets against forwarding table entries to determine the appropriate next hop
  - If multiple matches exist, it uses the most specific entry (longest match) to forward packet toward the destination
  - If no matching entry exists, it sends a destination unreachable notification back to the source device

Determining the Next Hop

When a packet enters a device running the Junos OS, it compares that packet against the entries within the forwarding table to determine the proper next hop. If the packet is destined to the local device, the Junos OS processes the packet locally. If the packet is destined to a remote device and a valid entry exists, the device running the Junos OS forwards the packet out the next-hop interface associated with the forwarding table entry.

If multiple destination prefixes match the packet's destination, the Junos OS uses the most specific entry (also called longest match) when forwarding the packet to its destination.

In situations where no matching entry exists, the device running the Junos OS responds to the source device with a destination unreachable notification.



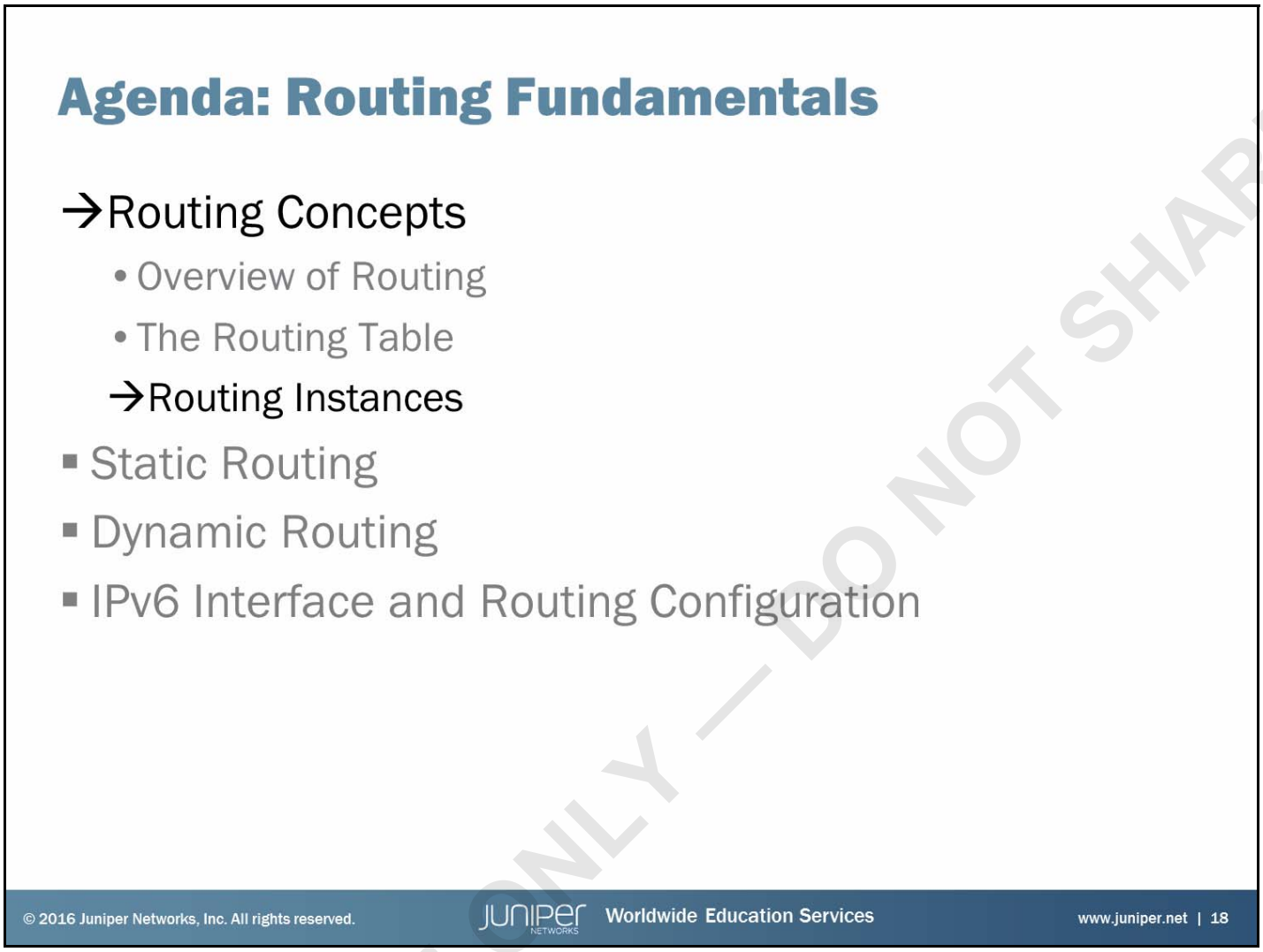
Test Your Knowledge

The slide displays a sample forwarding table and tests your understanding of how next-hop interfaces are determined. Keep in mind that although multiple entries might match a destination, the device uses the most specific (longest match) entry when determining a packet's next-hop interface.

The most specific forwarding entry matching packets destined to 172.19.52.101 is the 172.19.52.0/24 destination prefix. The next hop associated with this destination prefix is ge-0/0/1.0.

The most specific forwarding entry matching packets destined to 172.19.52.21 is the 172.19.52.16/28 destination prefix. The next hop associated with this destination prefix is ge-0/0/2.0.

The only forwarding entry matching packets destined to 172.25.100.27 is the user-defined default forwarding entry. The next hop associated with the user-defined default forwarding entry is ge-0/0/0.0.

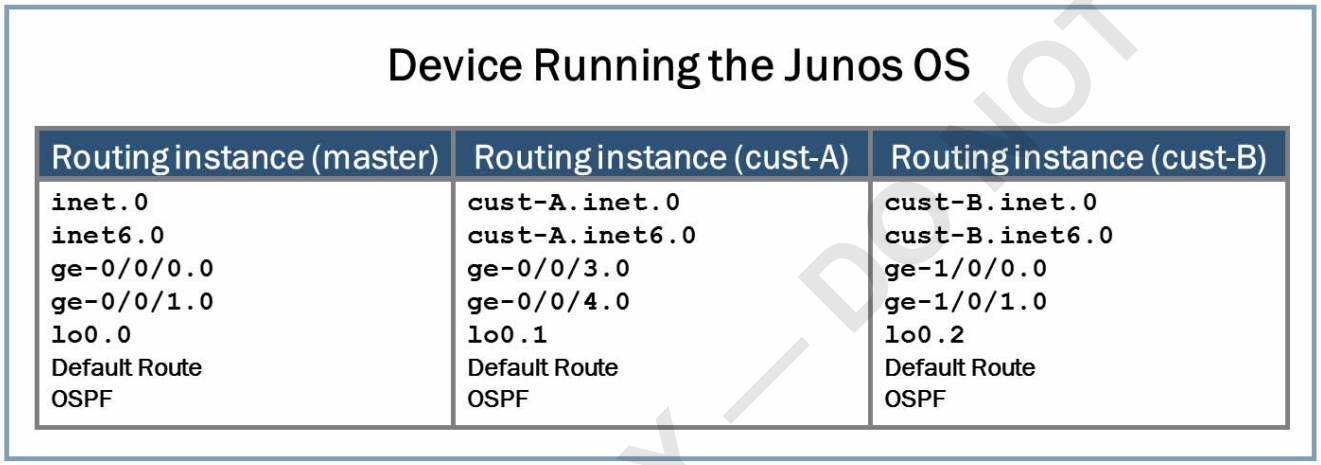


Routing Concepts: Routing Instances

The slide highlights the topic we discuss next.

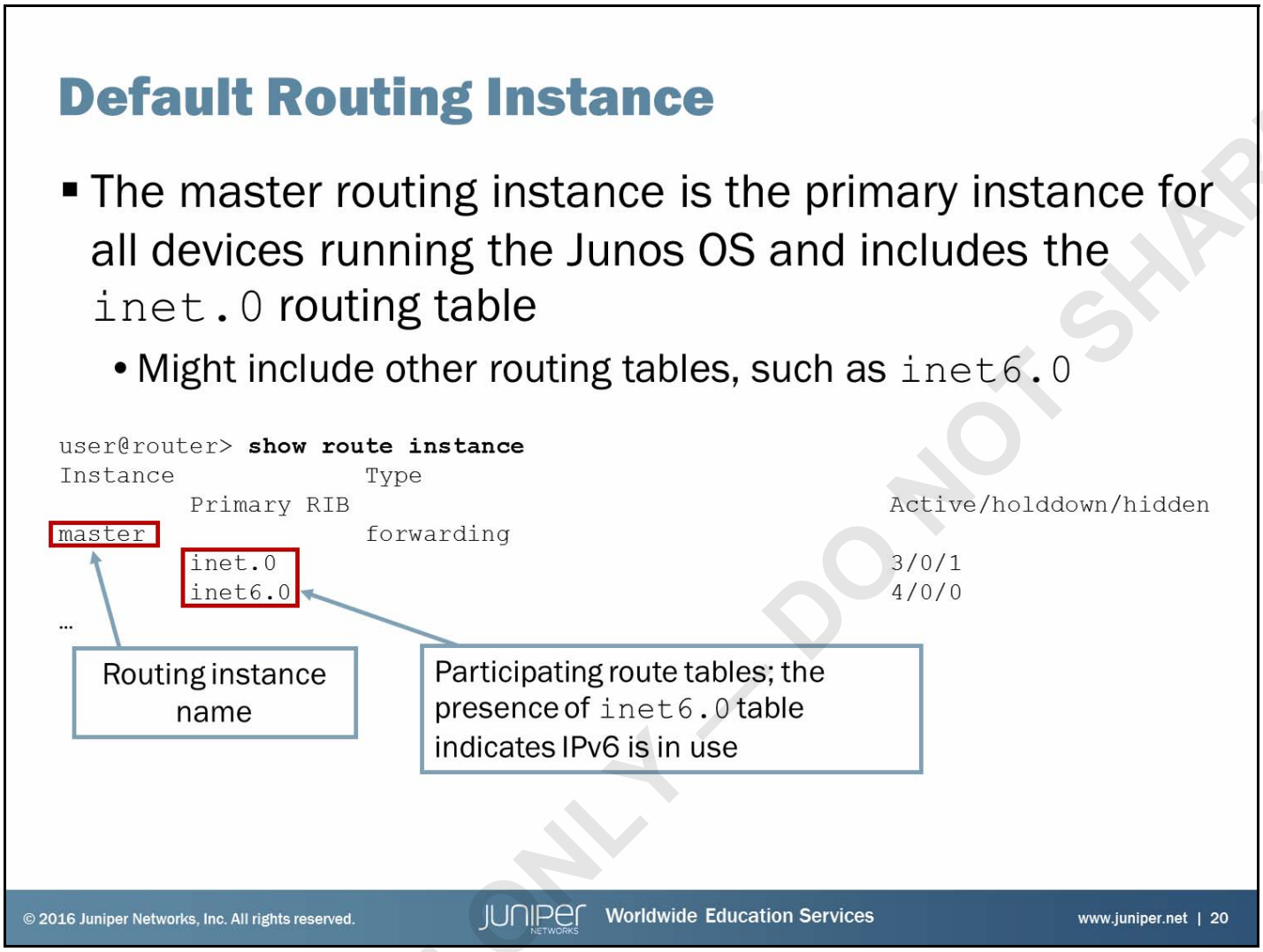
Overview of Routing Instances

- A routing instance is a unique collection of routing tables, interfaces, and routing protocol parameters



Overview of Routing Instances

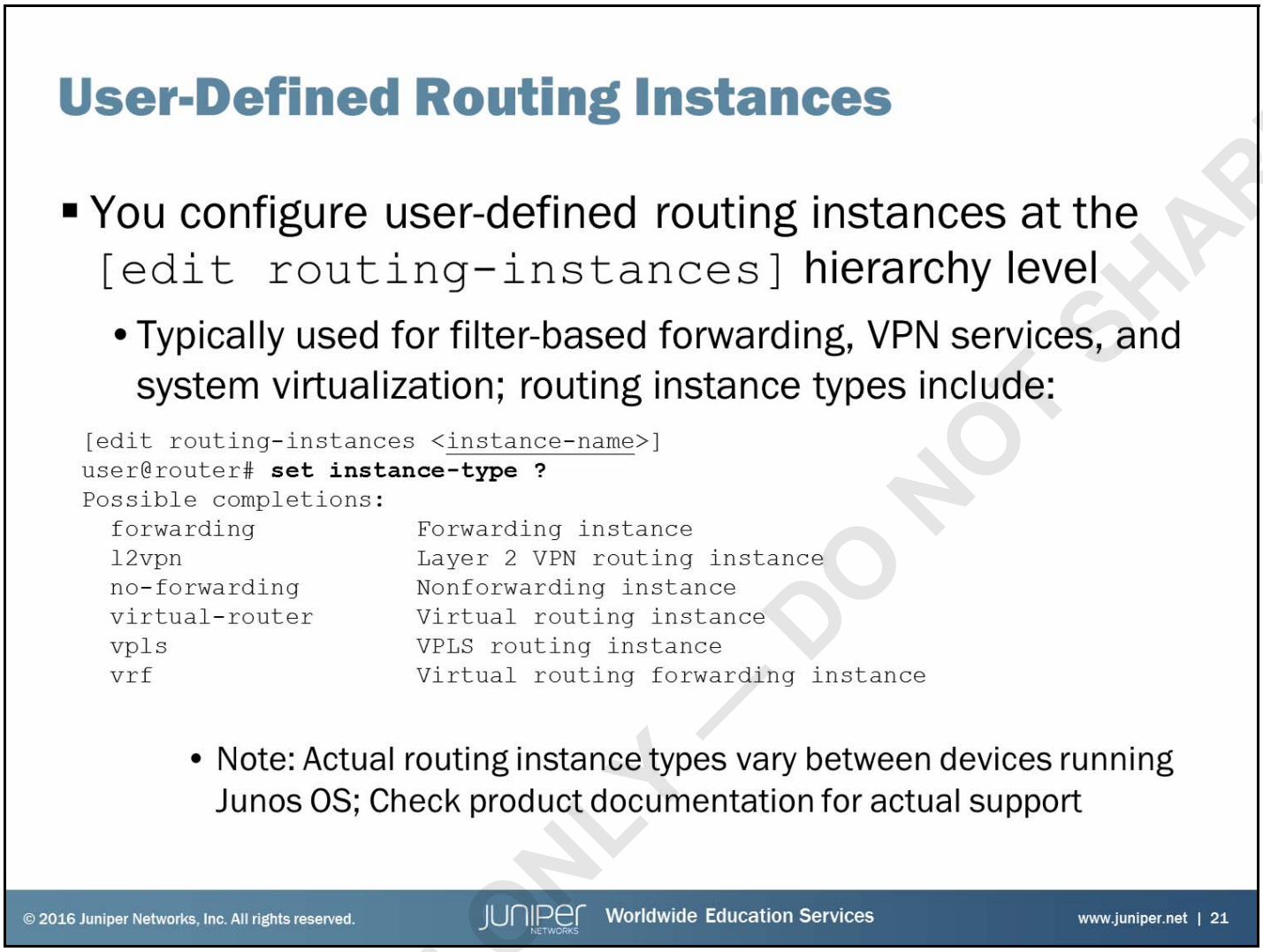
The Junos OS logically groups routing tables, interfaces, and routing protocol parameters to form unique routing instances. The device logically keeps the routing information in one routing instance apart from all other routing instances. The use of routing instances introduces great flexibility because a single device can effectively imitate multiple devices.



Master Routing Instance

The Junos OS creates a default unicast routing instance called the `master` routing instance. By default, the `master` routing instance includes the `inet.0` routing table, which the device uses for IPv4 unicast routing. The software creates other routing tables, such as `inet6.0`, adds them to their respective routing instance, and displays them when required by the configuration. The Junos OS also creates private routing instances, which the device uses for internal communications between hardware components. You can safely ignore these instances and their related information when planning your network. The following sample output shows all default routing instances:

```
user@router> show route instance
Instance             Type
        Primary RIB                                 Active/holddown/hidden
__juniper_private1__ forwarding
        __juniper_private1__.inet.0                 2/0/2
        __juniper_private1__.inet6.0                1/0/0

__juniper_private2__ forwarding
        __juniper_private2__.inet.0                 0/0/1

__master.anon__      forwarding

master               forwarding
        inet.0                                      7/0/0
```



User-Defined Routing Instances

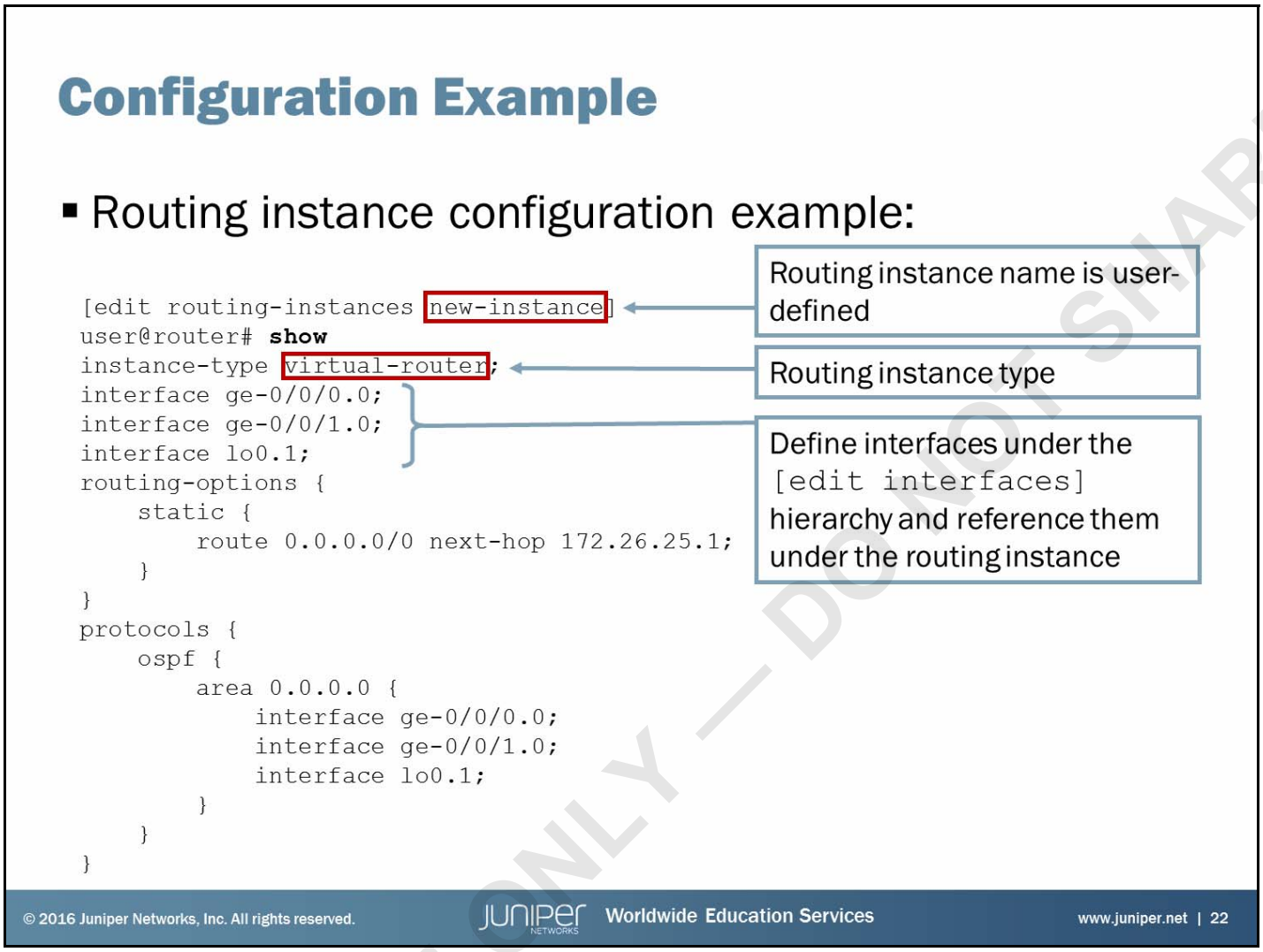
For added flexibility, the Junos OS allows you to configure additional routing instances under the `[edit routing-instances]` hierarchy. You can use user-defined routing instances for a variety of different situations, which provides you a great amount of flexibility in your environments.

Some typical uses of user-defined routing instances include filter-based forwarding (FBF), Layer 2 and Layer 3 VPN services, and system virtualization.

The following are some of the common routing instance types:

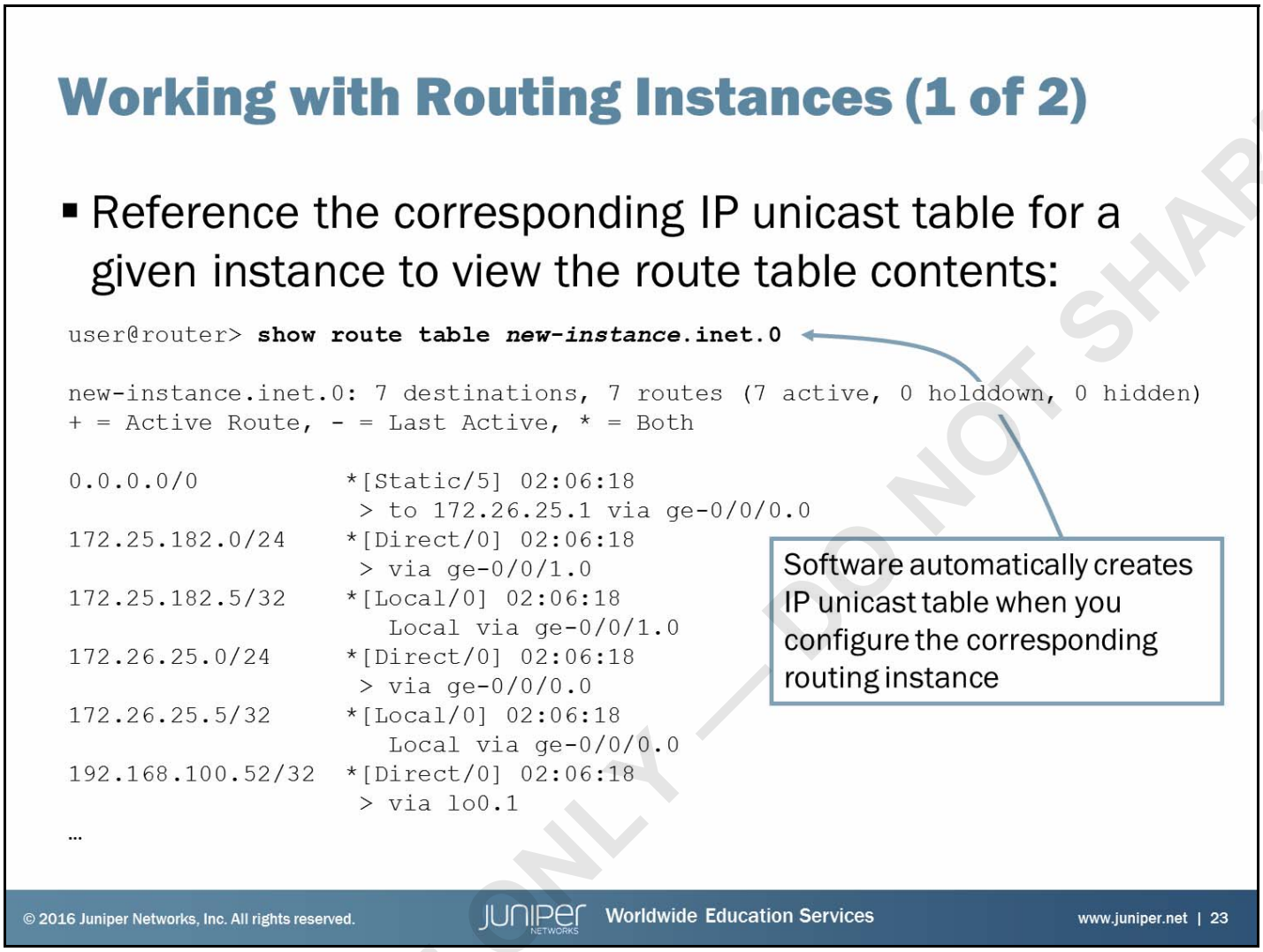
- `forwarding`: Used to implement filter-based forwarding for common Access Layer applications;
- `l2vpn`: Used in Layer 2 VPN implementations;
- `no-forwarding`: Used to separate large networks into smaller administrative entities;
- `virtual-router`: Used for non-VPN-related applications such as system virtualization;
- `vpls`: Used for point-to-multipoint LAN implementations between a set of sites in a VPN; and
- `vrf`: Used in Layer 3 VPN implementations.

Note that the actual routing instance types vary between platforms running the Junos OS. Be sure to check the technical documentation for your specific product.



Configuration Example: Routing Instances

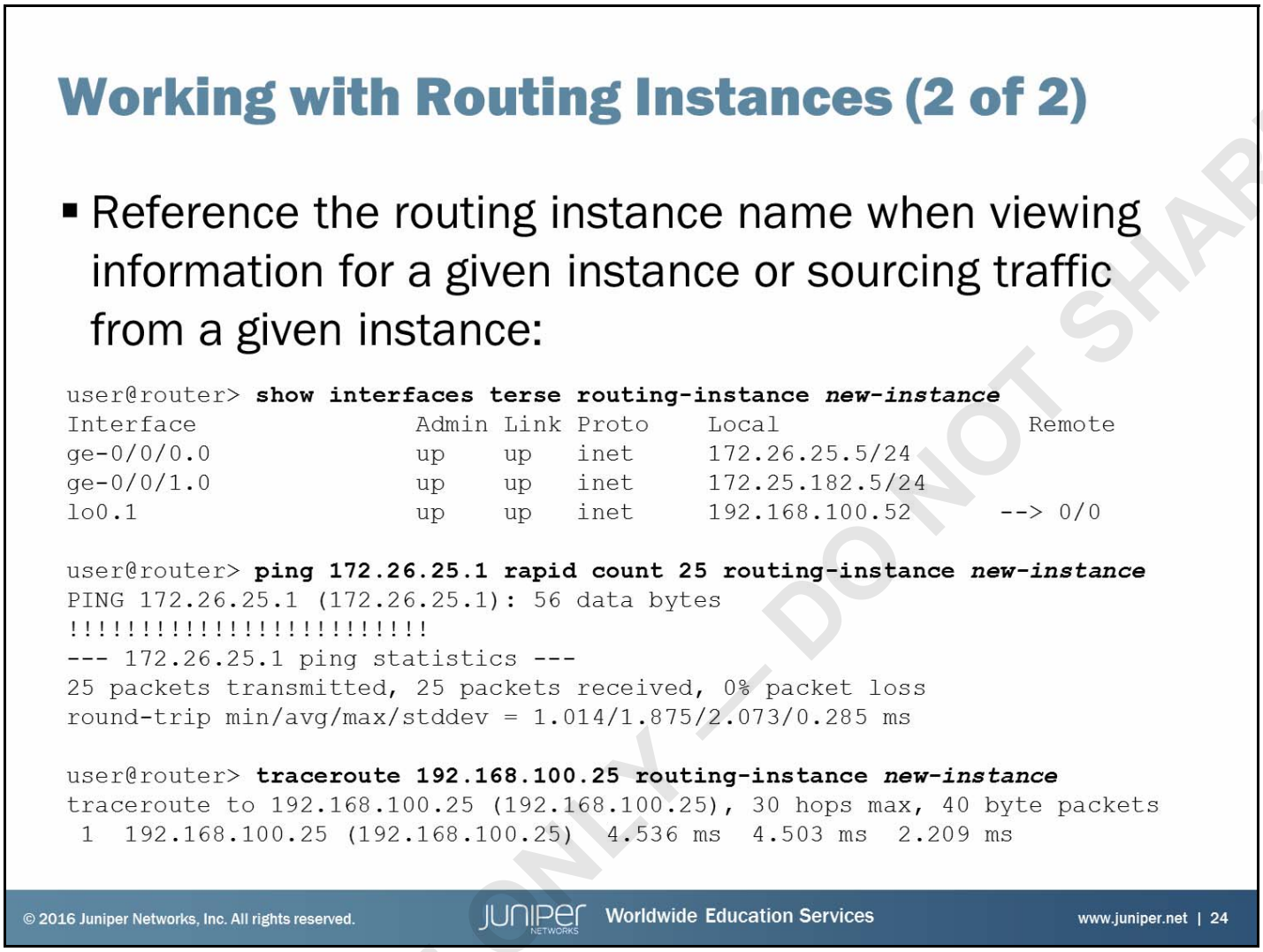
The slide illustrates a basic routing instance configuration example.



Working with Routing Instances: Part 1

Once you configure a routing instance and the device learns routing information within the instance, the Junos OS automatically generates a routing table. If you use IPv4 routing, the software creates an IPv4 unicast routing table. The name of the routing table uses the format *instance-name*.inet.0, where *instance-name* is the name of the routing instance within the configuration. Likewise, if you use IPv6 within the instance, the software creates an IPv6 unicast routing table and it follows the format *instance-name*.inet6.0.

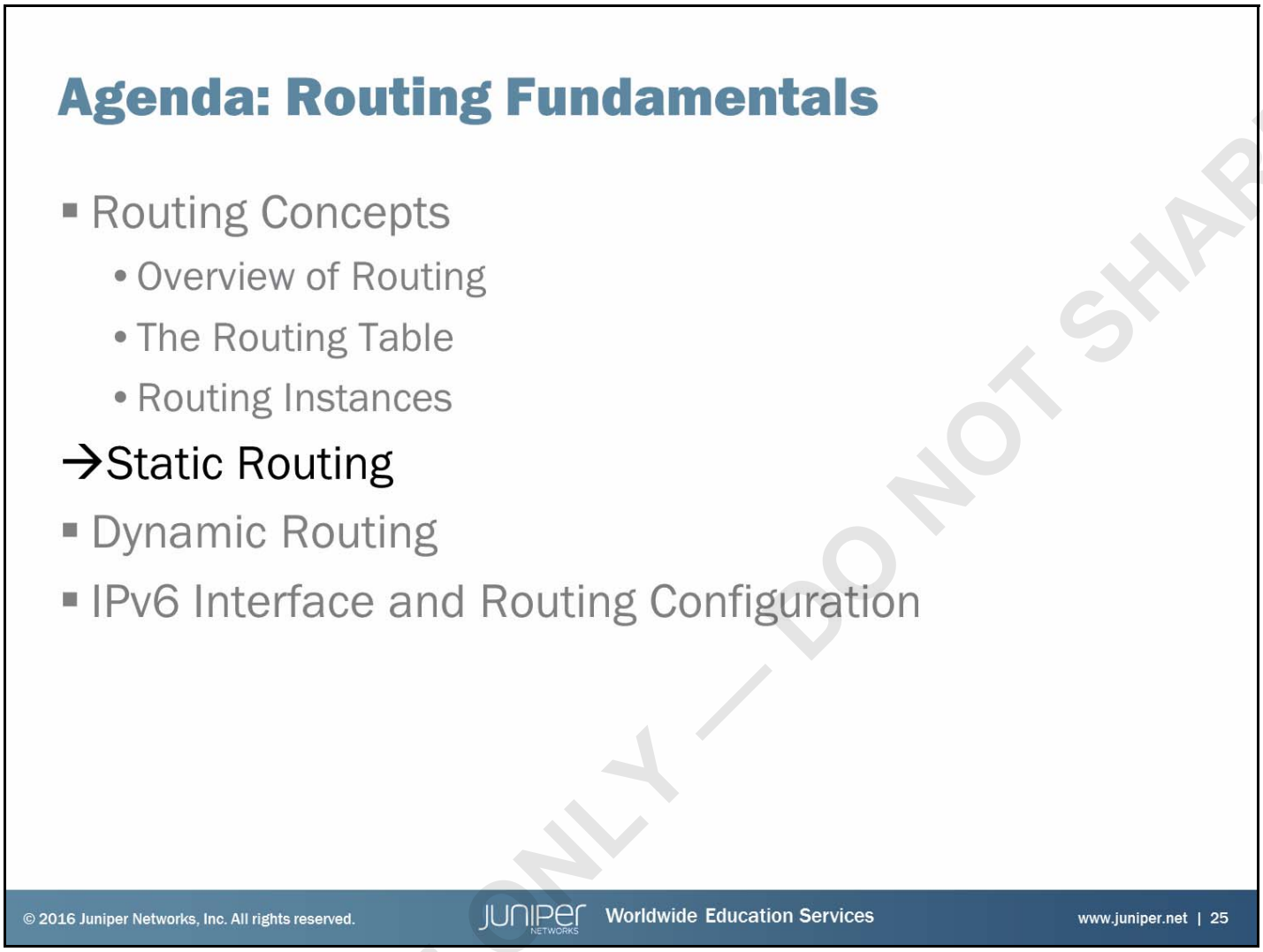
As illustrated on the slide, to view a routing table associated with a specific routing instance, you simply use the **show route table *table-name*** CLI command.



Working with Routing Instances: Part 2

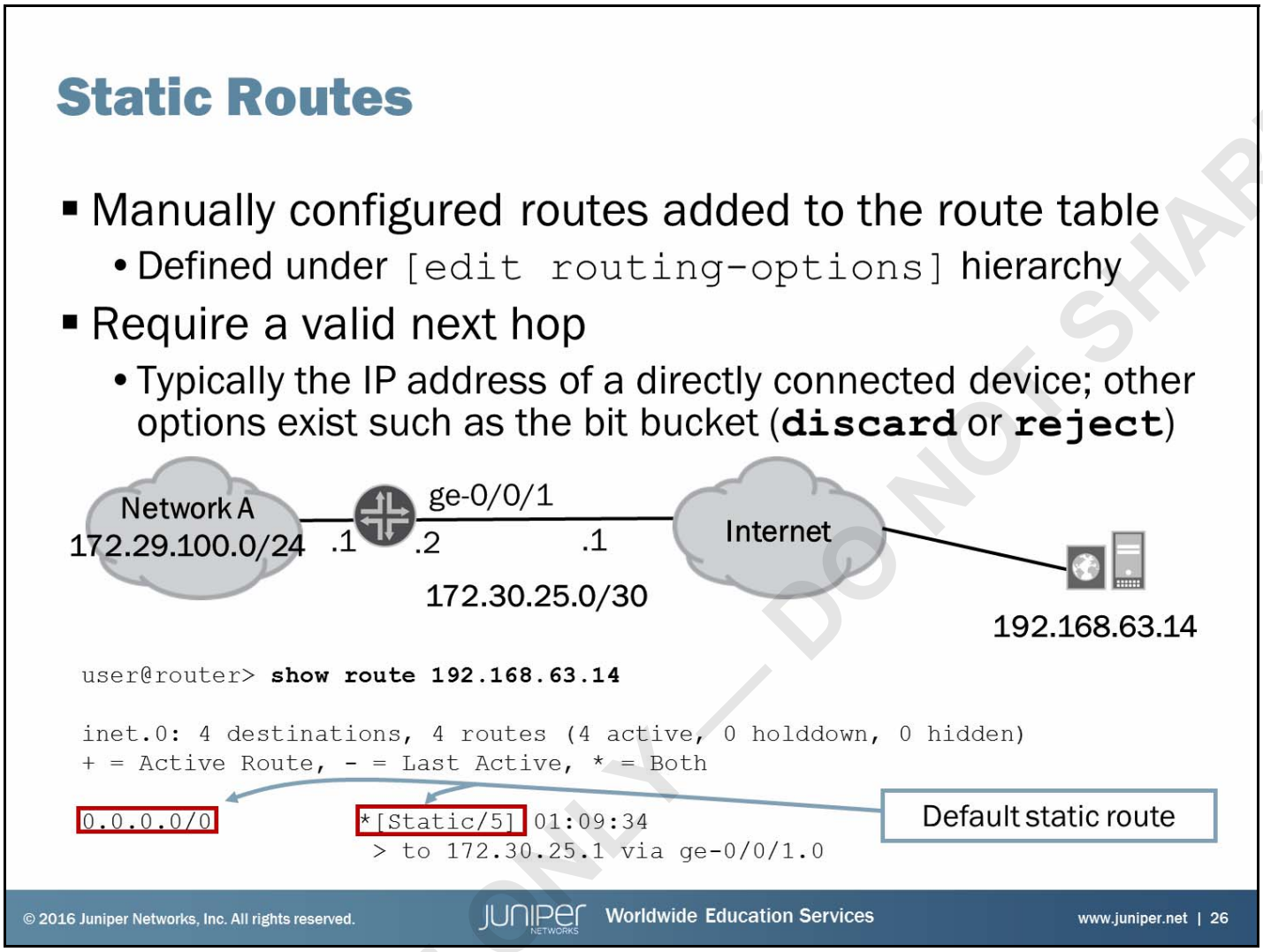
You can filter many of the common outputs generated through CLI **show** commands by referencing the name of a given routing instance. The first example on the slide shows a practical way of viewing interfaces that belong to a specific routing instance.

You can also source traffic from a specific routing instance by referencing the name of the desired routing instance. The last two examples on the slide show this option in action with the `ping` and `traceroute` utilities.



Static Routing

The slide highlights the topic we discuss next.



Static Routes

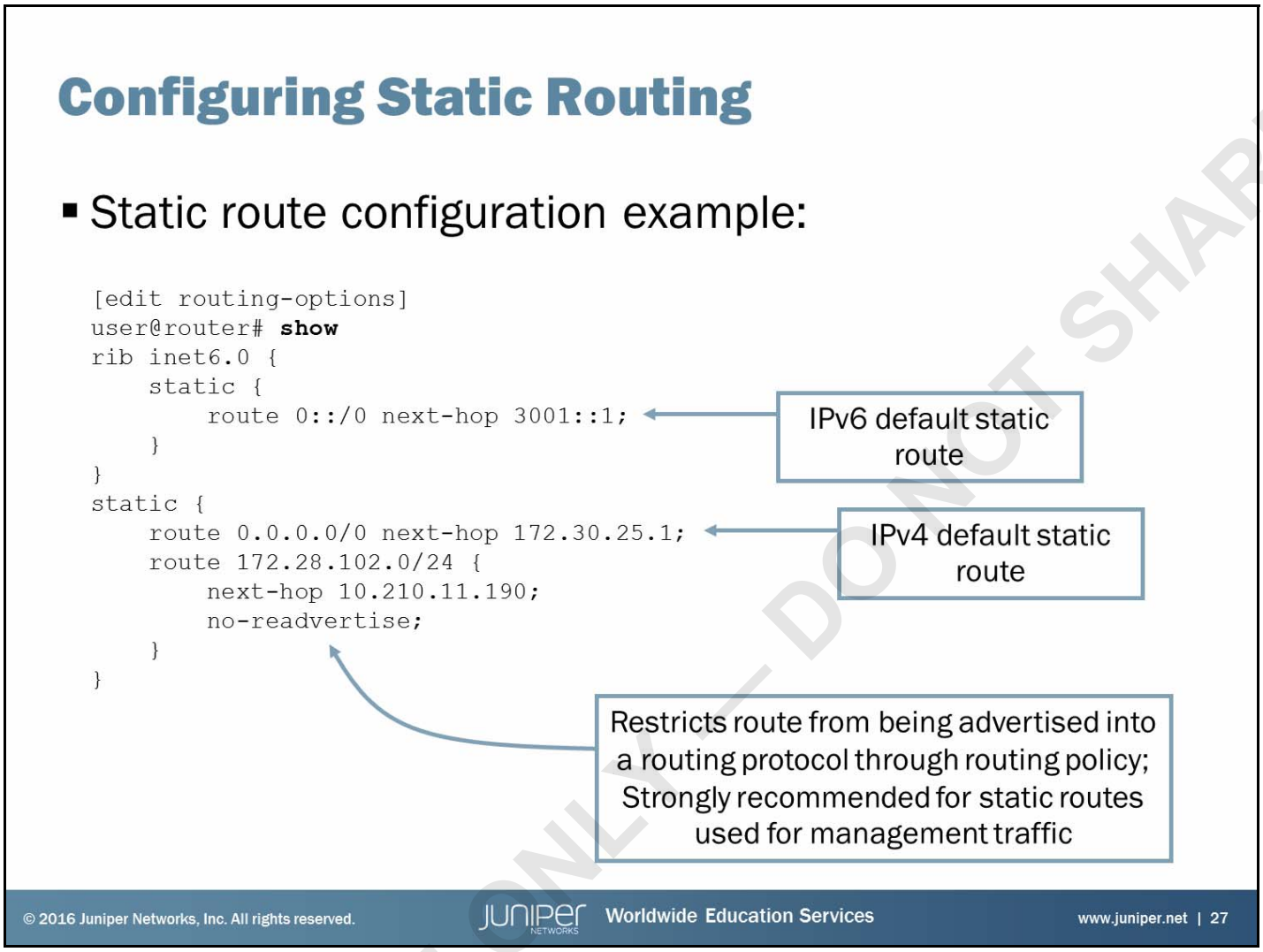
Static routes are used in a networking environment for multiple purposes, including a default route for the autonomous system (AS) and as routes to customer networks. Unlike dynamic routing protocols, you manually configure the routing information provided by static routes on each router or multilayer switch in the network. All configuration for static routes occurs at the `[edit routing-options]` level of the hierarchy.

Next Hop Required

Static routes must have a valid next-hop defined. Often that next-hop value is the IP address of the neighboring router headed toward the ultimate destination. On point-to-point interfaces, you can specify the egress interface name rather than the IP address of the remote device. Another possibility is that the next-hop value is the *bit bucket*. This phrase is analogous to dropping the packet off the network. Within the Junos OS, the way to represent the dropping of packets is with the keywords **`reject`** or **`discard`**. Both options drop the packet from the network. The difference between them is in the action the device running the Junos OS takes after the drop action. If you specify **`reject`** as the next-hop value, the system sends an ICMP message (the network unreachable message) back to the source of the IP packet. If you specify **`discard`** as the next-hop value, the system does not send back an ICMP message; the system drops the packet silently.

By default, the next-hop IP address of static routes configured in the Junos OS must be reachable using a direct route. Unlike routing software from other vendors, the Junos OS does not perform recursive lookups of next hops by default.

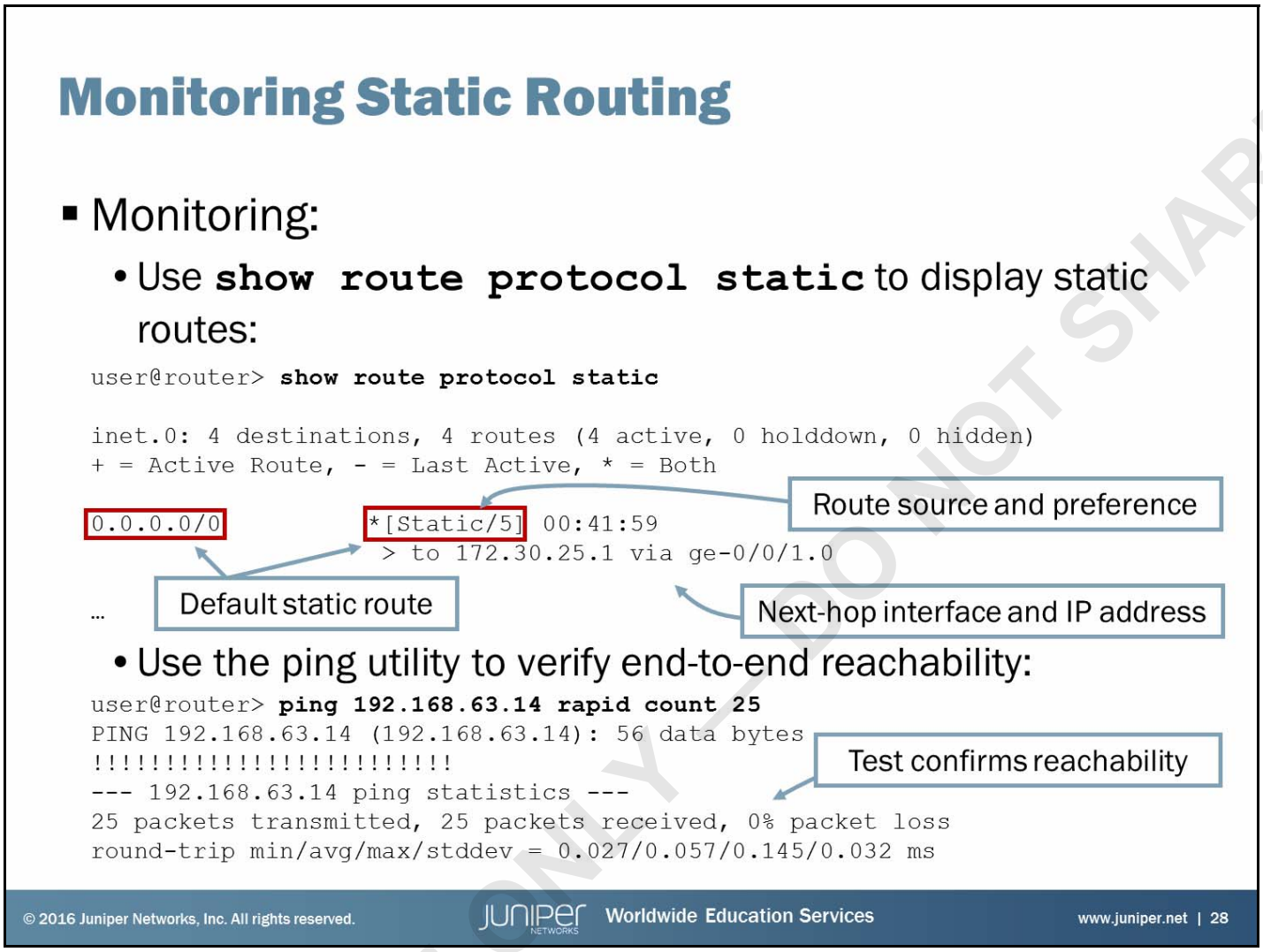
Static routes remain in the routing table until you remove them or until they become inactive. One possible scenario in which a static route becomes inactive is when the IP address used as the next hop becomes unreachable.



Configuration Example: Static Routing

The slide illustrates the basic configuration syntax for IPv4 and IPv6 static routes. The slide also highlights the `no-readvertise` option, which prohibits the redistribution of the associated route through routing policy into a dynamic routing protocol such as OSPF. We strongly recommend that you use the `no-readvertise` option on static routes that direct traffic out the management Ethernet interface and through the management network.

Note that IPv6 support varies between Junos devices. Be sure to check the technical documentation for your specific product for support information.

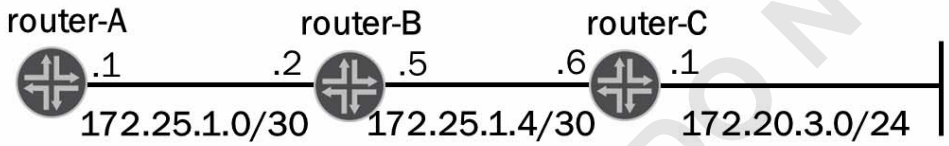


Monitoring Static Routing

The slide shows the basic verification steps when determining the proper operation of static routing.

Next-Hop Resolution

- Resolving indirect next hops:
  - By default, the software can resolve only directly connected next hops
  - Use the **resolve** option to allow resolution of indirectly connected next hops:




```
[edit routing-options]
user@router-A# show
static {
    route 172.20.3.0/24 {
        next-hop 172.25.1.6;
        resolve;
    }
}
```

Indirect next hop

resolve option required

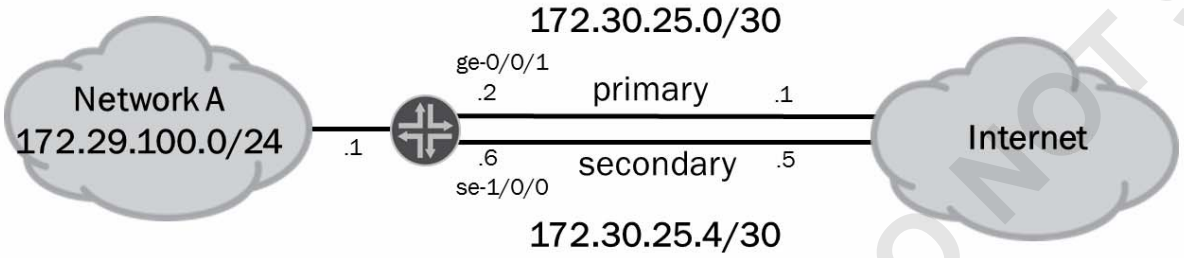
Resolving Indirect Next Hops

By default, the Junos OS requires that the next-hop IP address of static routes be reachable using a direct route. Unlike software from other vendors, the Junos OS does not perform recursive lookups of next hops by default.

As illustrated on the slide, you can alter the default next-hop resolution behavior using the `resolve` CLI option. In addition to the `resolve` CLI option, a route to the indirect next hop is also required. Indirect next hops can be resolved through another static route or through a dynamic routing protocol. We recommend, whenever possible, that you use a dynamic routing protocol as your method of resolution. Using a dynamic routing protocol, rather than a static route to resolve indirect next hops, dynamically removes the static route if the indirect next hop becomes unavailable.

Qualified Next Hops

- Use **qualified-next-hop** to allow independent preference for static routes to the same destination:




```
[edit routing-options]
user@router# show
static {
    route 0.0.0.0/0 {
        next-hop 172.30.25.1;
        qualified-next-hop 172.30.25.5 {
            preference 7;
        }
    }
}
```

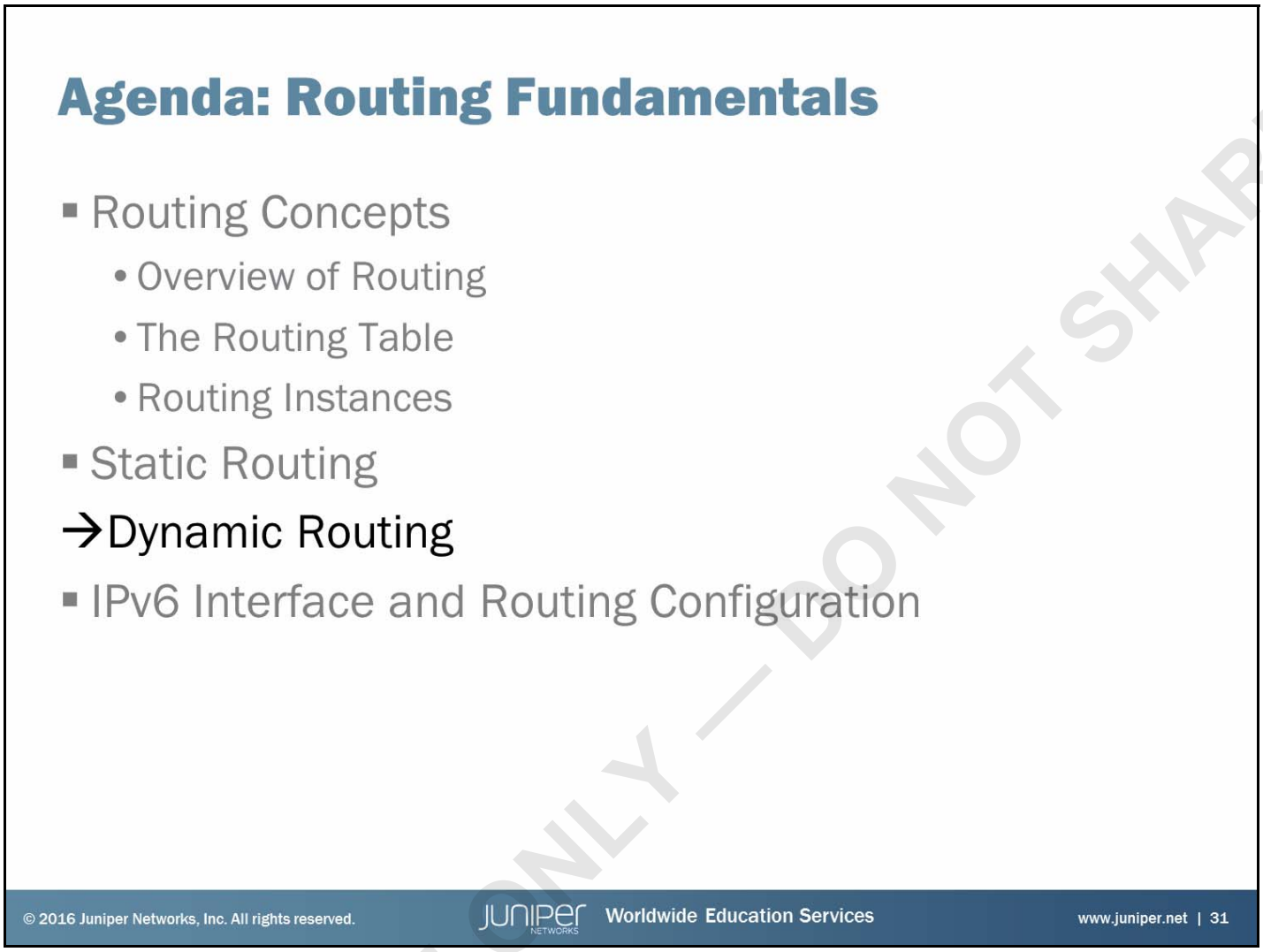
Primary next hop due to default route preference (5)

Secondary next hop due to configured route preference (7)

Qualified Next Hops

The `qualified-next-hop` option allows independent preferences for static routes to the same destination. The slide shows an example using the `qualified-next-hop` option.

In the sample configuration shown on the slide, the 172.30.25.1 next hop assumes the default static route preference of 5, whereas the qualified 172.30.25.5 next hop uses the defined route preference of 7. All traffic using this static route uses the 172.30.25.1 next hop unless it becomes unavailable. If the 172.30.25.1 next hop becomes unavailable, the device uses the 172.30.25.5 next hop. Some vendors refer to this implementation as a *floating static route*.

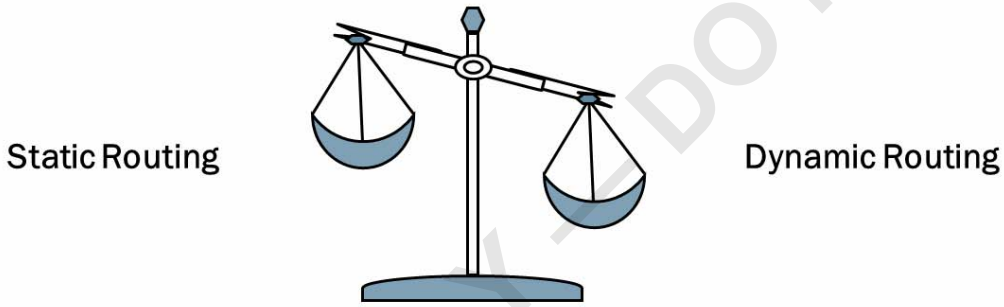


Dynamic Routing

The slide highlights the topic we discuss next.

Dynamic Routing

- Method of dynamically learning routing information
- Dynamic routing has the following benefits:
  - Lower administrative overhead
  - Increased network availability
  - Greater network scalability



Dynamic Routing

Static routing is ideal in small networks where only a few routes exist or in networks where absolute control of routing is necessary. However, static routing has certain drawbacks that might make it cumbersome and hard to manage in large environments where growth and change are constant. For large networks or networks that change regularly, dynamic routing might be the best option.

With dynamic routing, you simply configure the network interfaces to participate in a routing protocol. Devices running routing protocols can dynamically learn routing information from each other. When a device adds or removes routing information for a participating device, all other devices automatically update.

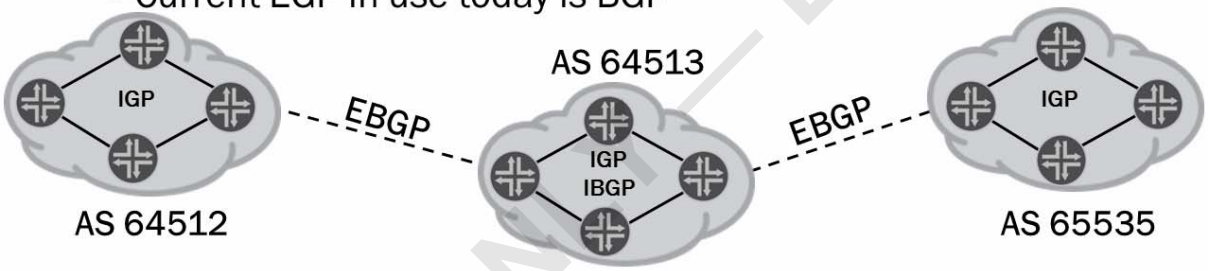
Benefits of Dynamic Routing

Dynamic routing resolves many of the limitations and drawbacks of static routing. Some of the general benefits of dynamic routing include:

- *Lower administrative overhead*: The device learns routing information automatically, which eliminates the need for manual route definition;
- *Increased network availability*: During failure situations, dynamic routing can reroute traffic around the failure automatically (the ability to react to failures when they occur can provide increased network uptime); and
- *Greater network scalability*: The device easily manages network growth by dynamically learning routes and calculating the best paths through a network.

Dynamic Routing Protocols

- A summary of dynamic routing protocols:
  - IGPs operate within a single autonomous system
    - Single network administration that provides for unique routing policy and flexible use of network resources
    - Examples include RIP, IS-IS, and OSPF
  - EGPs operate among different autonomous systems
    - Independent administrative entities that communicate between independent network infrastructures
    - Current EGP in use today is BGP

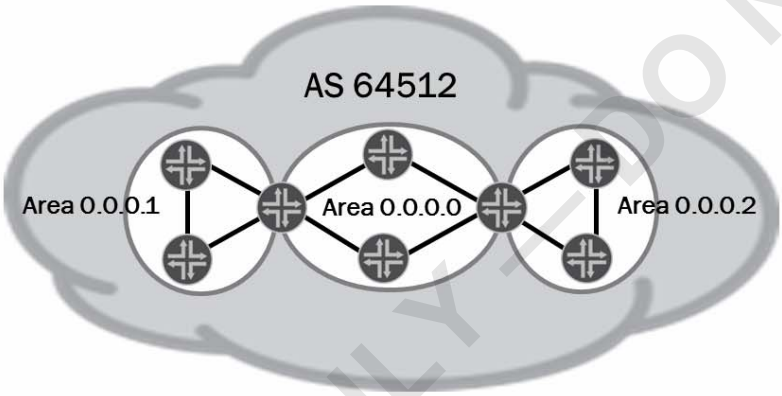


A Summary of Dynamic Routing Protocols

The slide provides a high-level summary of interior gateway protocols (IGPs) and exterior gateway protocols (EGPs).

OSPF Protocol Overview

- A link-state IGP:
  - Reliably floods link-state information to neighbors
  - Creates a complete database of network
  - Calculates best path to each destination
  - Uses areas to incorporate hierarchy and allow for scalability



OSPF Protocol

OSPF is a link-state routing protocol designed for use within an AS. OSPF is an IGP. Link-state protocols allow for faster reconvergence, support larger internetworks, and are less susceptible to bad routing information than distance-vector protocols.

Devices running OSPF send out information about their network links and the state of those links to other routers in the AS. This information transmits reliably to all other routers in the AS by means of link-state advertisements (LSAs). The other routers receive this information, and each router stores it locally. This total set of information now contains all possible links in the network.

In addition to flooding LSAs and discovering neighbors, a third major task of the link-state routing protocol is establishing the link-state database (LSDB). The link-state (or topological) database stores the LSAs as a series of records. The important information for the shortest path determination process is the advertising router's ID, its attached networks and neighboring routers, and the cost associated with those networks or neighbors.

Continued on the next page.

OSPF Protocol (contd.)

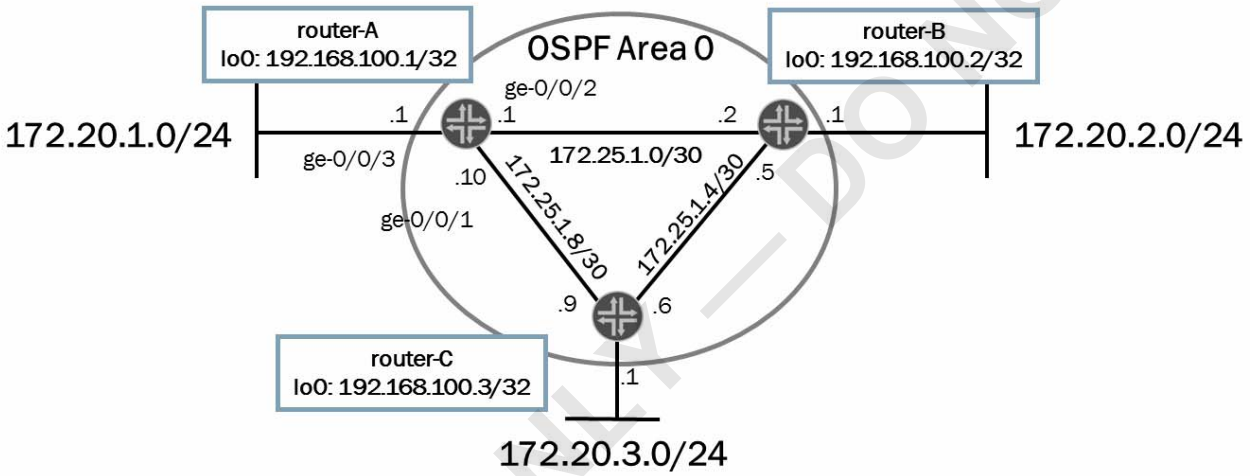
OSPF uses the shortest-path-first (SPF) algorithm (also called the Dijkstra algorithm) to calculate the shortest paths to all destinations. It performs this calculation by calculating a tree of shortest paths incrementally and picking the best candidate from that tree.

OSPF uses areas to allow for a hierarchical organization and to facilitate scalability. An OSPF area is a logical group of routers. The software can summarize the routing information from an OSPF area and the device can pass it to the rest of the network. Areas can reduce the size of the LSDB on an individual router. Each OSPF router maintains a separate LSDB for each area to which it is connected. The LSDB for a given area is identical for all participating routers within that area.

To ensure correct routing knowledge and connectivity, OSPF maintains a special area called the *backbone* area. OSPF designates the backbone area as Area 0.0.0.0. All other OSPF areas must connect to the backbone. All data traffic between OSPF areas must transit the backbone.

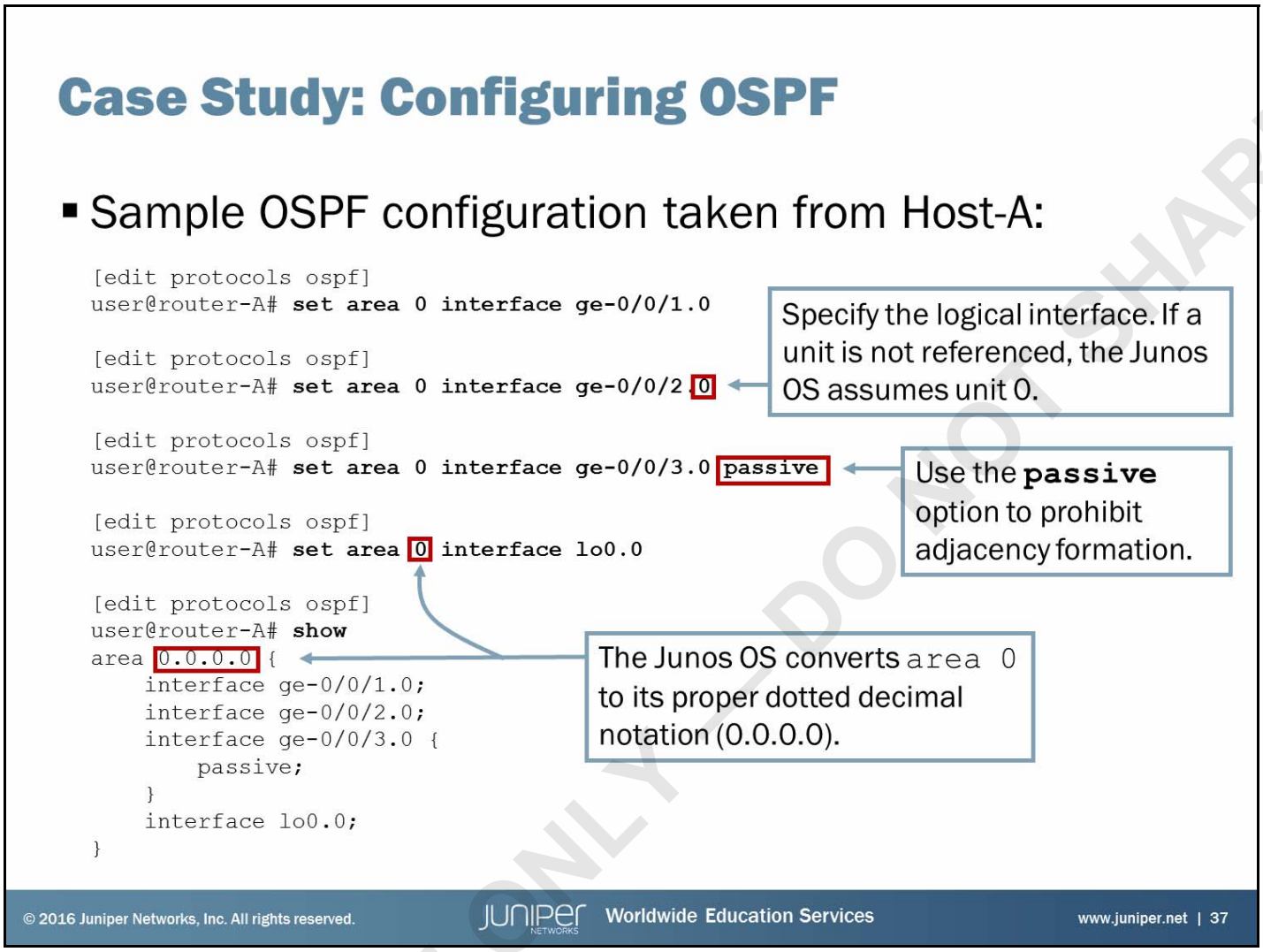
Case Study: Objective and Topology

- Use a single OSPF area to provide connectivity among all connected subnets and loopback addresses; ensure that no adjacencies are formed on interfaces connecting to the 172.20.x.0/24 subnets



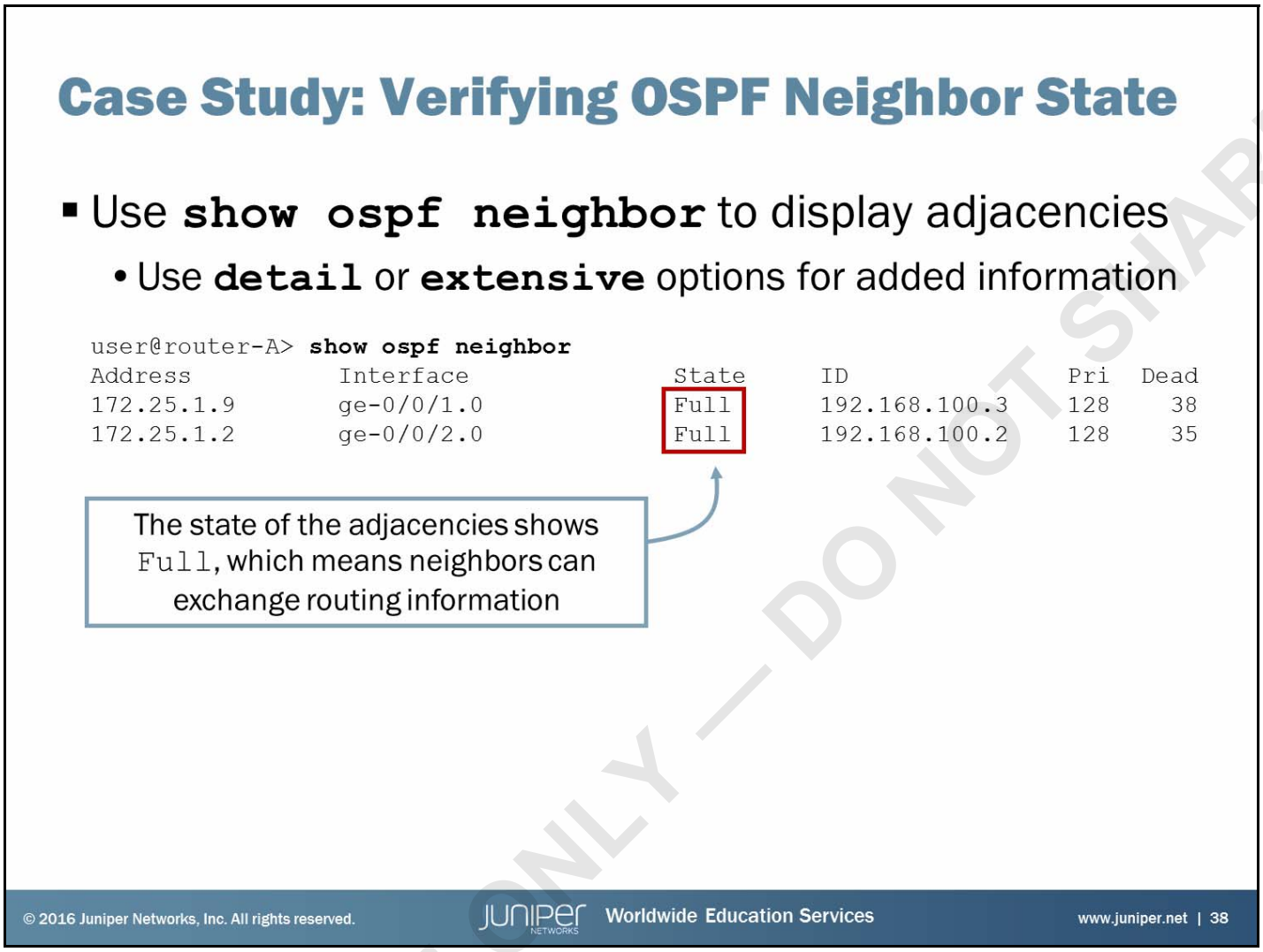
Case Study: Objective and Topology

The slide provides the objective and sample topology used in this case study.



Case Study: Configuring OSPF

The slide illustrates the required OSPF configuration for router-A. Although not shown, router-B and router-C require a similar OSPF configuration to establish adjacencies and share routing information.



Case Study: Verifying OSPF Neighbor State

The slide shows the CLI command used to determine OSPF adjacencies. In the sample output on the slide, you can see that router-A has formed adjacencies with both router-B and router-C. The following is a description of the fields displayed in the output:

- `Address`: The address of the neighbor.
- `Interface`: The interface through which the neighbor is reachable.
- `State`: The state of the neighbor, which can be `Attempt`, `Down`, `Exchange`, `ExStart`, `Full`, `Init`, `Loading`, or `2 Way`.
- `ID`: The router ID of the neighbor.
- `Pri`: The priority of the neighbor to become the designated router, used only on broadcast networks during designated router elections. By default, this value is set to 128, indicating the highest priority and the most likely router to be elected designated router.
- `Dead`: The number of seconds until the neighbor becomes unreachable.

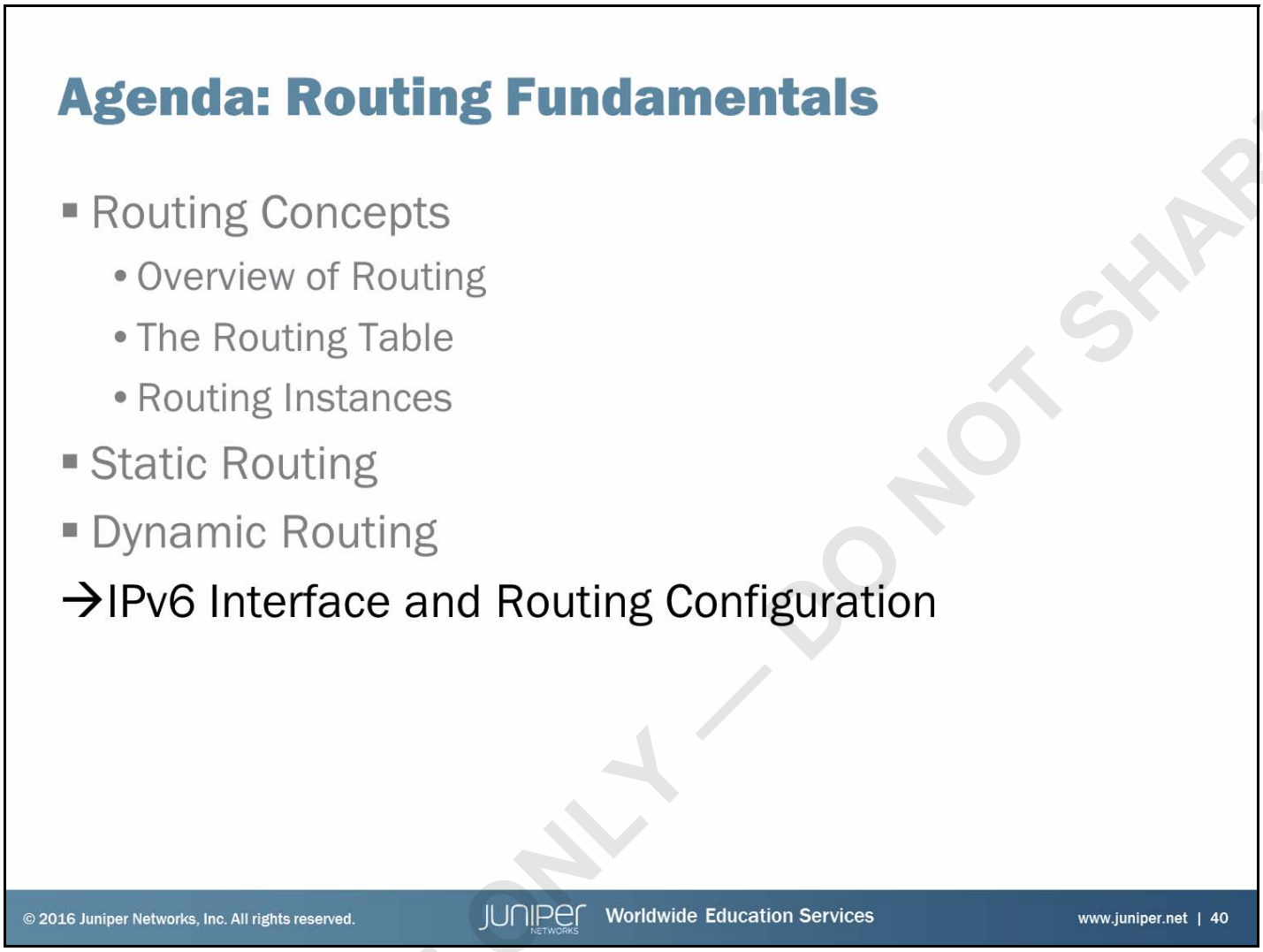
Case Study: Viewing OSPF Routes

- Use **show route protocol ospf** to display OSPF routes

```
user@router-A> show route protocol ospf

inet.0: 15 destinations, 15 routes (15 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

172.20.2.0/24      *[OSPF/10] 00:03:55, metric 2
                    > to 172.25.1.2 via ge-0/0/2.0
172.20.3.0/24      *[OSPF/10] 00:00:04, metric 2
                    > to 172.25.1.9 via ge-0/0/1.0
172.25.1.4/30      *[OSPF/10] 00:03:46, metric 2
                    > to 172.25.1.9 via ge-0/0/1.0
                      to 172.25.1.2 via ge-0/0/2.0
192.168.100.2/32   *[OSPF/10] 00:03:55, metric 1
                    > to 172.25.1.2 via ge-0/0/2.0
192.168.100.3/32   *[OSPF/10] 00:03:46, metric 1
                    > to 172.25.1.9 via ge-0/0/1.0
224.0.0.5/32       *[OSPF/10] 00:16:13, metric 1
                      MultiRecv…
```

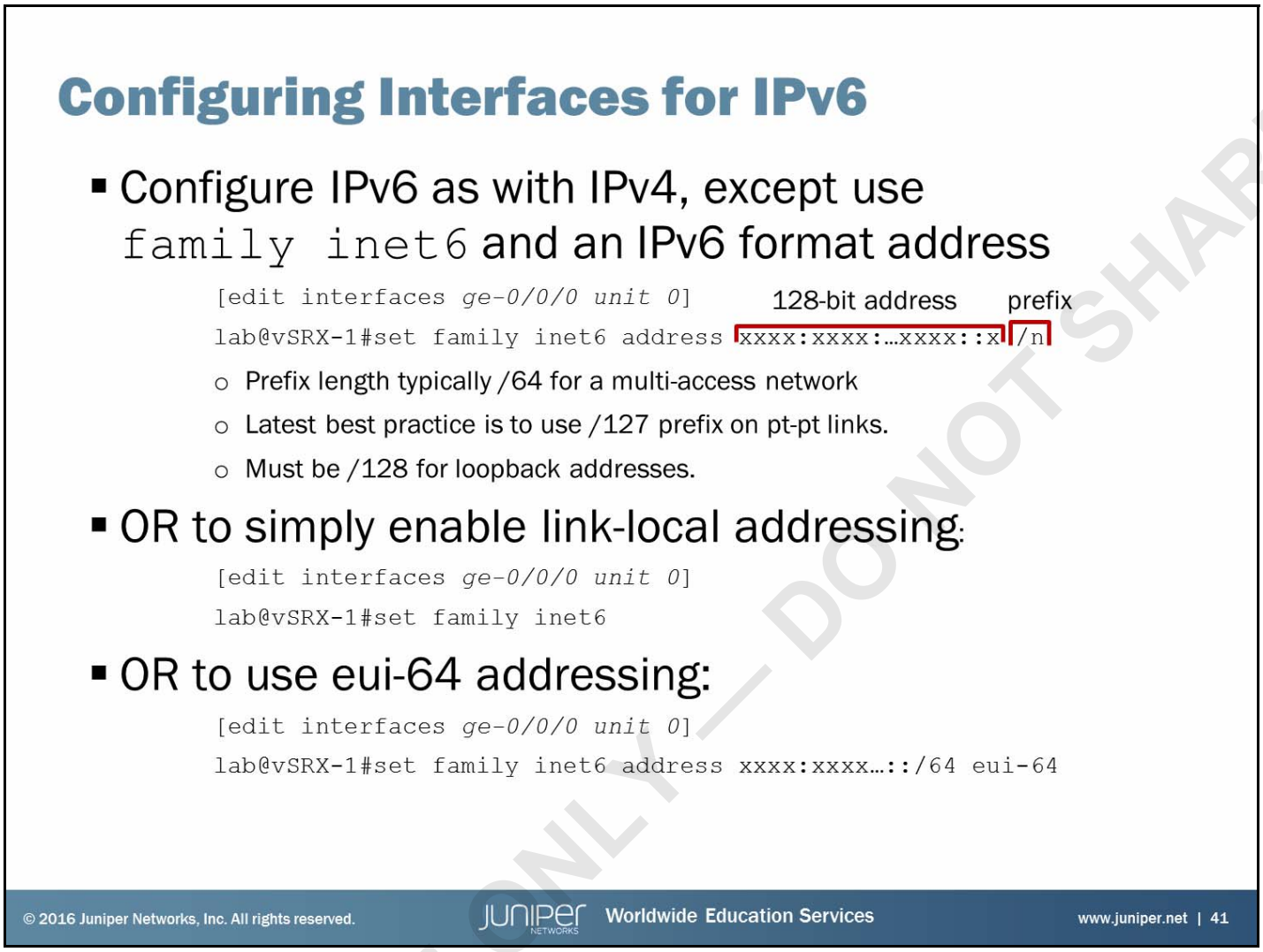

Case Study: Viewing OSPF Routes

The slide illustrates the **show route protocol ospf** command, which displays OSPF routes learned by router-A. Note that router-A does not actually install its directly connected subnets in its route table as OSPF routes—it installs them as direct routes.



IPv6 Interface and Routing Configuration

The slide highlights the topic we discuss next.

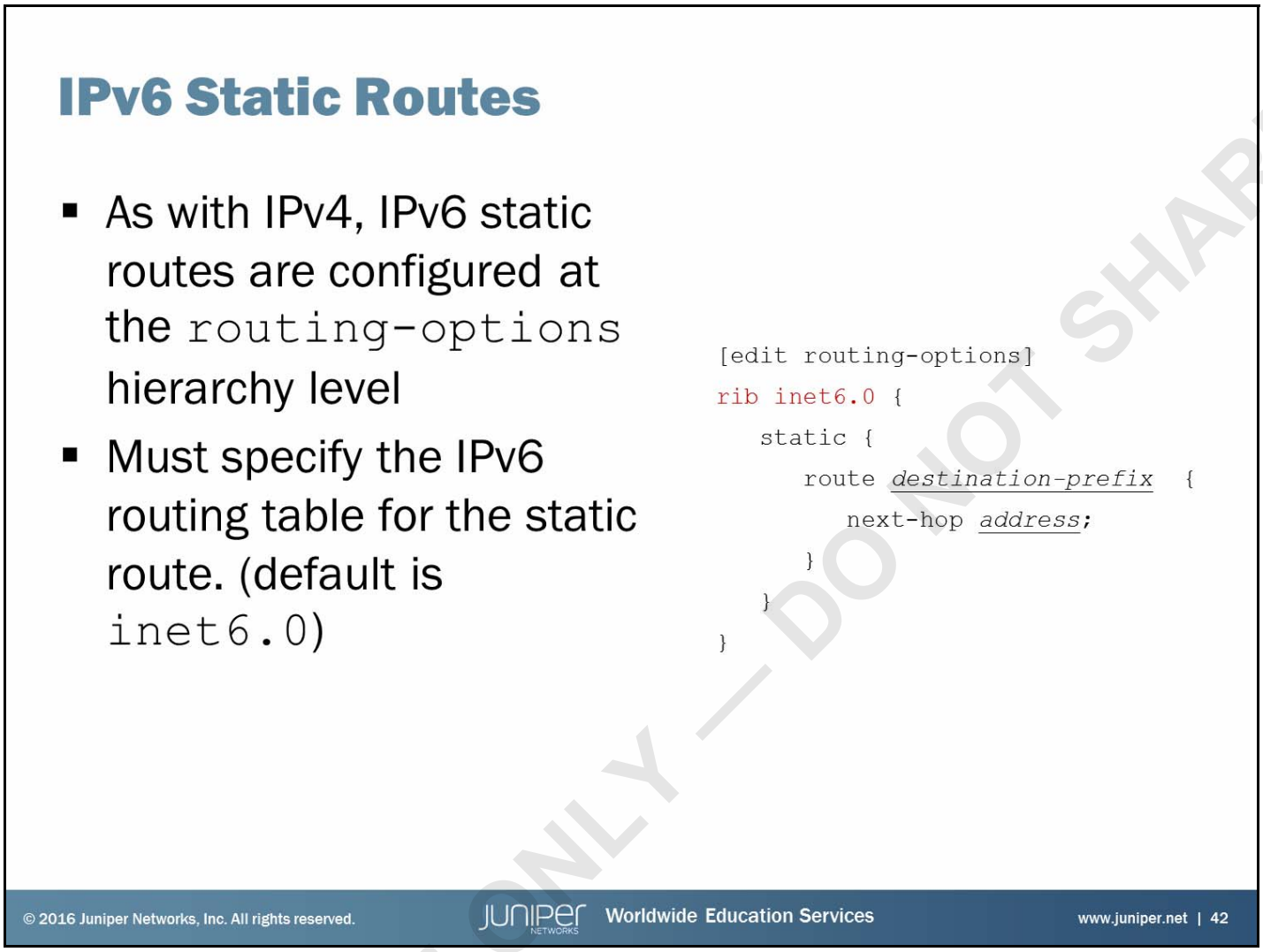


Enabling IPv6

IPv6 is already enabled on Juniper routers, so no global command is required. You must, however, configure IPv6 packet processing on an interface by adding family inet6. Enabling IPv6 on an interface automatically configures the interface's link-local address and activated IPv6 processing for that interface. Additional addresses can be configured, and the link-local address can be manually configured to override the automatically created address.

Configuring IPv6 Addresses

- As you can see from the above example, explicitly configuring an IPv6 address on an interface differs from configuring an IPv4 address only in that you use `family inet6` and an IPv6-format address.
- Simply entering the configuration command `set family inet6` will cause IPv6 packet processing to be enabled on the interface and a link-local address to be generated.
- If you configure eui-64 addressing, the router will automatically generate an interface ID based on the interface's MAC address.
- For multi-access networks, /64 prefix lengths are most commonly used
- For pt-pt links, many prefix lengths have been used, including /64, /124 and /126 but current best practice according to RFC6164 is to use /127 prefix length.
- Details of IPv6 addressing architecture are beyond the scope of this course. Students are referred to RFC4921 and its updates for further information.

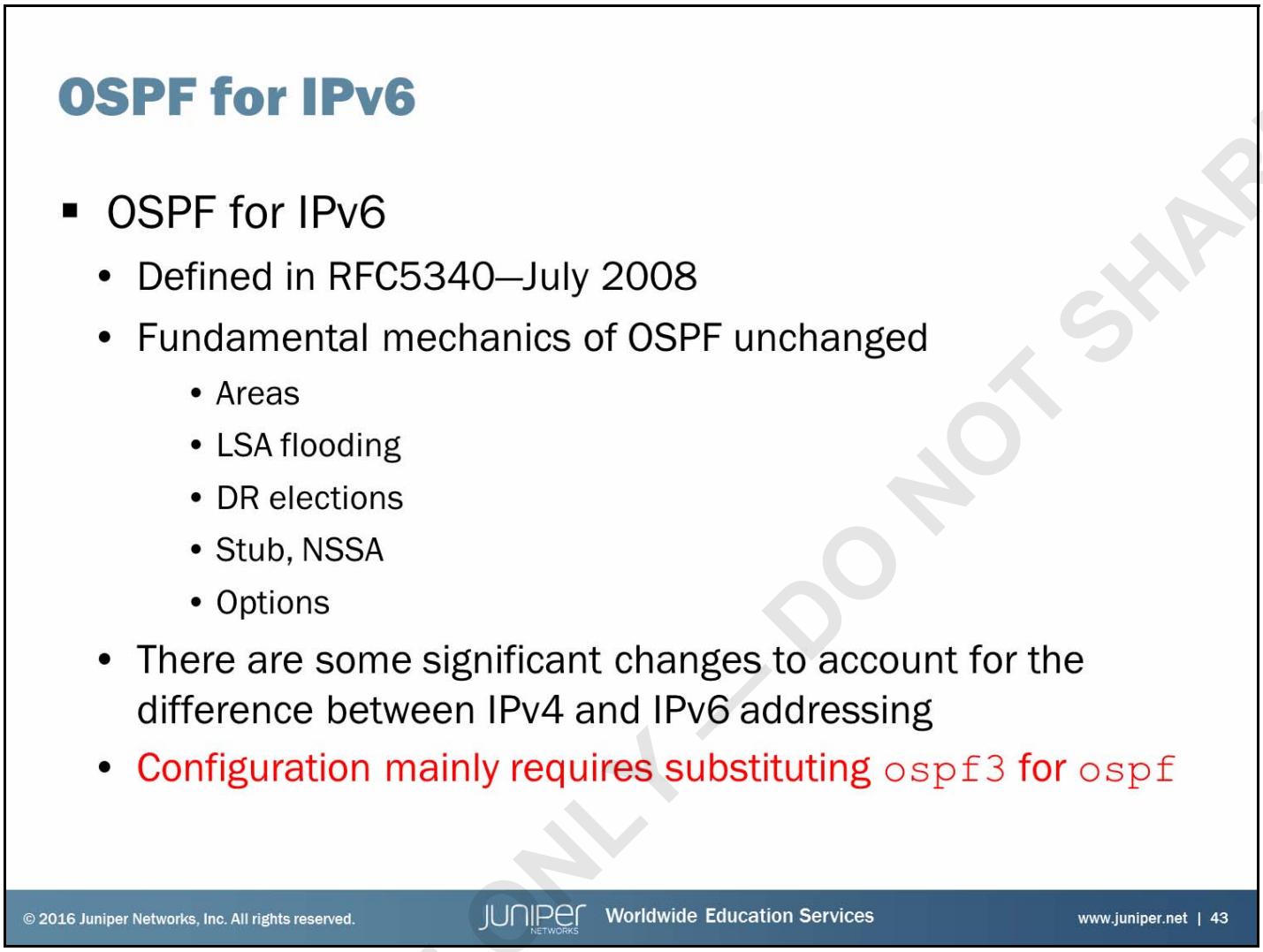


IPv6 Static Routes

IPv6 static routes are conceptually identical to IPv4 static routes. Configuration in Junos is also very similar for IPv4 and IPv6. The only difference is that for IPv6, you need to specify the IPv6 RIB, which by default is `inet6.0`.

A Link-local address may be used as the next-hop for an IPv6 static route, but the outgoing interface must also be specified, since link-local addresses are not globally unique. An example of this is shown below:

```
[edit routing-options]
lab@vSRX-1# set rib inet6.0 static route ::/0 qualified-next-hop fe80::1 interface
    ge-0/0/0
```

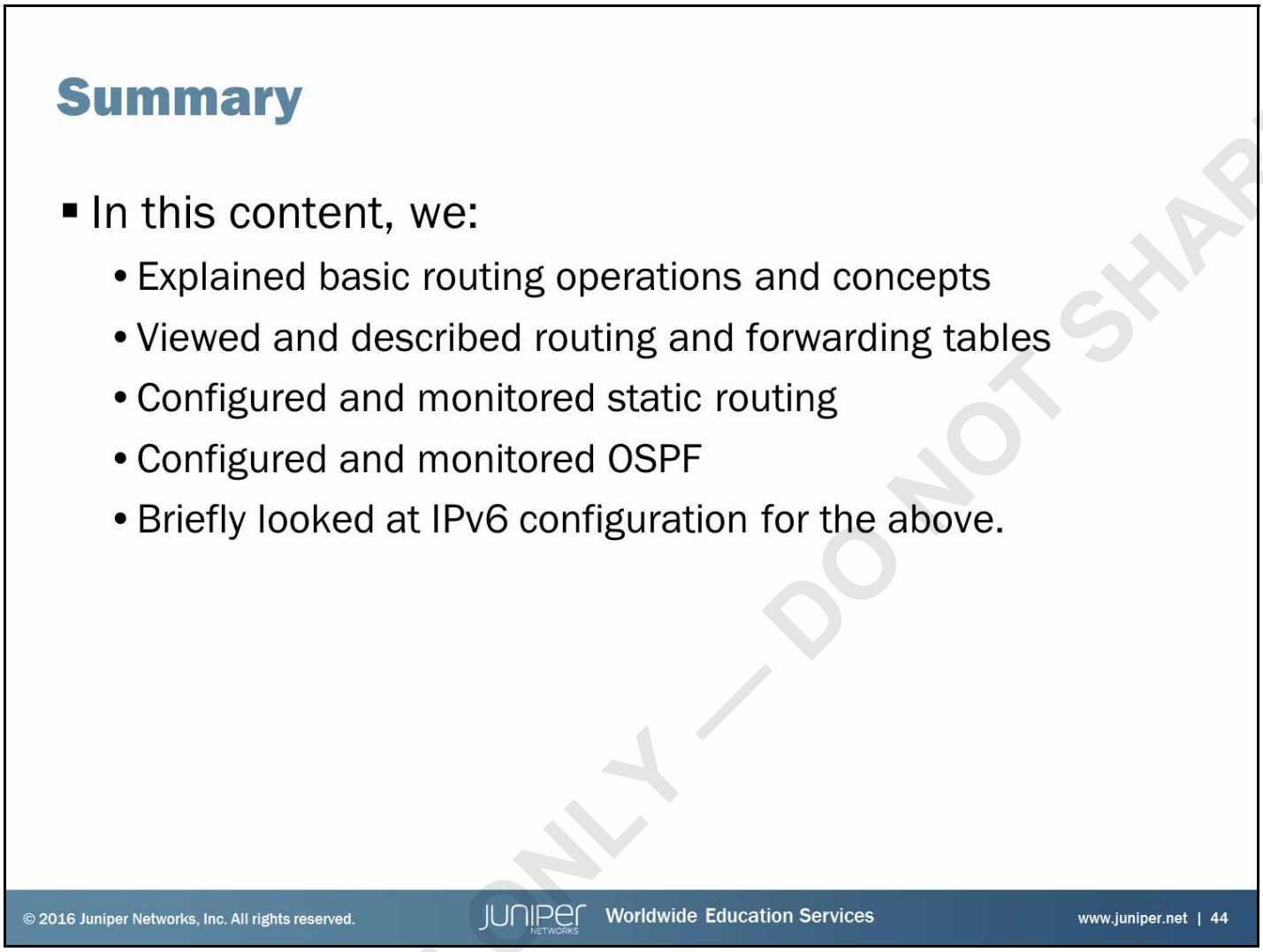


OSPF for IPv6

OSPF version 3 (or just OSPF) for IPv6 is defined in RFC5340, with some additional features, such as grateful restart and authentication, defined on separate documents (RFC5781—OSPFv3 Graceful Restart and RFC4552— Authentication/ Confidentiality for OSPFv3).

OSPFv3 maintains the fundamental mechanisms of OSPF, including LSA flooding scopes, areas, DR election, stub areas, NSSAs, and so on. However, some changes are necessary to account for the differences in IPv4 versus IPv6 addressing.

Notwithstanding the differences under the covers, for you to configure OSPFv3 on the Junos OS, all you need to do is replace `ospf` with `ospf3` in your configuration commands.



We Discussed:

- Basic routing operations and concepts;
- Routing and forwarding tables;
- Configuration and monitoring of static routing; and
- Configuration and monitoring of basic OSPF.
- Basic IPv6 interface and routing configuration.

Review Questions

1. What are two key requirements for routing traffic between two remote devices?
2. List the default IPv4 and IPv6 unicast routing tables.
3. Which primary criterion determines the active routes within the routing table?
4. Which configuration option allows unique preference values for static routes to the same destination?
5. List some advantages in using a dynamic routing protocol instead of static routing.

Review Questions

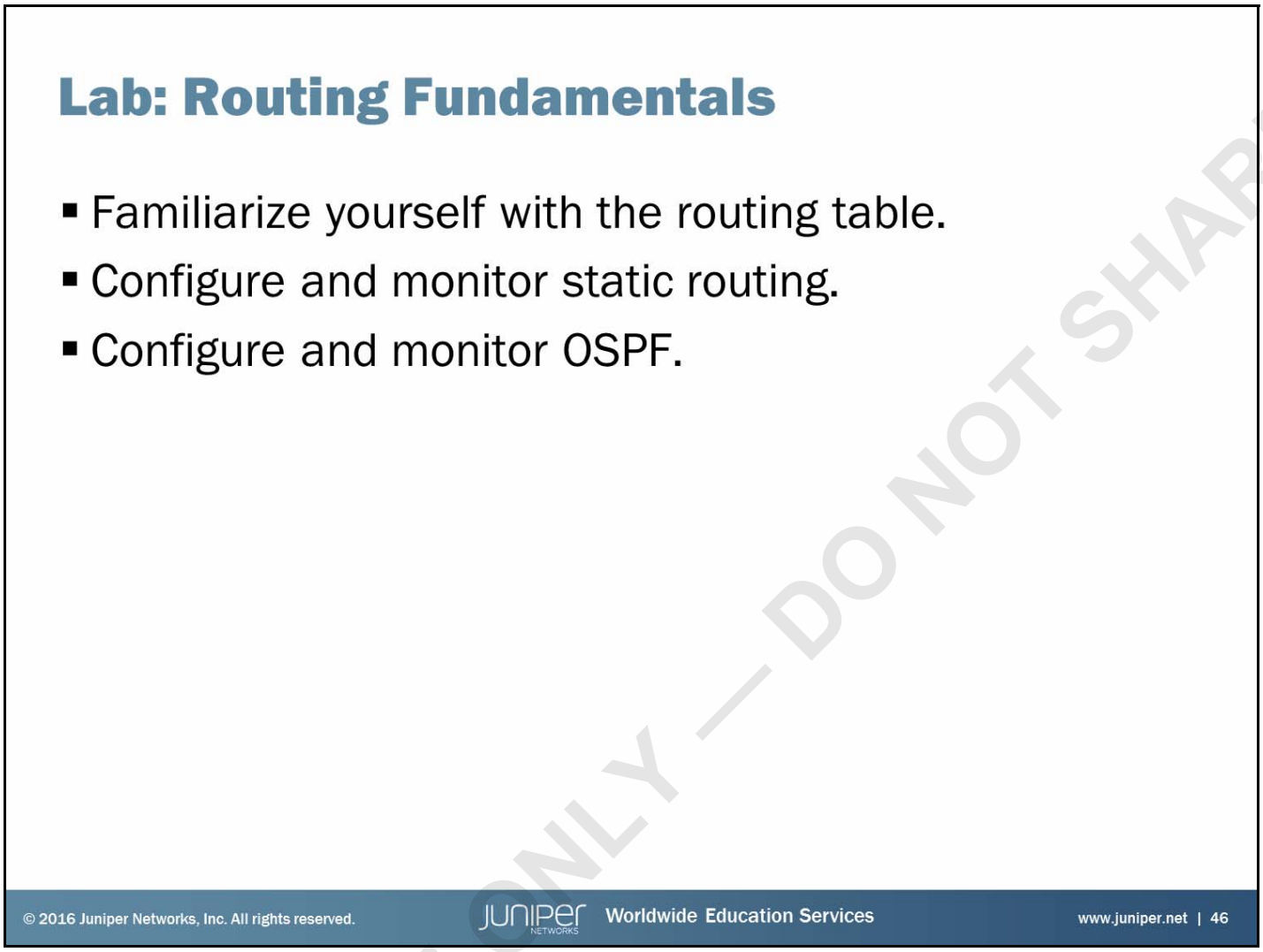
1.

2.

3.

4.

5.



Lab: Routing Fundamentals

The slide provides the objectives for this lab.

Answers to Review Questions

1.

Two key requirements for routing traffic between two remote devices mentioned in this chapter include an end-to-end communications path and the necessary routing information on all participating Layer 3 devices in the communications path.

2.

The default IPv4 and IPv6 unicast routing tables are `inet.0` and `inet6.0`.

3.

The primary criterion for determining the active routes within the routing table is route preference. Lower preference values are more preferred than higher preference values.

4.

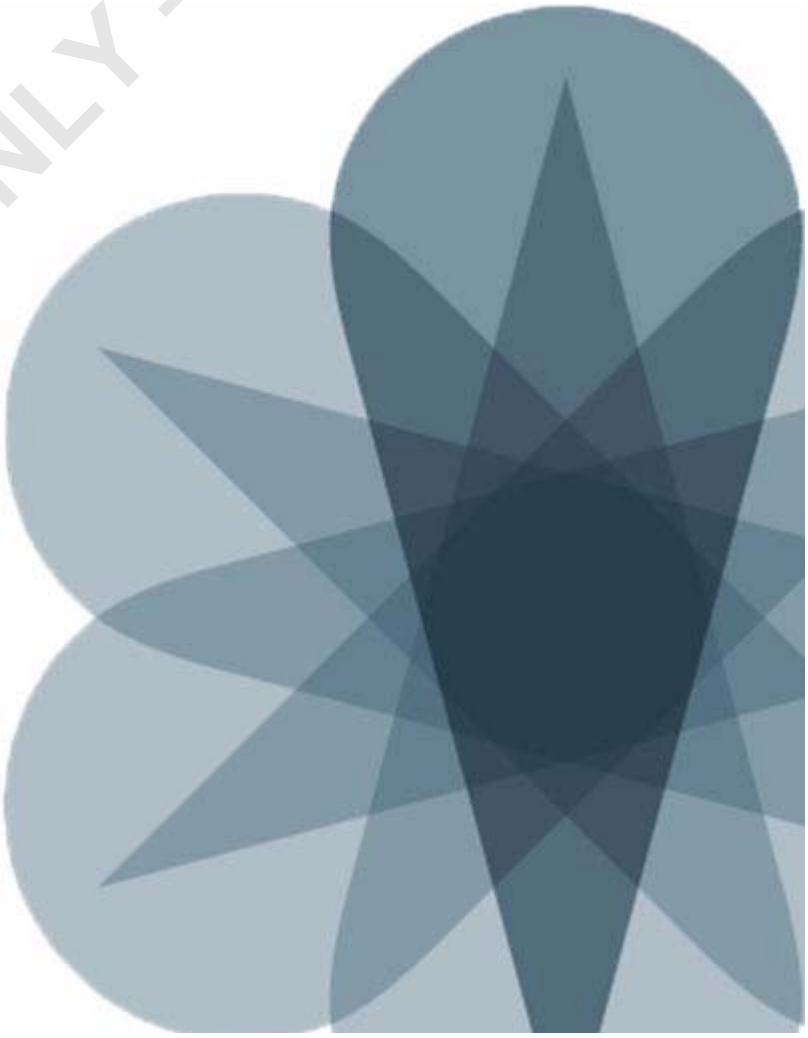
The `qualified-next-hop` CLI option allows unique preference values for static routes to the same destination.

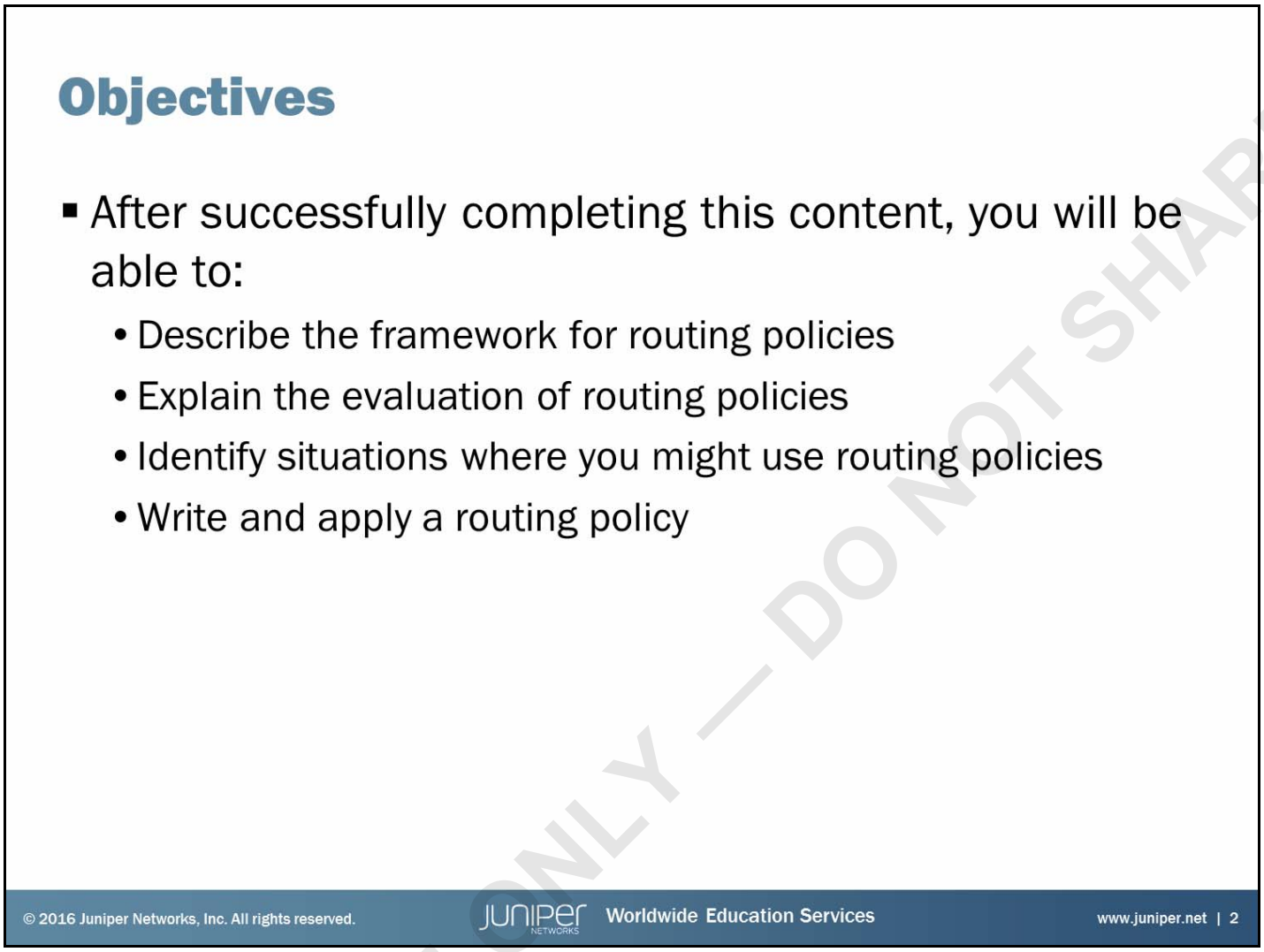
5.

Some of the general benefits of dynamic routing include lower administrative overhead, increased network availability, and greater network scalability.

Junos Routing Essentials

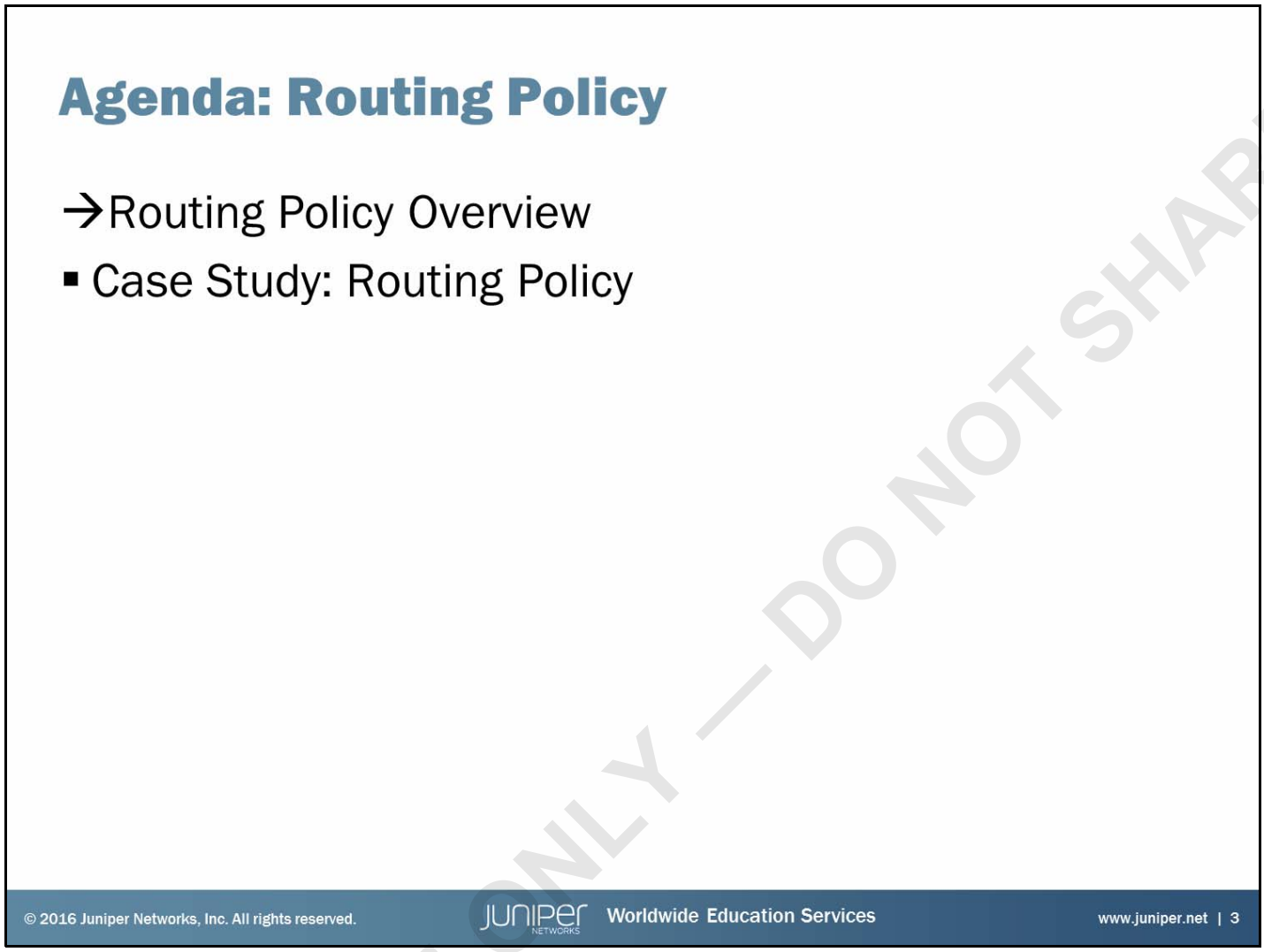
Chapter 3: Routing Policy





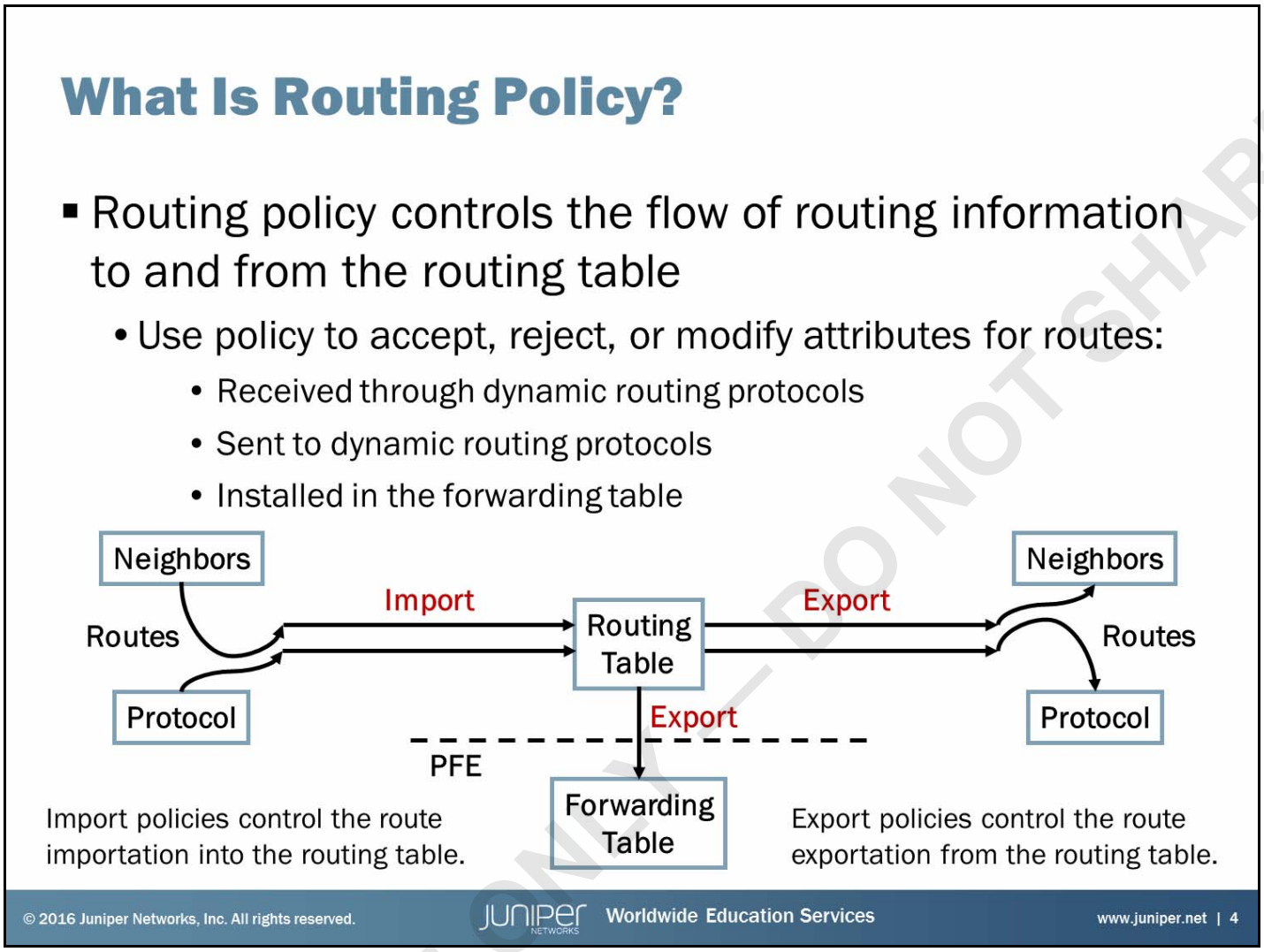
We Will Discuss:

- The framework of routing policies;
- Routing policy evaluation;
- Typical usage scenarios for routing policy; and
- Configuration and application of a routing policy.



Routing Policy Overview

The slide lists the topics we will discuss. We discuss the highlighted topic first.



An Overview of Routing Policy

Routing policy allows you to control the flow of routing information to and from the routing table. You can apply routing policy as information enters the routing table and as information is copied from 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.

Routing policy allows you to control the flow of routing information into the forwarding table. This use allows you to control which routes you install in the forwarding table and to control some of the attributes associated with those routes.

Policies that control how the software imports routes into the routing table are named *import policies*. The software applies import policies before placing routes in the routing table. Thus, an import policy can change the routes that are available in the routing table and can affect the local route selection process.

Policies that control how the software sends routes from the routing table are named *export policies*. The software applies export policies as it exports routes from the routing table to dynamic routing protocols or to the forwarding table. Only active routes are available for export from the routing table. Thus, although an export policy can choose which active routes to export and can modify attributes of those routes, it cannot cause the exportation of inactive routes.

For example, suppose you have an OSPF route (preference 10) and a BGP route (preference 170) for the same prefix. An export policy determines whether to send the active OSPF route and modifies attributes of the route as the software sends it. However, the export policy cannot cause the software to send the inactive BGP route.

The Junos operating system applies export policies as it exports routes from the routing table, so attribute changes do not affect the local routing table; rather, the software applies them to the route while exporting it.

Default Routing Policies

Protocol	Import Policy	Export Policy
BGP	Accept all BGP routes and import into `inet.0`	Accept all active BGP routes
OSPF	Accept all OSPF routes and import into `inet.0`	Reject everything (protocol floods by default)
IS-IS	Accept all IS-IS routes and import into `inet.0`	Reject everything (protocol floods by default)
RIP	Accept all RIP routes from explicitly configured neighbors and import into `inet.0`	Reject everything

Default Routing Policies

Every protocol has a default import policy and a default export policy. The chart on the slide summarizes the default import and export policies for several common routing protocols.

BGP's default import policy is to accept all routes from BGP neighbors and install them in the routing table. BGP's default export policy is to advertise all active BGP routes. For BGP, you can configure import and export policies at the protocol, group, and neighbor levels.

The default OSPF import policy is to import all OSPF routes. As a link-state protocol, OSPF maintains a consistent link-state database (LSDB) throughout each OSPF area by flooding link-state advertisements (LSAs). You cannot apply policy to affect the maintenance of the local LSDB or the flooding of LSAs. Additionally, you cannot apply policy that prevents the software from installing internal (including interarea) routes in the routing table. (A link-state protocol assumes that all devices have the same routing information for internal routes, which causes all devices 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 a policy that blocks external routes.

Continued on the next page.

Default Routing Policies (contd.)

The default OSPF export policy (which rejects everything) does not cause the system to stop flooding LSAs through the area. Rather, the system always floods LSAs throughout the OSPF area, and the routing policy cannot control that behavior. The default export policy simply blocks the advertising of additional routes from other sources to OSPF neighbors. If you want to advertise other routes through OSPF, you must configure an explicit export policy.

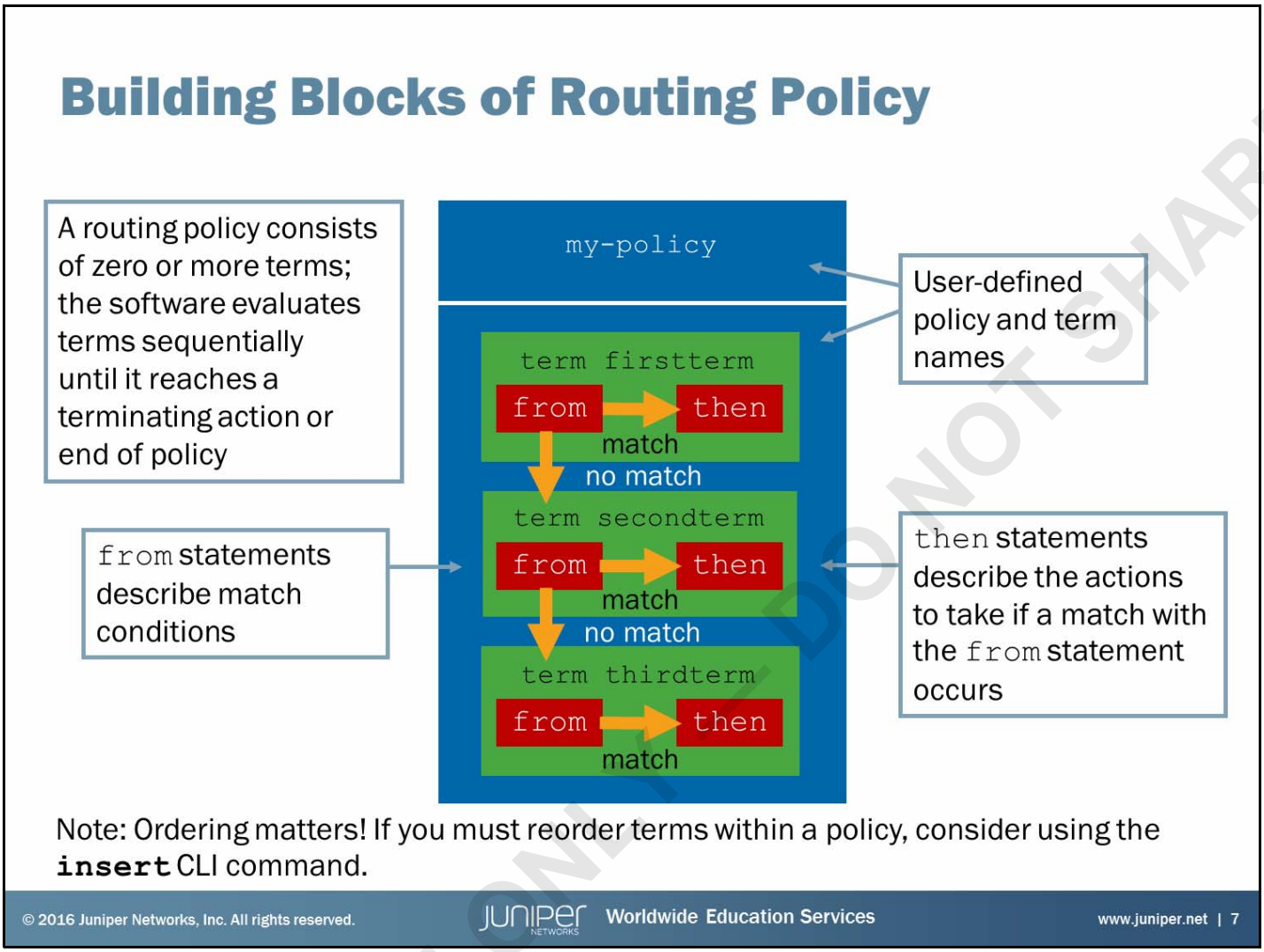
Because link-state protocols rely on all participating devices having consistent LSDBs, you can configure import and export policies only at the protocol level.

The default policy for RIP is to import all routes learned from explicitly configured neighbors. The software ignores routes learned from neighbors not explicitly defined within the configuration. By default, the software does not export routes to RIP neighbors, including RIP routes. Thus, to advertise *any* routes to RIP neighbors, you must configure an export policy that matches and accepts RIP routes as shown in the following sample output:

```
[edit policy-options]
user@router# show
policy-statement export-rip-routes {
    term match-rip-routes {
        from protocol rip;
        then accept;
    }
}
```

For RIP, you can apply import policies at the protocol level and neighbor level, whereas you can configure export policies only at the group level as shown in the following sample output:

```
[edit protocols rip]
user@router# show
group my-rip-group {
    export export-rip-routes;
    neighbor ge-0/0/1.0;
    neighbor se-1/0/0.0;
}
```



Building Blocks of Routing Policy

Routing policies contain ordered groups of terms. You give routing policies a name that you use to identify them at different locations in the configuration.

Terms are the basic building blocks of all Junos OS 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), 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 provides a meaningful identifier that you can use when referring to the term.

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

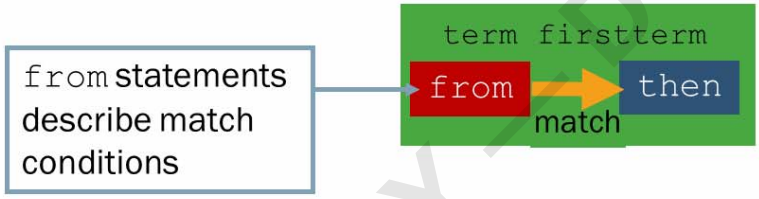
If a route matches all the conditions in the *from* statement of a term, the Junos OS executes all the actions specified in the *then* statement of that term. Provided that one of those actions is a *terminating action*, the evaluation of the policy stops.

The actions that control the acceptance and rejection of routes (`accept` and `reject`) are terminating actions. Using these terminating actions results in a *first-match* policy evaluation because the software takes the specified action immediately and performs no further evaluation of the policy.

When the Junos OS evaluates a policy, it evaluates each term sequentially. If needed, you can use the **`insert`** CLI command in configuration mode to modify the order in which terms appear.

Common Match Criteria

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



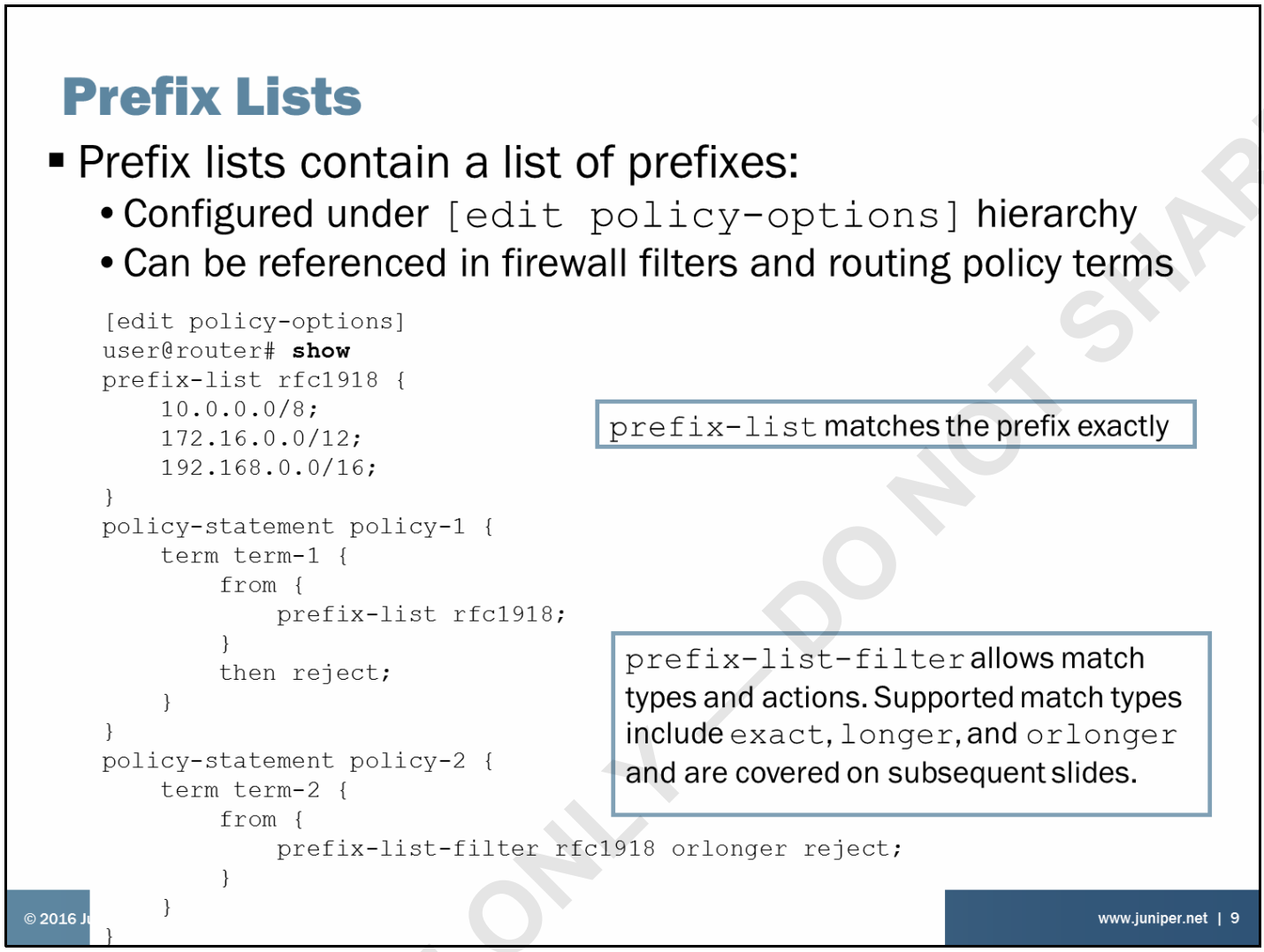
Note: If you omit the `from` statement, all routes match and will take the specified action.

Common Selection Criteria

The 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 information. We highlight the `route-filter` and `prefix-list` match criteria options on subsequent slides.

Note that if you omit the `from` statement in a policy or a policy's term, the software subjects all routes to the referenced action specified in the `then` statement.

You can view the full list of match criteria in the CLI interactive help and in the *Junos Policy Framework Configuration Guide*.

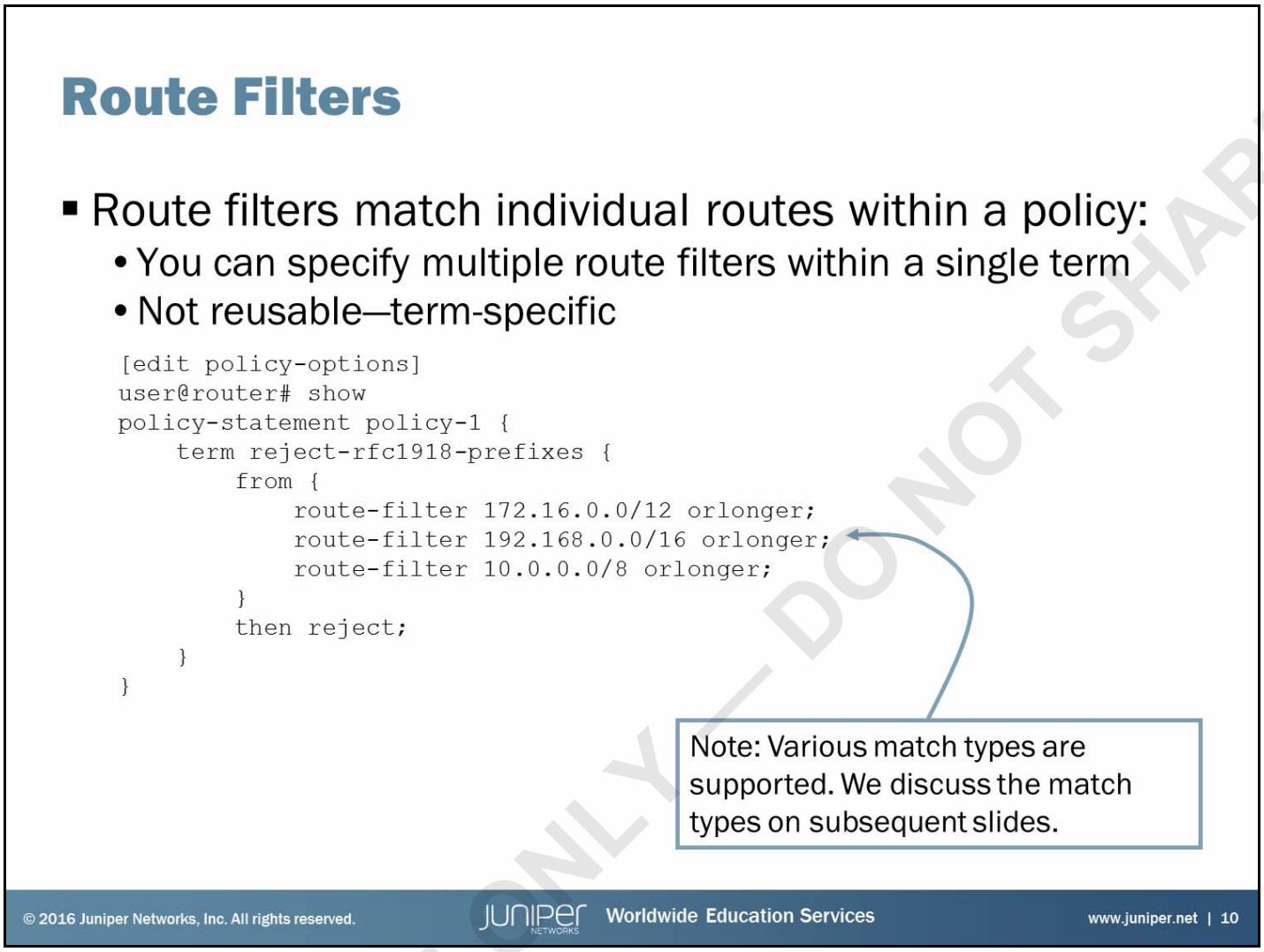


Prefix Lists

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

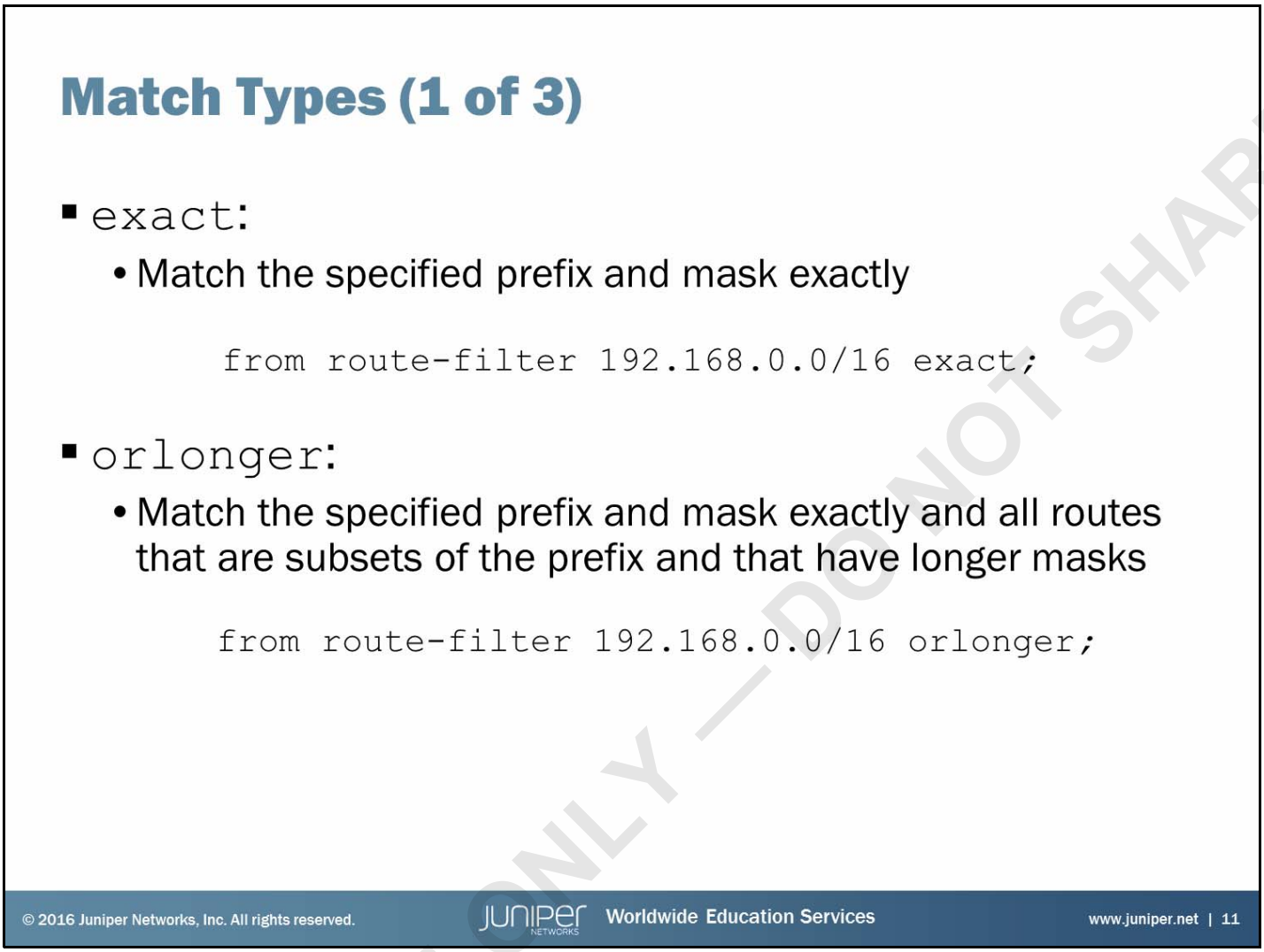
Prefix lists are named lists of prefixes configured under the `[edit policy-options]` hierarchy. One of the advantages of prefix lists is that you can use them in multiple places. You can reference prefix lists in multiple terms in a single policy or in different policies. In addition, you can use prefix 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 match only if they *exactly* match one of the prefixes in the list. When referenced in a `prefix-list-filter` statement, 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 later in this chapter.



Route Filters

Route filters are lists of prefixes configured within a single routing policy or policy term. Unlike prefix lists, they are not reusable but rather are specific to the policy or 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 with 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.

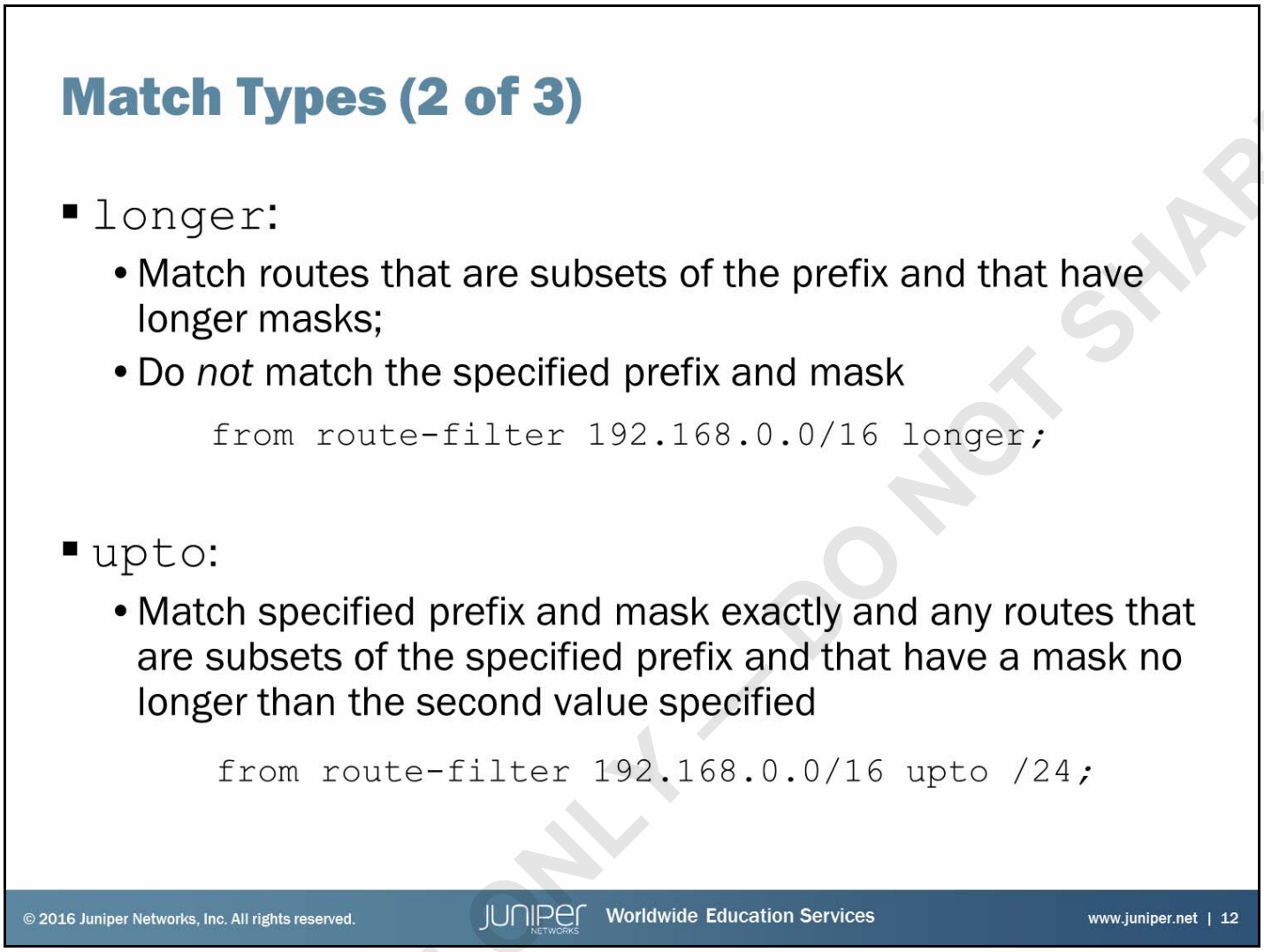


exact

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, matches 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.

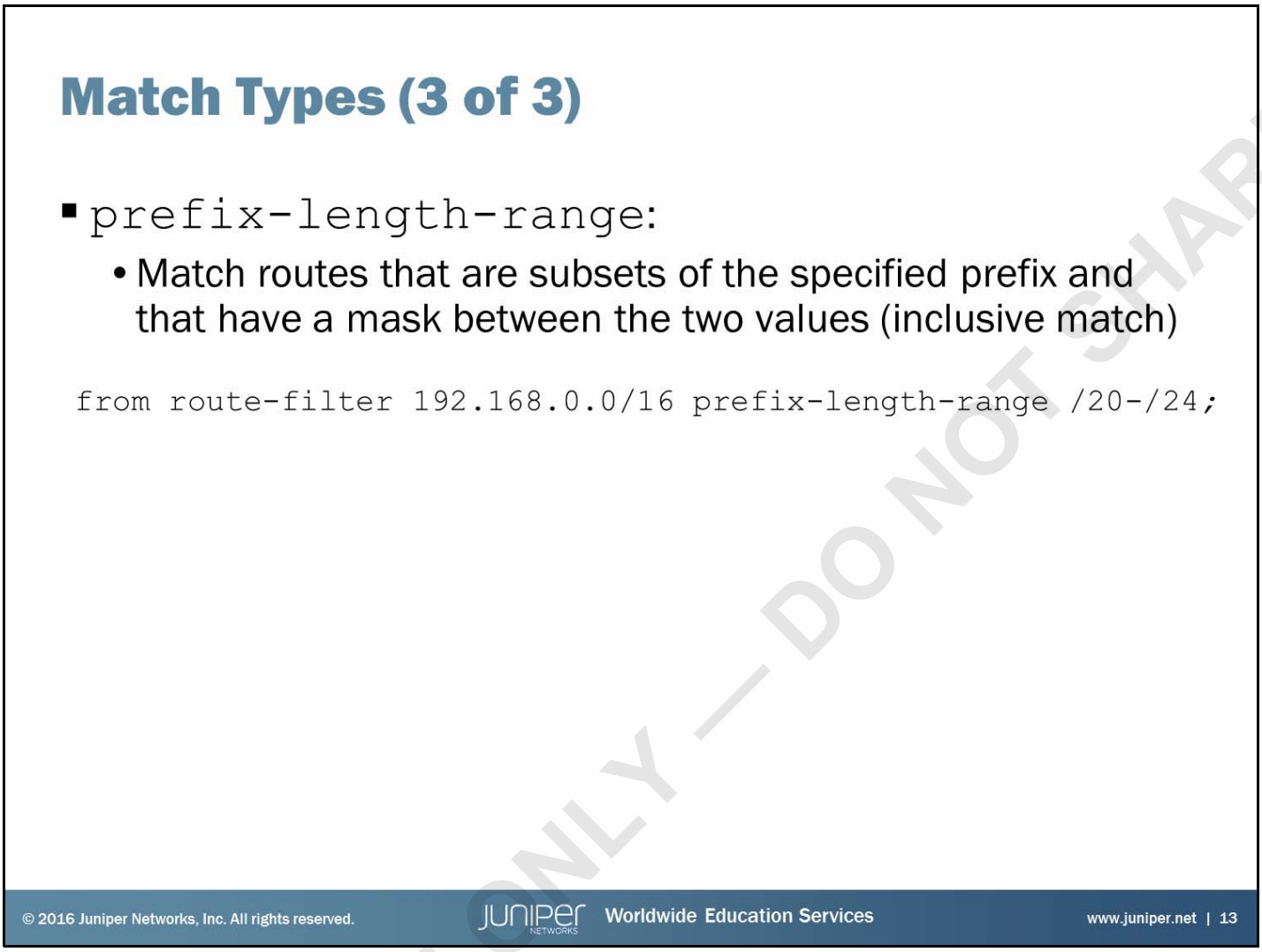


longer

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, whereas 192.168.0.0/16 does not.

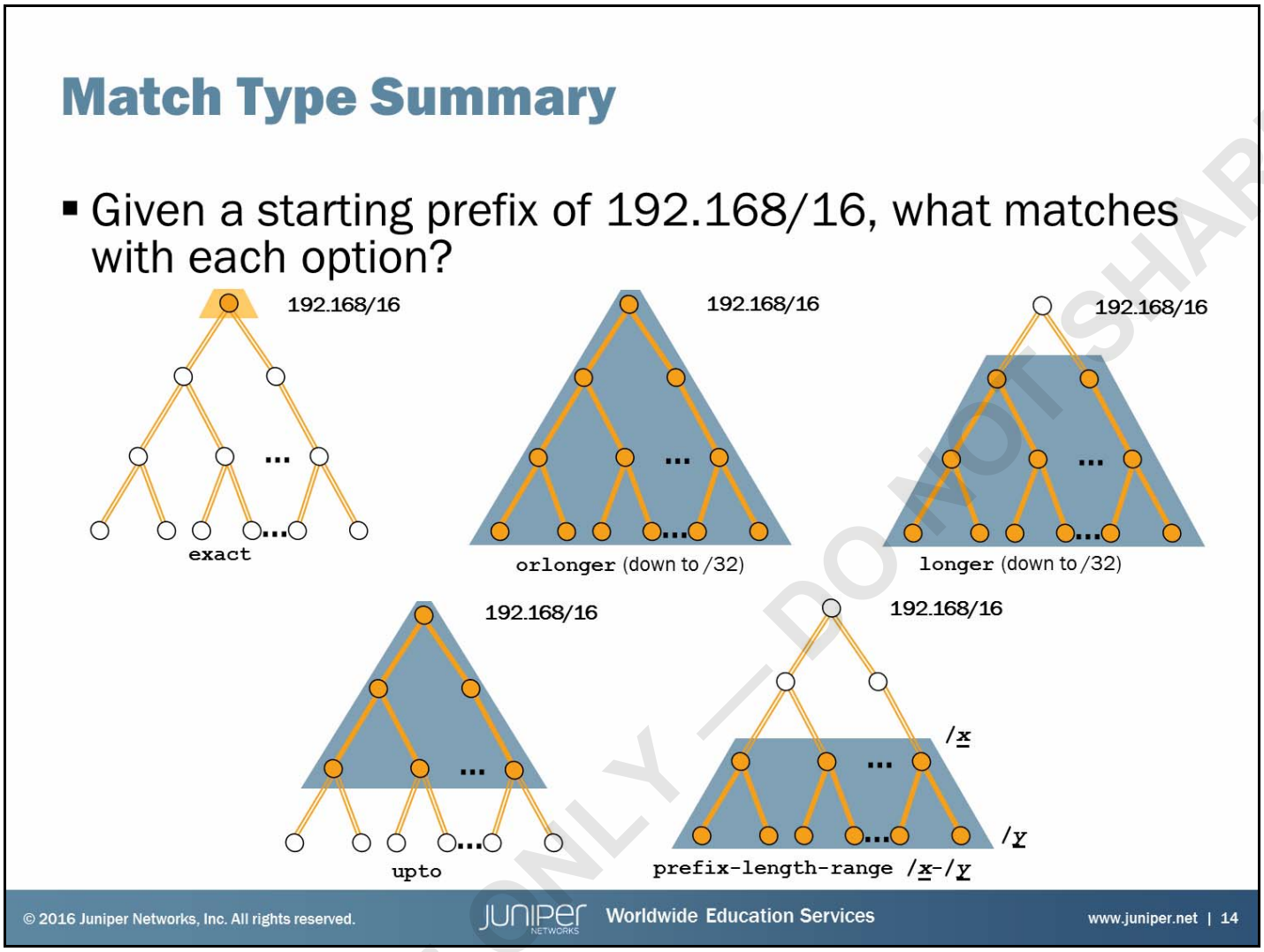
upto

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.



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.



Route Filter Match Types

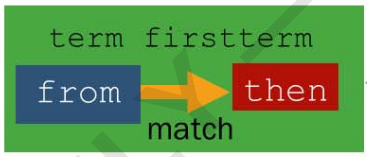
The diagram on the slide illustrates the different route filter match types.

Some intricacies exist to the way that Junos devices evaluate route filters when you use differing mask lengths. The Junos OS 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 the software 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 "How a Route List is Evaluated" section 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:
    - `community` (add, delete, and set)
    - `preference`



`then` statements describe the actions to take if a match with the `from` statement occurs

Common Actions

Some common routing policy actions include the terminating actions of `accept` and `reject`. These are named *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.

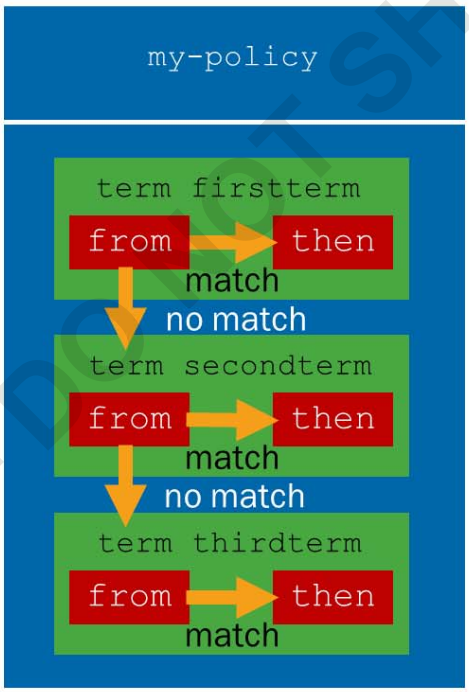
Other common routing policy actions affect the flow of policy evaluation. The `next term` and `next policy` actions cause the Junos OS to evaluate the next term or next policy, respectively.

Other common actions modify protocol attributes such as BGP communities, route preference, and many others.

Implementing Routing Policy (1 of 2)

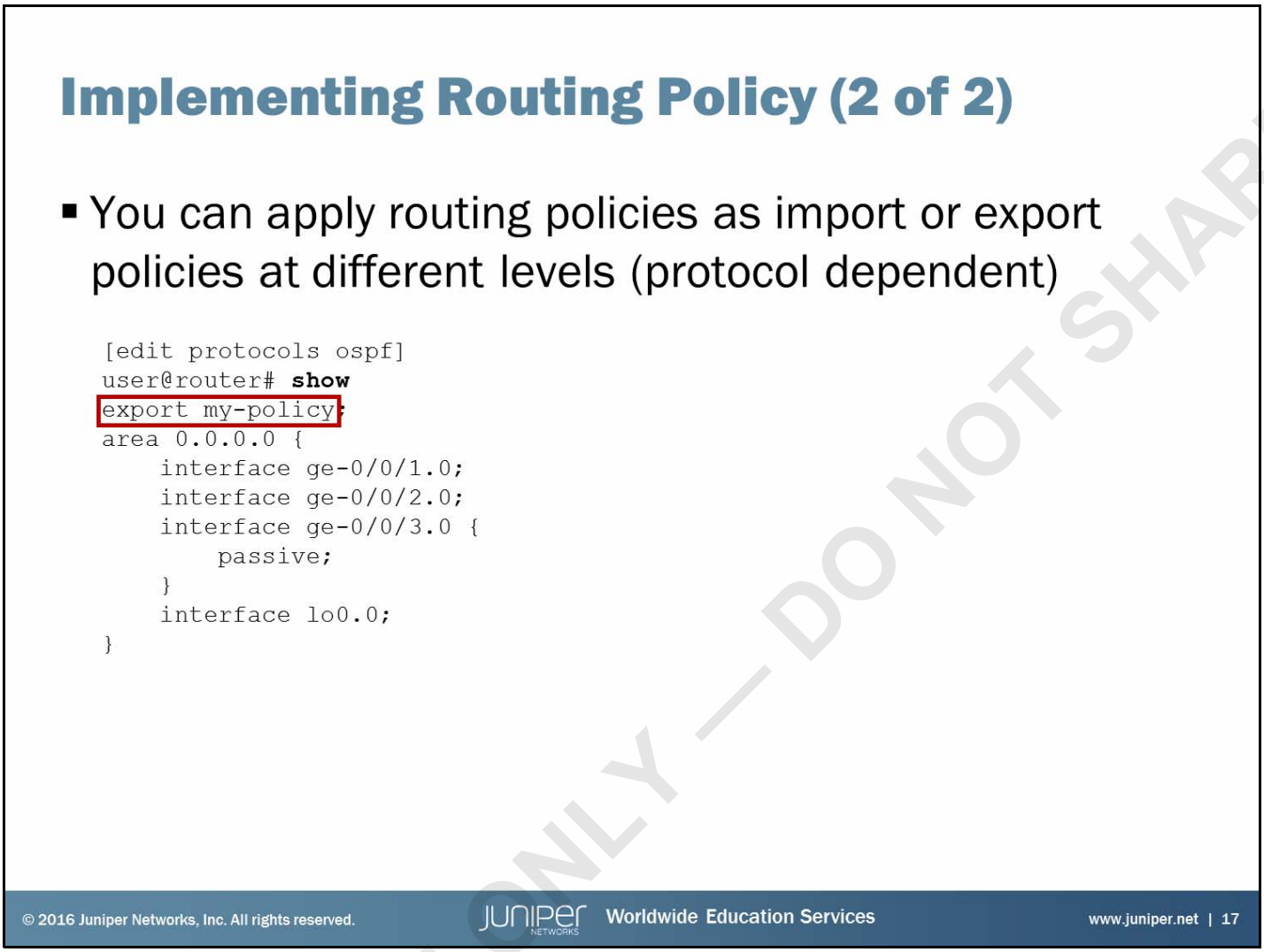
- Definition of routing policy is always under the `[edit policy-options]` hierarchy:

```
[edit policy-options]
user@router# show
policy-statement my-policy {
    term accept-local-route {
        from {
            protocol direct;
            interface ge-0/0/0;
        }
        then accept;
    }
    term accept-some-static-routes {
        from {
            protocol static;
            route-filter 172.18.1.0/24 exact;
            route-filter 172.18.2.0/24 exact;
        }
        then accept;
    }
    term accept-rip-routes {
        from protocol rip;
        then accept;
    }
}
```



Defining Routing Policy

Implementing policy requires two distinct steps. The first step is to define the routing policy. In the Junos OS, you define routing policy under the `[edit policy-options]` hierarchy level. The slide illustrates a sample policy with three distinct terms.

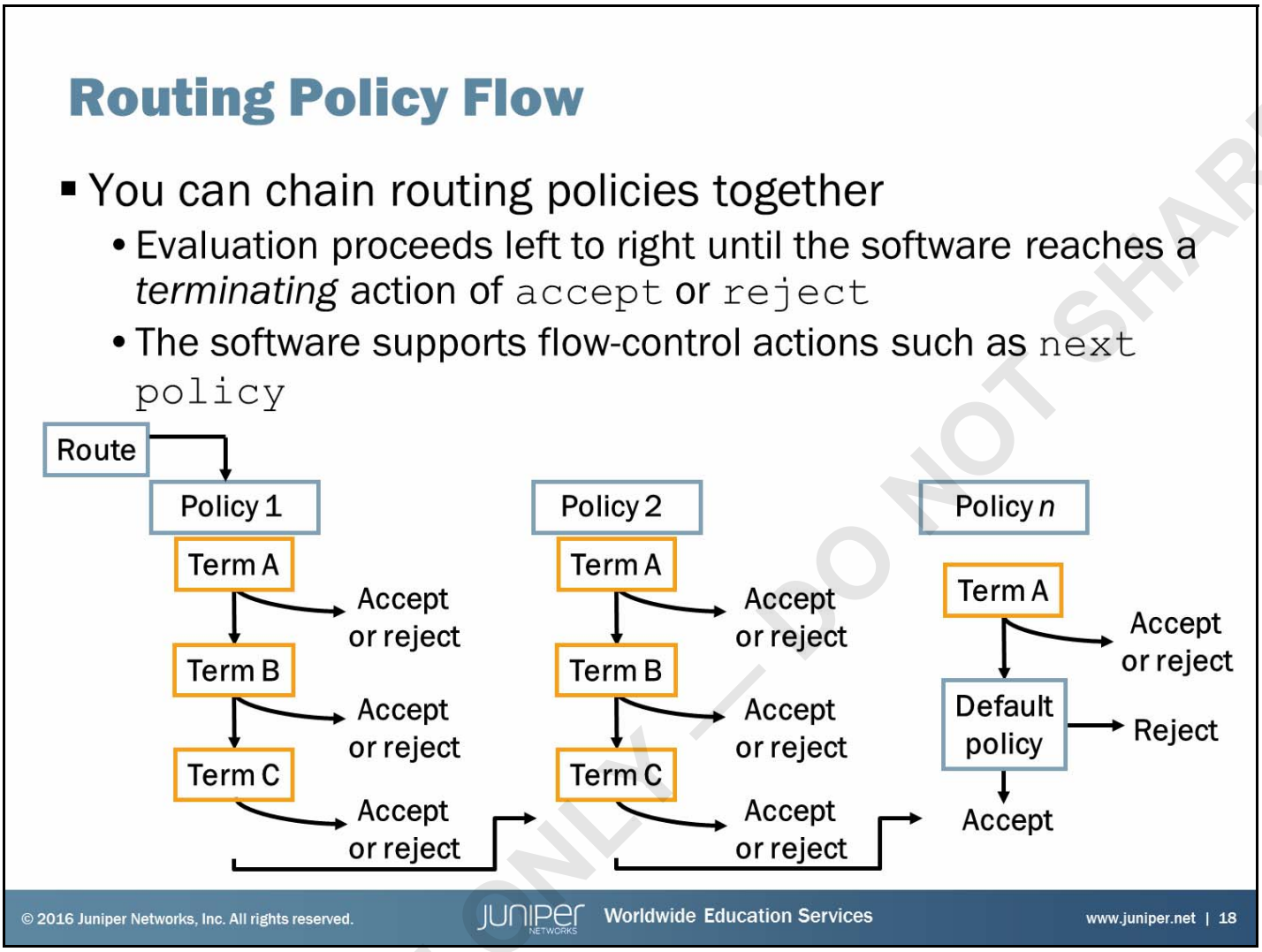


Applying Routing Policy

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 policy configuration at all these levels. For example, OSPF allows only protocol-level export and import policies because of the need to maintain a globally consistent LSDB. (This same requirement also limits the number of changes an OSPF import or export policy can make.)

Junos devices always apply 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 apply to lower levels of the configuration if no other policy configuration exists at that level. However, if you configure a policy at a lower level, the system applies only that policy.



Policy Chaining

You can cascade policies to form a chain of policy processing. You can create this chain of policies to solve a complex set of route manipulation tasks in a modular manner.

The Junos OS evaluates policies from left to right based on the order in which they are applied to a routing protocol. The Junos OS checks the match criteria of each policy and performs the associated action when a match occurs. If the first policy does not match or if the match is associated with a nonterminating action, the Junos OS evaluates the route against the next policy in the chain. This pattern repeats itself for all policies in the chain. The Junos OS ultimately applies the default policy for a given protocol when no terminating actions occur while evaluating the user-defined policy chain. We defined the default routing policies earlier in this chapter.

Policy processing stops once a route meets a terminating action—unless you are grouping policies with Boolean operators. Grouping policies for logical operations, such as AND or OR, is a subject that is beyond the scope of this class.

Continued on the next page.

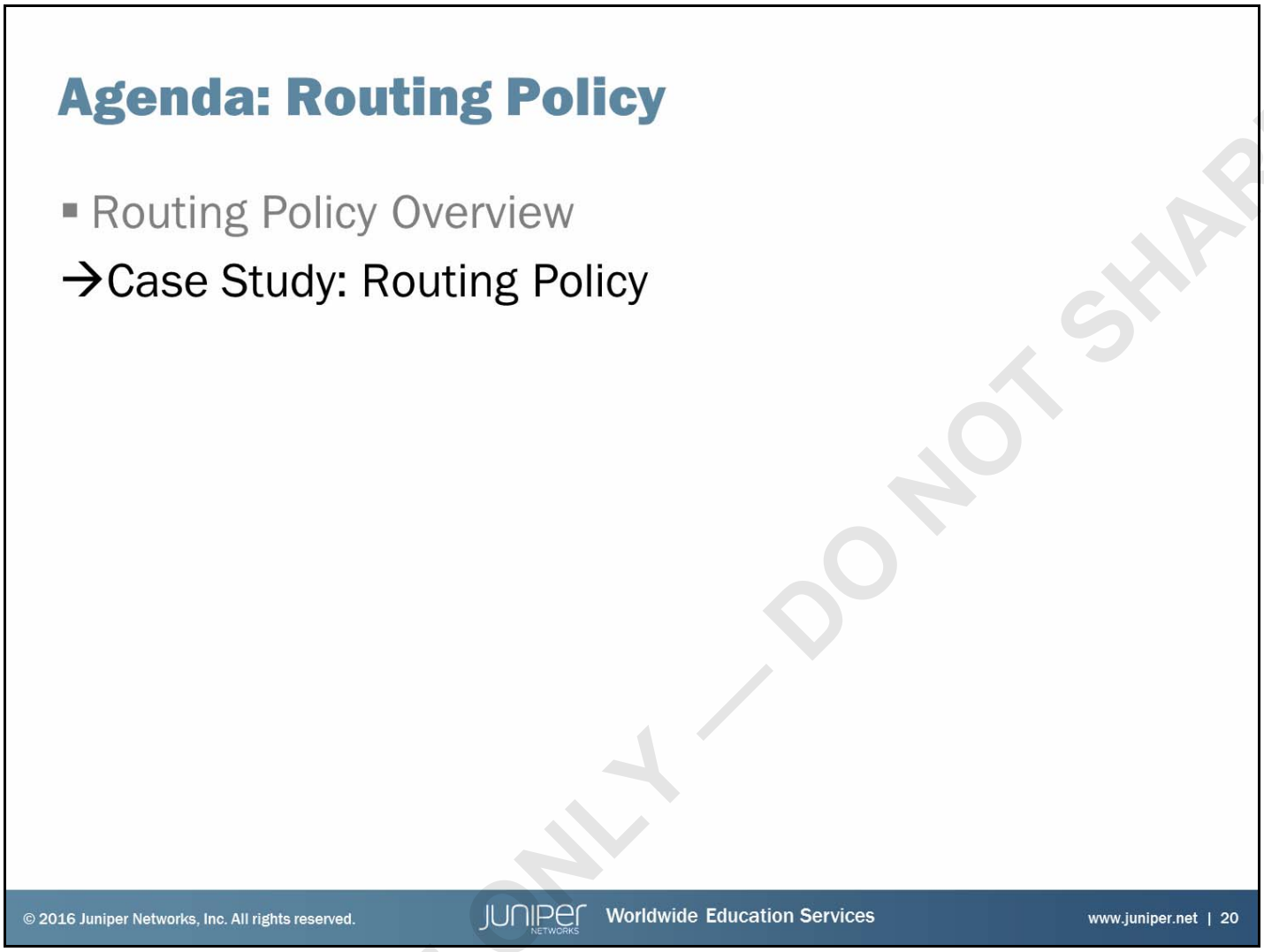
Policy Chaining (contd.)

As previously mentioned, individual policies can comprise multiple terms. Terms are individual match and action pairs that you can name numerically or symbolically.

The Junos OS lists terms sequentially from top to bottom and evaluates them in that manner. The software checks each term for its match criteria. When a match occurs, the software performs the associated action. If no match exists in the first term, the software checks the second term. If no match exists in the second term, the Junos OS checks the third term. This pattern repeats itself for all terms. If no match exists in the last term, the Junos OS checks the next applied policy and then, eventually, the default policy for the protocol.

When it finds a match within a term, the Junos OS takes the corresponding action. If the action is a terminating action, the processing of the terms and the applied policies stops—otherwise processing continues.

The Junos OS also supports flow-control actions that affect the flow of policy evaluation. The `next term` and `next policy` actions cause the Junos OS to evaluate the next term or next policy, respectively.

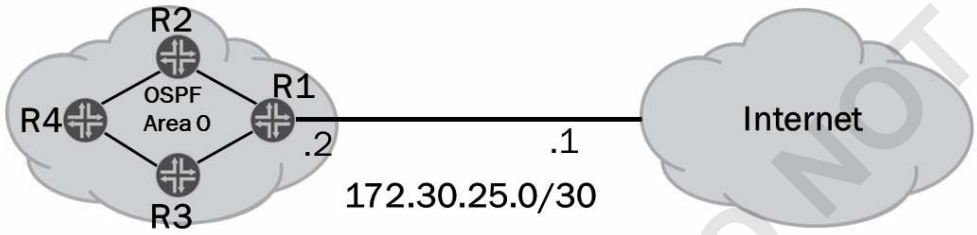


Case Study: Routing Policy

The slide highlights the topic we discuss next.

Case Study: Objective and Topology

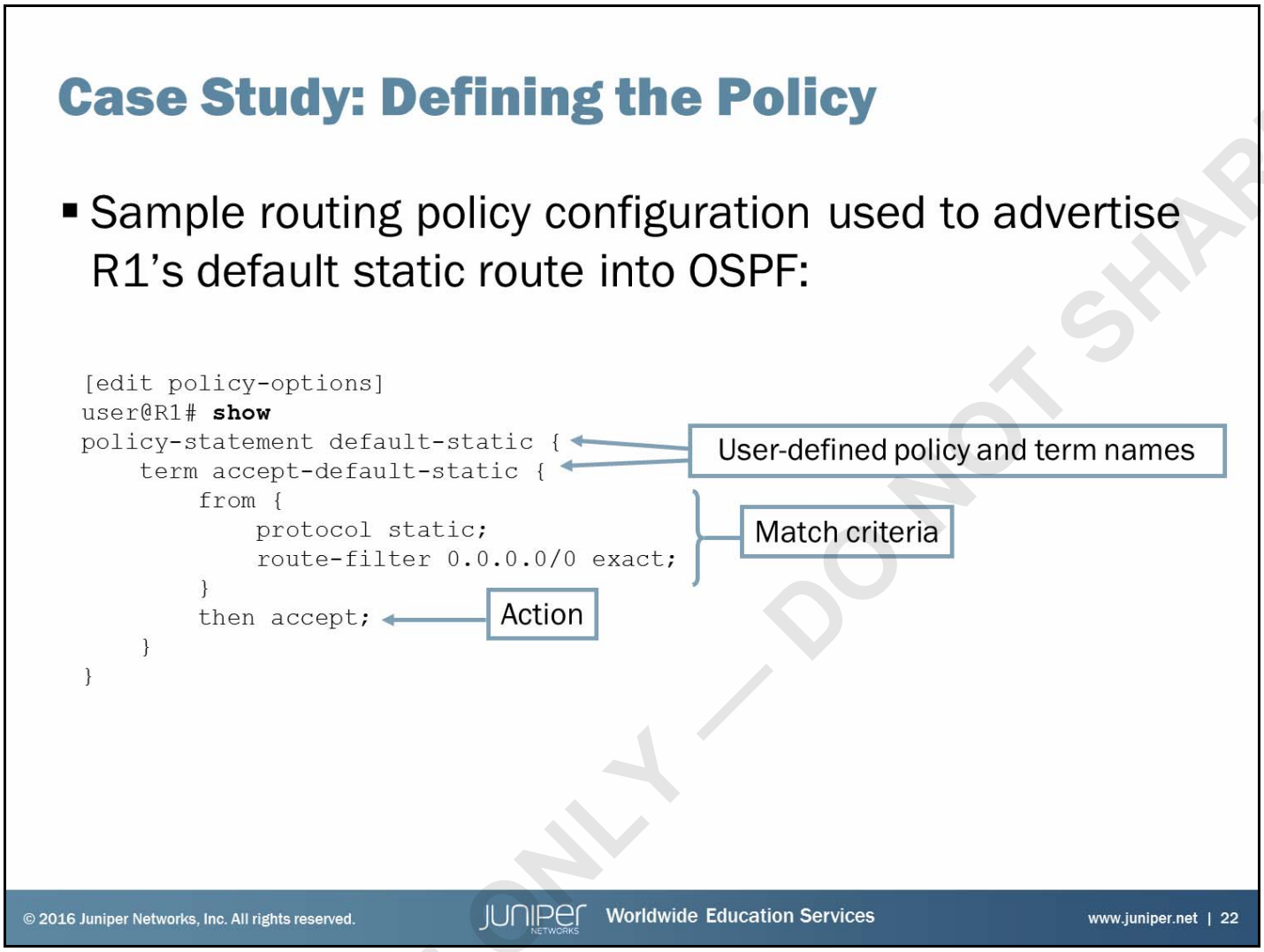
- Advertise the default static route defined on R1 into OSPF using routing policy:




```
user@R1> show route protocol static

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

0.0.0.0/0          *[Static/5] 00:00:44
                    > to 172.30.25.1 via ge-0/0/1.0
```


Case Study: Objective and Topology

The slide introduces a routing policy case study objective and topology. The stated objective for this case study is to use routing policy to advertise R1's default static route into OSPF.



Case Study: Defining the Policy

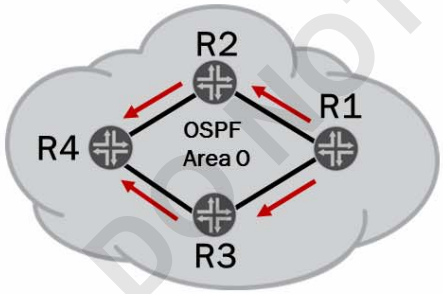
The slide shows a sample routing policy configuration that you can use to accomplish our previously stated objective.

Case Study: Applying the Policy

- Sample application of routing policy used to advertise R1's default static route into OSPF:

```
[edit protocols ospf]
user@R1# show
export default-static;
area 0.0.0.0 {
    interface ge-0/0/2.0;
    interface ge-0/0/3.0;
    interface lo0.0;
}
```

Export default static route from route table to OSPF



Note: Once you define routing policy and apply it, R1 floods an external LSA for the default static route to all OSPF routers in Area 0.

Case Study: Applying the Policy

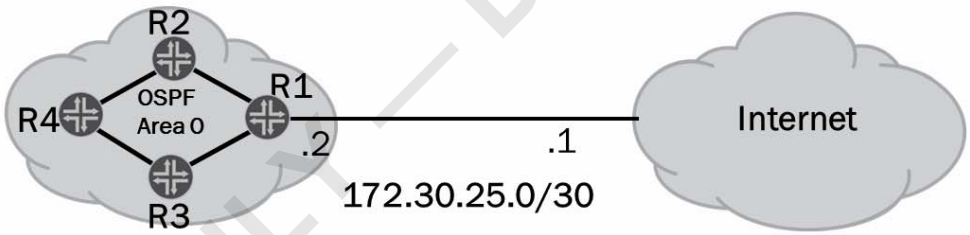
The slide shows the application of the sample routing policy defined on the previous slide. As noted on the slide, once the routing policy is applied and the new configuration is activated through a **commit**, R1 should begin advertising the default route as an external LSA to other routers within OSPF Area 0. We verify this advertisement on the next slide.

Case Study: Monitoring the Results

- Sample verification step to ensure the routing policy works as expected (capture is taken from R4):

```
user@R4> show route protocol ospf exact 0/0

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

0.0.0.0/0          *[OSPF/150] 00:03:33, metric 0, tag 0
                    > to 172.19.3.1 via ge-0/0/2.0
                      to 172.19.4.1 via ge-0/0/3.0
```

R4 installs external default OSPF route flooded by R1



Case Study: Monitoring the Results

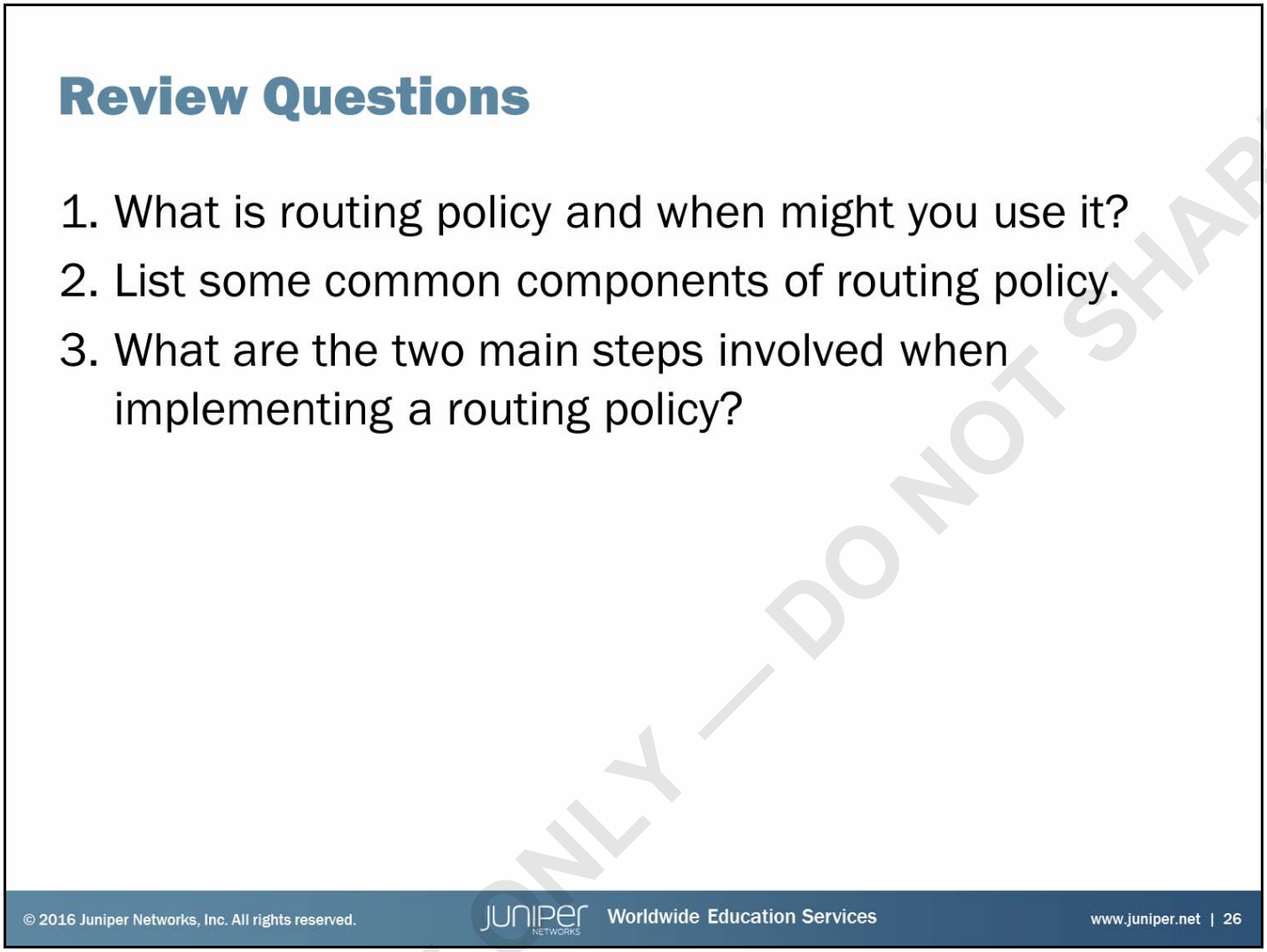
The slide shows a sample verification step that confirms the routing policy works as designed. Using the **`show route protocol ospf exact 0/0`** CLI command, we see R4 has added the default external OSPF route to its routing table.

Summary

- In this content, we:
  - Described the framework for routing policies
  - Explained how the software evaluates routing policies
  - Identified situations where you might use routing policies
  - Wrote and applied a routing policy

We Discussed:

- The framework of routing policy;
- Routing policy evaluation;
- Typical usage scenarios for routing policy;
- Configuration and application of a routing policy;

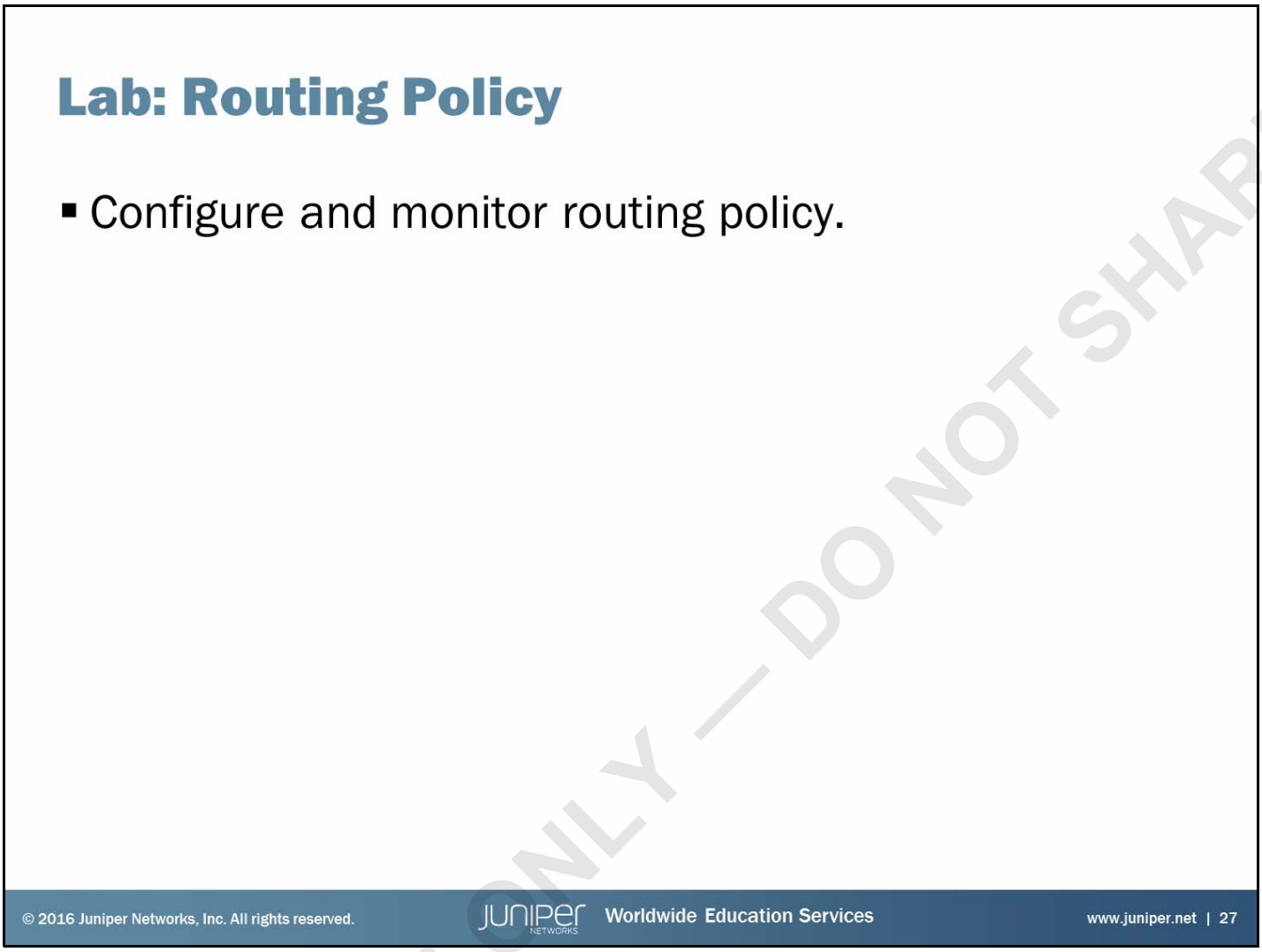


Review Questions

1.

2.

3.



Lab: Routing Policy

The slide provides the objective for this lab.

Answers to Review Questions

1.

Routing policy is used to control routing information within the routing table by choosing to accept, reject, or modify attributes for routes received and sent through dynamic protocols as well as for routes installed in the forwarding table.

2.

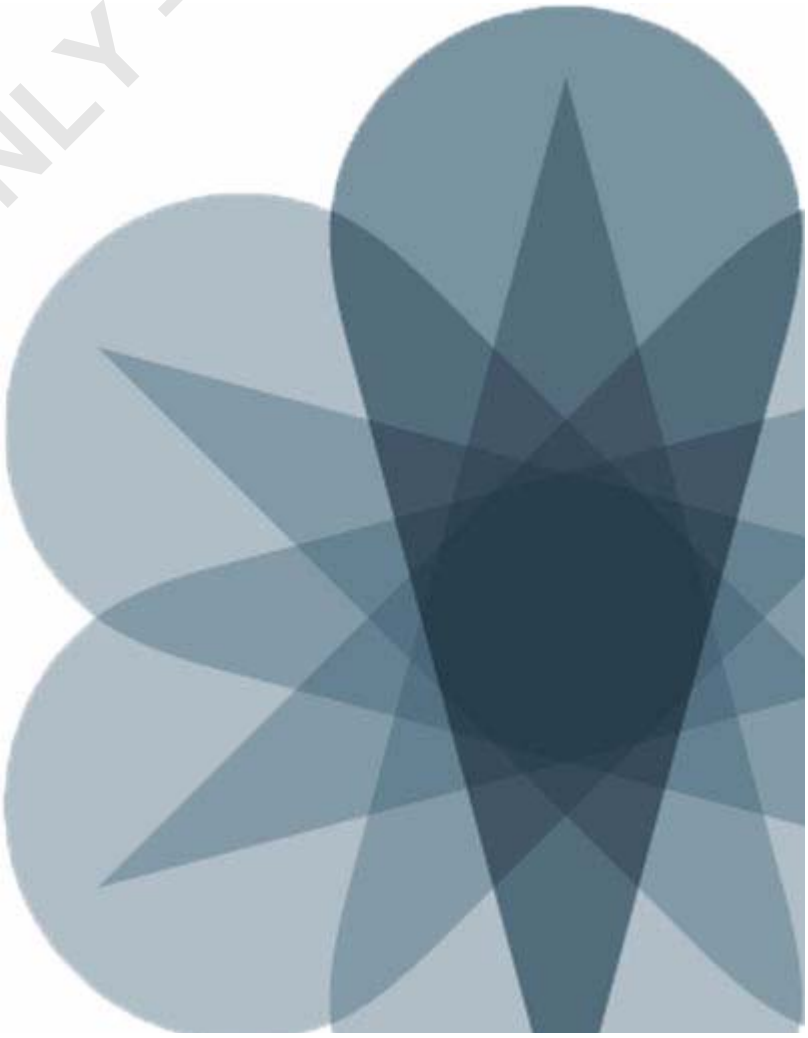
Routing policies use terms that consist of from and then statements. The `from` statements describe the match conditions that must be met before taking the defined action. The `then` statement describes the action the system should take if a packet or route meets the defined match conditions.

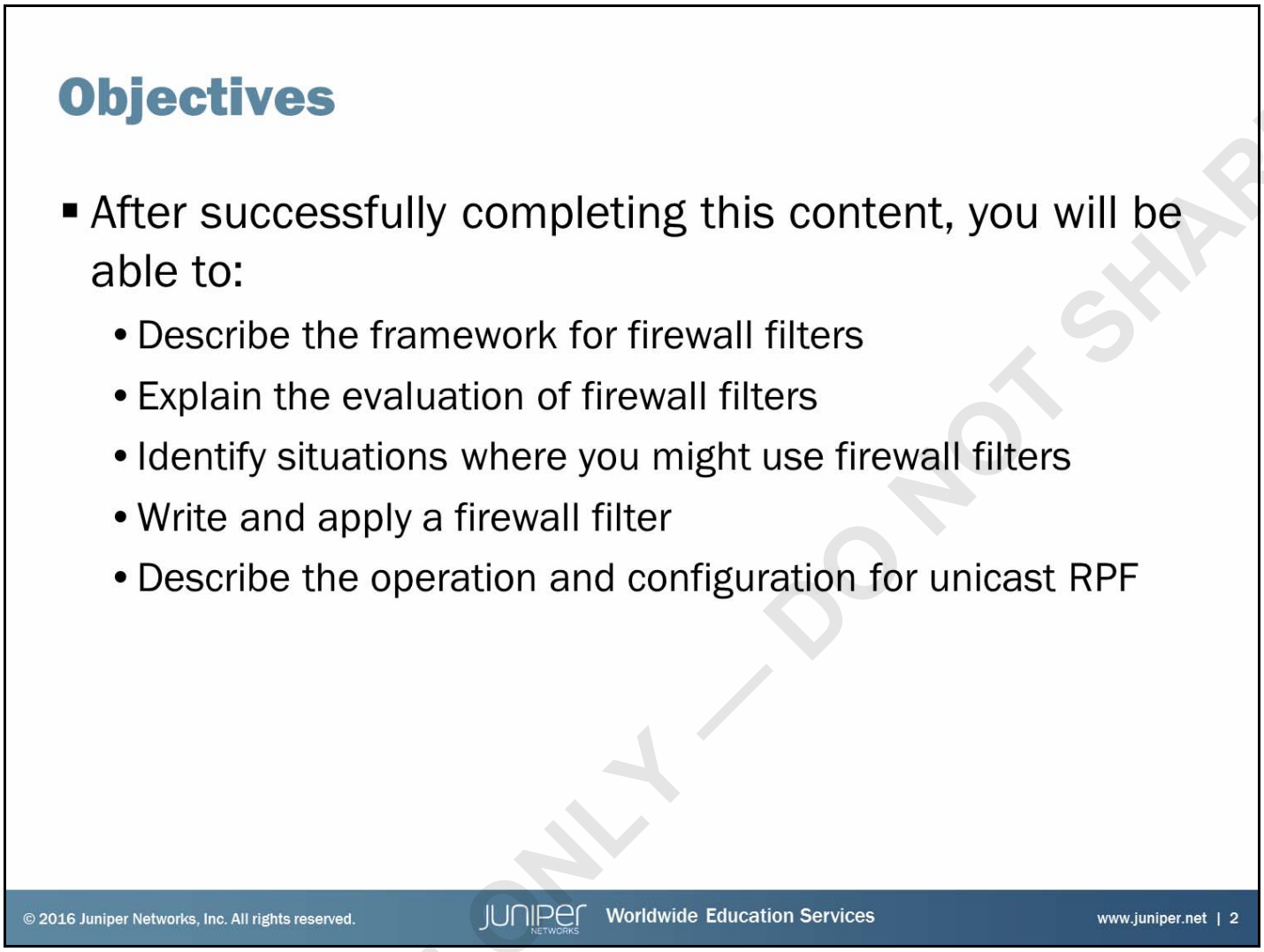
3.

The two main steps involved when implementing policies are definition and application. You must first define the policy or filter under the respective hierarchy level. Once you define the policy or filter, you must then apply it.

Junos Routing Essentials

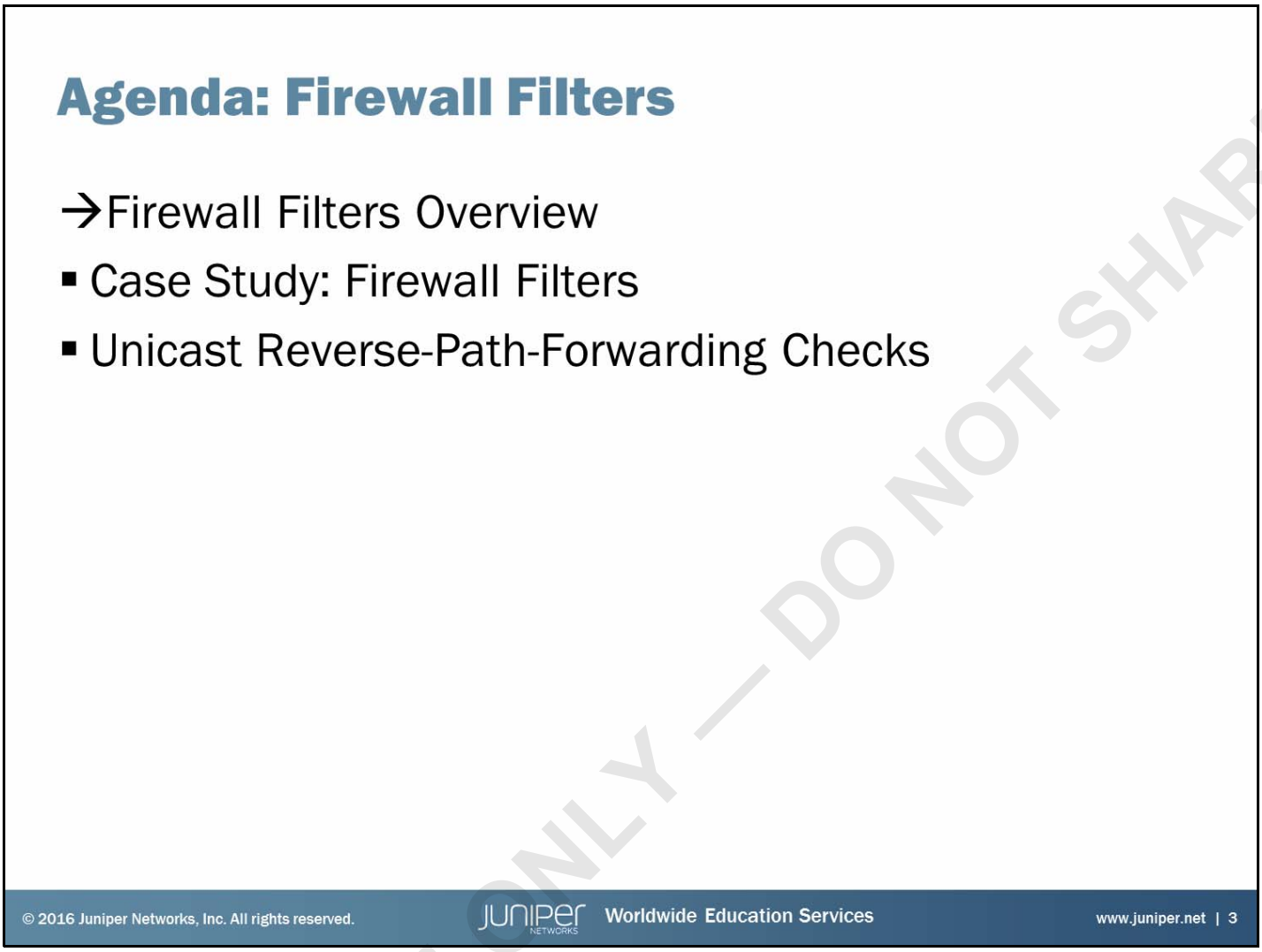
Chapter 4: Firewall Filters





We Will Discuss:

- The framework of firewall filters;
- Firewall filter evaluation;
- Typical usage scenarios for firewall filters;
- Configuration and application of firewall filters; and
- Unicast reverse path forwarding (RPF).

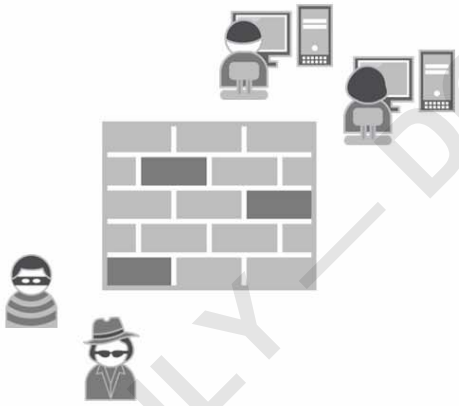


Firewall Filters Overview

The slide lists the topics we will discuss. We discuss the highlighted topic first.

What Is a Firewall Filter?

- Firewall filters control the traffic entering and leaving a networking device in a stateless fashion:
  - Processes every packet independently
  - Used to filter and monitor network traffic

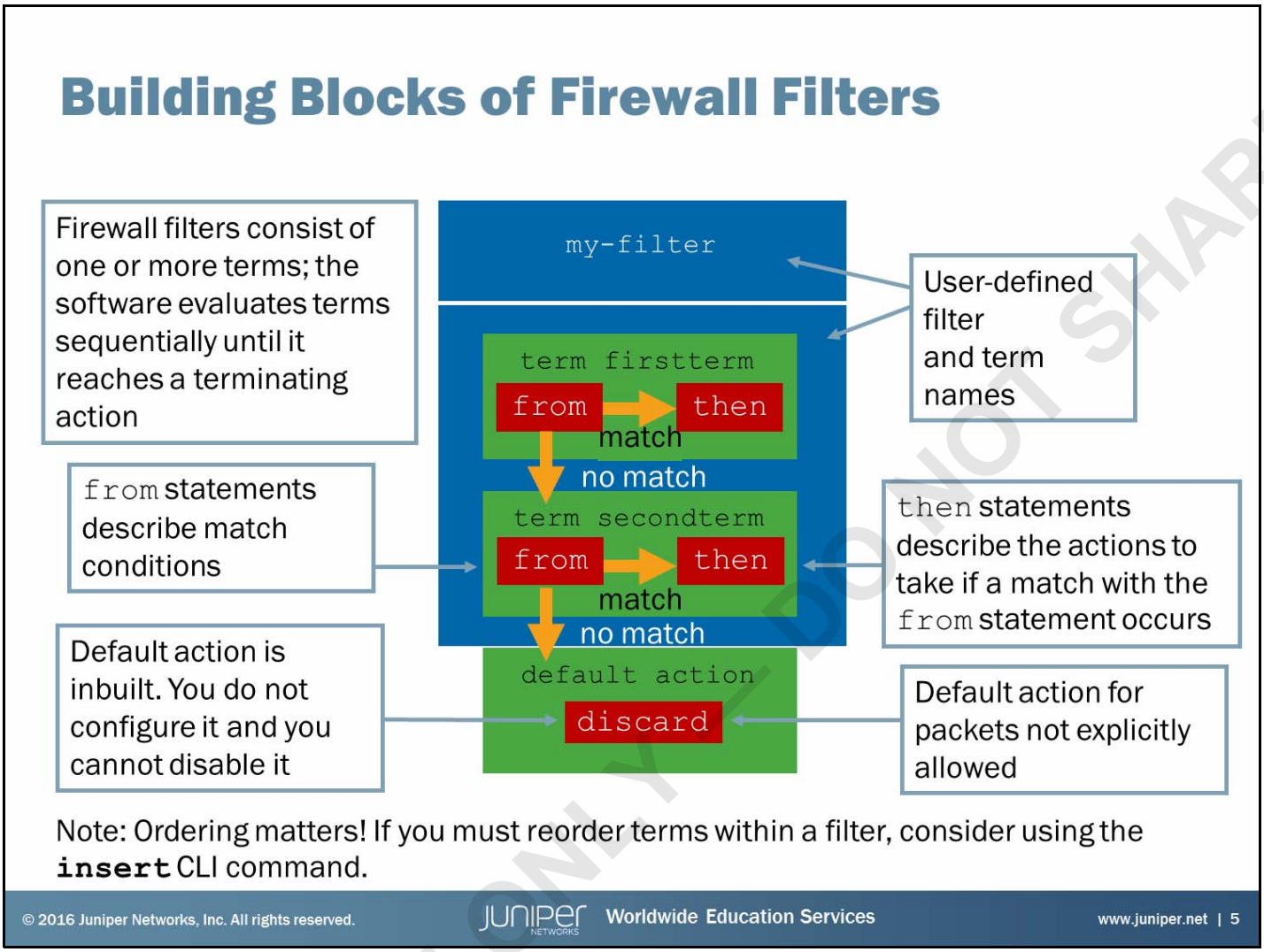


Firewall Filters

Firewall filters are often referred to as access control lists (ACLs) by other vendors. The Junos firewall filters are stateless in nature, and the software primarily uses them to control traffic passing through a Junos device.

Stateless firewall filters examine each packet individually. Thus, unlike a stateful firewall that tracks connections and allows you to specify an action to take 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. Because the system 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 automatically permit bidirectional communications for this connection.

You can use firewall filters to restrict certain types of traffic from passing into and out of your network. You can also use firewall filters to perform monitoring tasks that help you formulate an effective security strategy for your environment.

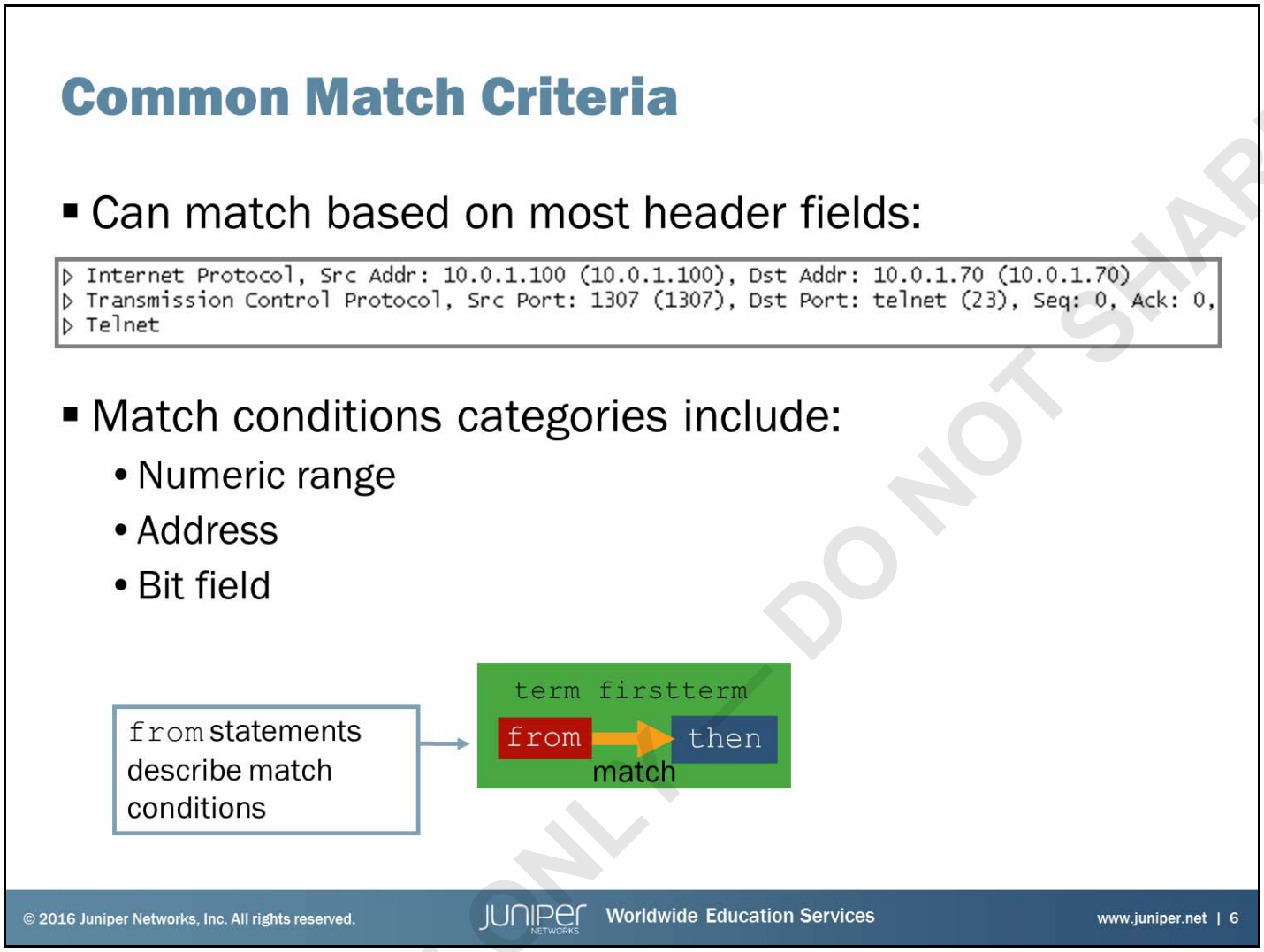


Building Blocks of Firewall Filters

Although routing policies and firewall filters serve different purposes and have different match and action conditions, they both have a common structure.

As with routing policy, the fundamental building block of a firewall filter is the *term*. A term contains zero or more match conditions and one or more actions. If all the match conditions are true, the Junos OS takes the specified action within the term. If no match conditions are specified, all traffic matches the firewall filter term and is subjected to the stated action. You use a filter to group together multiple terms and establish the order in which the system evaluates the terms. The Junos firewall filters require at least one term.

Firewall filters always include an invisible default term that discards all packets that the configuration does not explicitly permit through the defined terms. When implementing firewall filters, keep in mind that the order of the terms is important and can impact the results.



Common Match Criteria

You specify the criteria to use for matching 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 because of the way firewall filters are processed.

When you specify a header field, the Junos OS 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 system both check that the packet was a TCP packet and whether the TCP ACK flag was set.

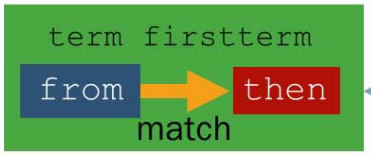
The stateless nature of firewall filters can affect the information available in the processing of fragmented packets. Processing fragments is more complicated with stateless firewall filters than with a stateful firewall filter. The first fragment should have all the Layer 4 headers but subsequent fragments will not. Additionally, attempting to check Layer 4 headers in fragments produces unpredictable results. As we explained previously, the Junos OS still attempts to evaluate the Layer 4 headers but 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. Several text synonyms exist that function as match conditions. A text synonym match condition is equivalent to one or more match conditions. (For example, the `tcp-established` match condition is a text synonym for the `tcp-flag ack` or the `tcp-flag rst` match conditions.)

Firewall Filter Actions

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



then statements describe the actions to take if a match with the from statement occurs

The software discards all traffic not allowed by a matched term!

Common Actions

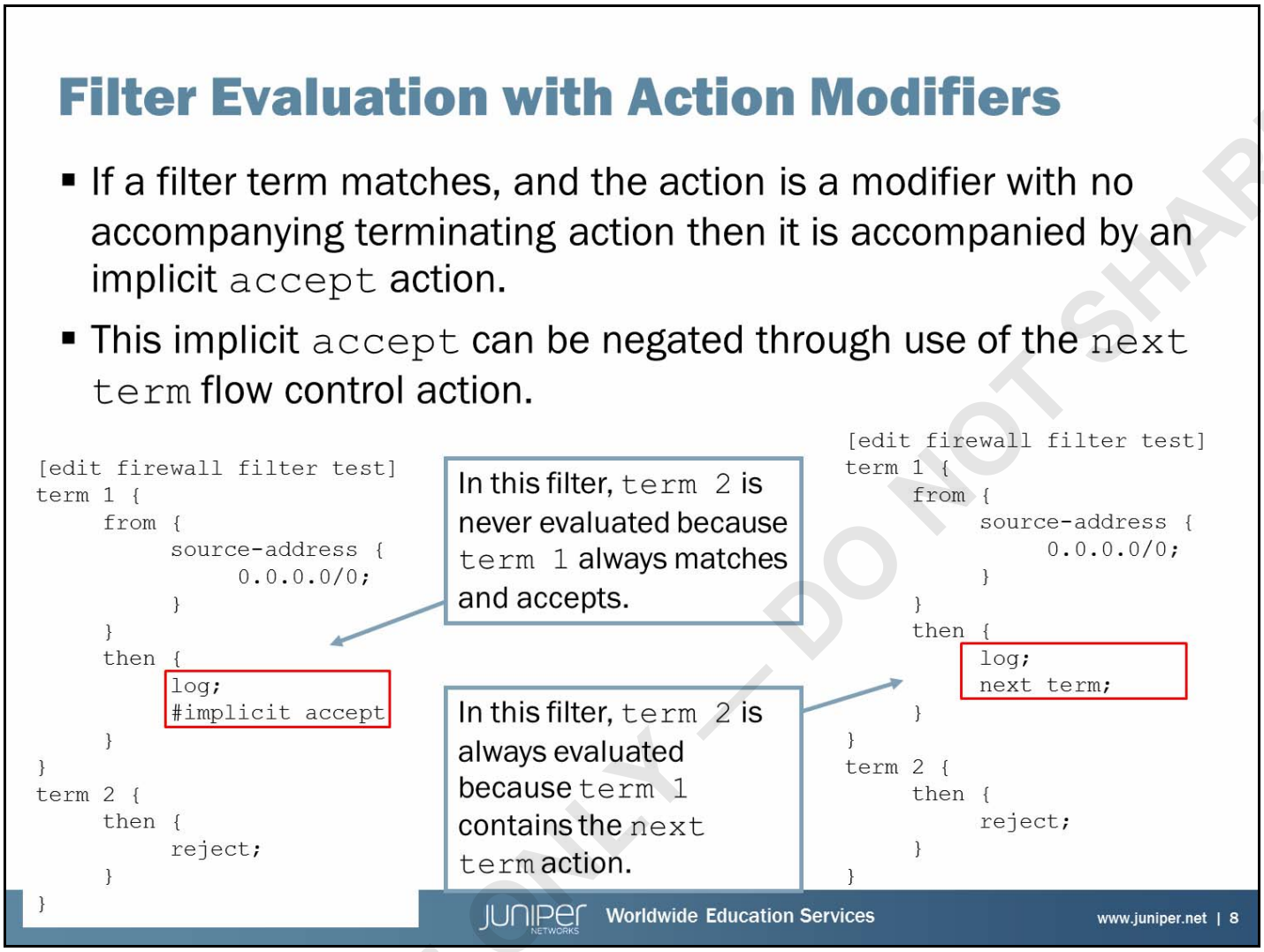
You specify actions in the *then* clause of a term. Common firewall filter actions include terminating actions, flow control, and action modifiers.

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

Other common firewall filter actions affect the flow of policy evaluation. The `next term` action cause the Junos OS to evaluate the next term. The `next term` action is useful if you want to set a policer or DiffServ code point (DSCP) value and still have the traffic evaluated by the rest of the filter. No `next filter` action exists for firewall filters.

You can specify one or more action modifiers with any terminating or flow-control action. If you specify an action modifier, but do not specify a terminating action, the system 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. The policer action modifier allows you to invoke a traffic policer, which we cover later in this chapter.

Note that when you apply a firewall filter then traffic that does not match a term that has an explicit or implicit `allow` action will be discarded even if it does not match any term with a `discard` action.



Filter Evaluation when a Term Matches, and no Terminating Action is Specified.

Sometimes the action specified in a term is not one of the terminating actions, `accept`, `discard`, or `reject`. In this case, the default behavior is that an implicit accept is applied and filter evaluation terminates.

If you want to apply an action modifier (`count`, `log`, `syslog`, `forwarding-class`, `loss-priority`) to a packet and then continue to evaluate terms to decide whether to accept, `discard` or reject, you need to use the `next term` flow control action.

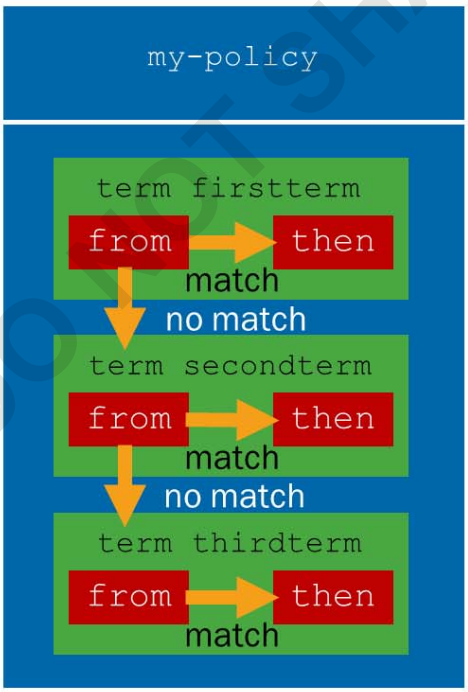
Implementing Firewall Filters (1 of 2)

- Define firewall filters based on protocol family under the [edit firewall] hierarchy level:

```
[edit firewall family inet]
user@router# show
filter filter-in {
    term block-some-packets {
        from {
            source-address {
                10.10.10.0/24;
            }
        }
        then {
            count spoof-in;
            discard;
        }
    }
    term accept-others {
        then accept;
    }
}
...
```

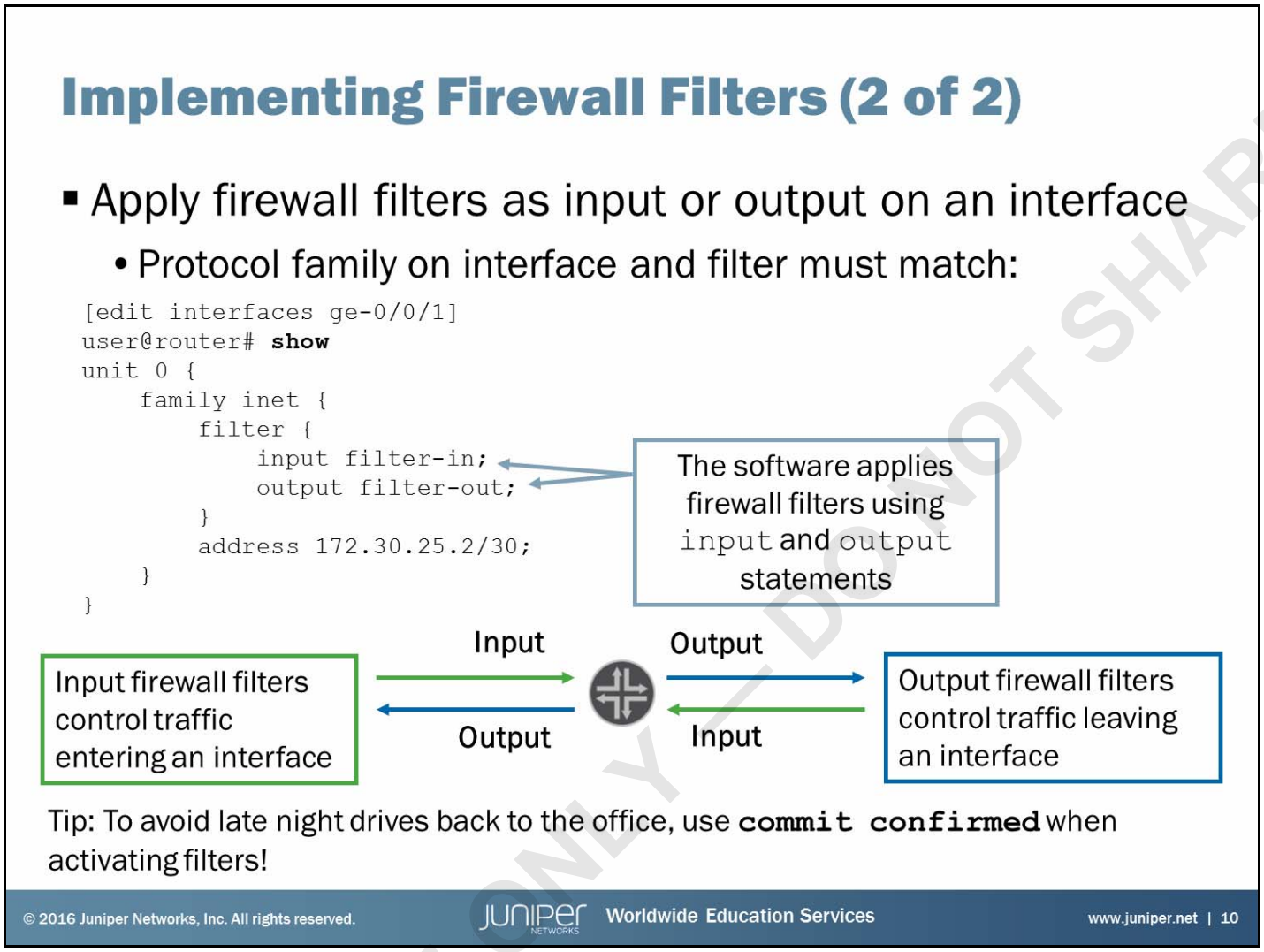
The software applies family inet filters only to interfaces running IPv4

If discard is not present then packets are accepted



Defining a Firewall Filter

Implementing a firewall filter requires two distinct steps. The first step is to define the firewall filter. In the Junos OS, you define firewall filters under the [edit firewall] hierarchy level. Because the Junos OS supports multiple protocol families, you must navigate down to the appropriate family hierarchy level. The slide illustrates a sample IPv4 firewall filter defined under the [edit firewall family inet] hierarchy level. The software supports other protocol families. Check your product specific documentation for details.



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. Additionally, you can apply them to the lo0 logical interfaces to filter traffic destined for the system.

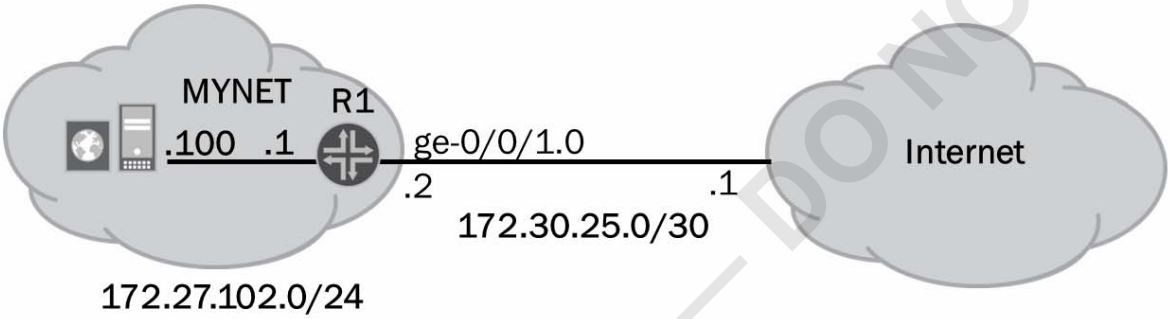
You apply IPv4 firewall filters to interfaces in the `[edit interfaces` *`interface-name`* `unit` *`unit-number`* `family inet filter]` hierarchy. To apply a single input or output filter, use the **`input`** ***`filter-name`*** or **`output`** ***`filter-name`*** configuration options. You can specify both input and output filters on the same interface. You cannot, however, apply an IPv6 firewall filter to an IPv4 interface. In other words, the protocol family for the firewall filter and the interface must match.

You can also apply multiple filters to filter traffic using the **`input-list`** or **`output-list`** configuration options in the `[edit interfaces` *`interface-name`* `unit` *`unit-number`* `family inet filter]` hierarchy.

Any time you perform configuration changes from a remote location, you should use the **`commit confirmed`** option when activating a new configuration. This habit might prove to be especially helpful when working with firewall filters and might save you from a late night trip back to the office!

Test Your Knowledge (1 of 2)

- Apply a filter on R1's ge-0/0/1.0 interface to allow HTTP traffic to 172.27.102.100
  - Should the filter be applied as an input or output filter?



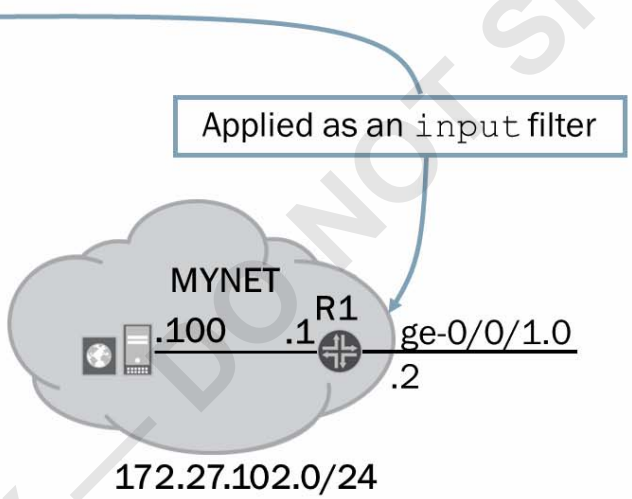
Test Your Knowledge: Part 1

The slide tests your understanding of how firewall filters are applied. Because the objective is to filter inbound HTTP traffic on the ge-0/0/1.0 interface, you should apply the appropriate filter to the ge-0/0/1.0 interface as an input filter. We look at a sample configuration that helps achieve this objective on the next slide.

Test Your Knowledge (2 of 2)

- Which inbound traffic does the router permit?

```
filter web-server {
    term allow-web-traffic {
        from {
            destination-address {
                172.27.102.100/32;
            }
            protocol tcp;
            port http;
        }
        then accept;
    }
    term deny-other-web-traffic {
        from {
            protocol tcp;
            port http;
        }
        then {
            discard;
        }
    }
}
```

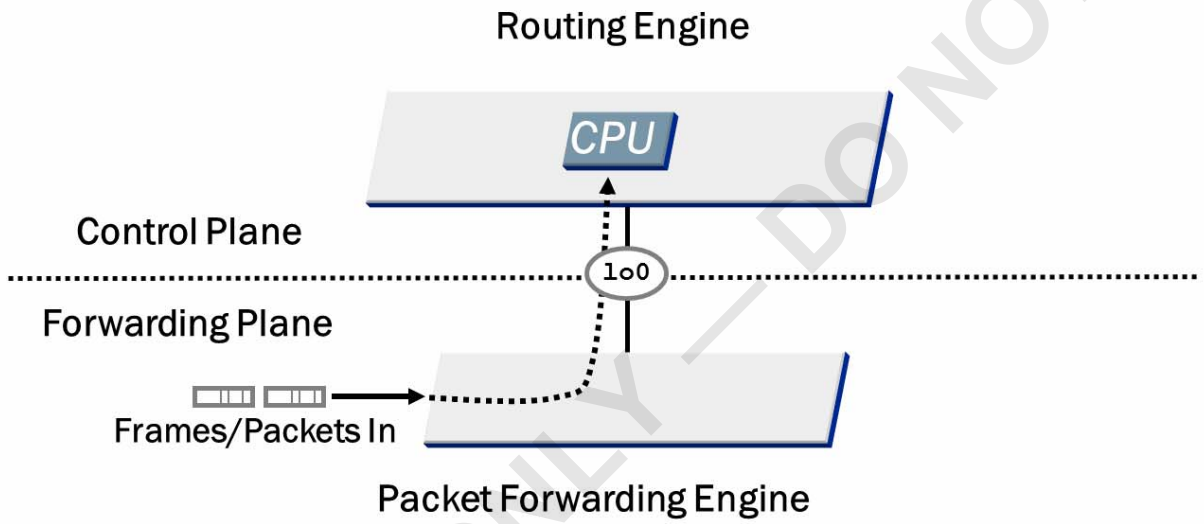


Test Your Knowledge: Part 2

Based on the sample configuration and, specifically, the `allow-web-traffic` term, the software permits inbound HTTP traffic to address 172.27.102.100/32 only. Note that the `deny-other-web-traffic` term specifically denies all other HTTP traffic. This denial is not strictly required because the default action for all traffic not explicitly allowed is `discard`.

Filtering Local Traffic (1 of 2)

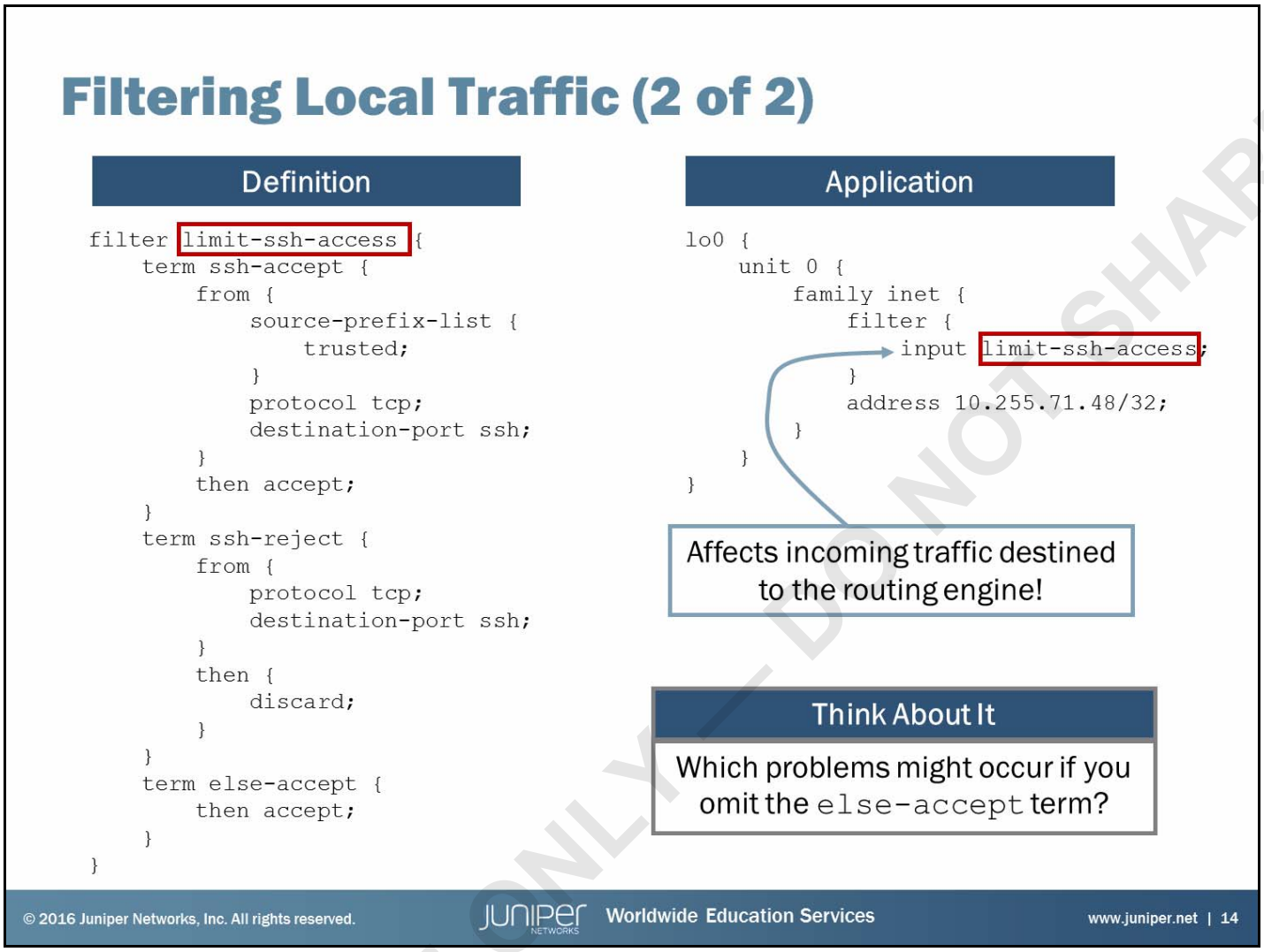
- Apply filters to lo0 interface to filter local traffic
  - Filter must account for routing and management protocols



Filtering Local Traffic: Part 1

Transit firewall filters act on packets flowing from one interface to another interface within a device running the Junos OS. These filters can protect sites from unauthorized access and other threats. But what about protecting the system from unauthorized management access and other harmful effects? This concern is the idea behind applying a firewall filter to protect the Routing Engine (RE). The Packet Forwarding Engine (PFE) applies these filters before traffic ever reaches the control plane.

The software does not create automatic *holes* 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 implicit silent discard, which discards all packets not explicitly allowed through a defined term, has been known to cause undesirable effects!



Filtering Local Traffic: Part 2

The slide shows a basic firewall filter named `limit-ssh-access`, which controls management access to the local system. The software applies the filter to the lo0 interface as an input filter and evaluates all incoming traffic destined to the RE.

The `limit-ssh-access` filter includes three distinct terms. The first term, named `ssh-accept`, permits all SSH traffic from a defined prefix list named `trusted`. The `trusted` prefix list follows:

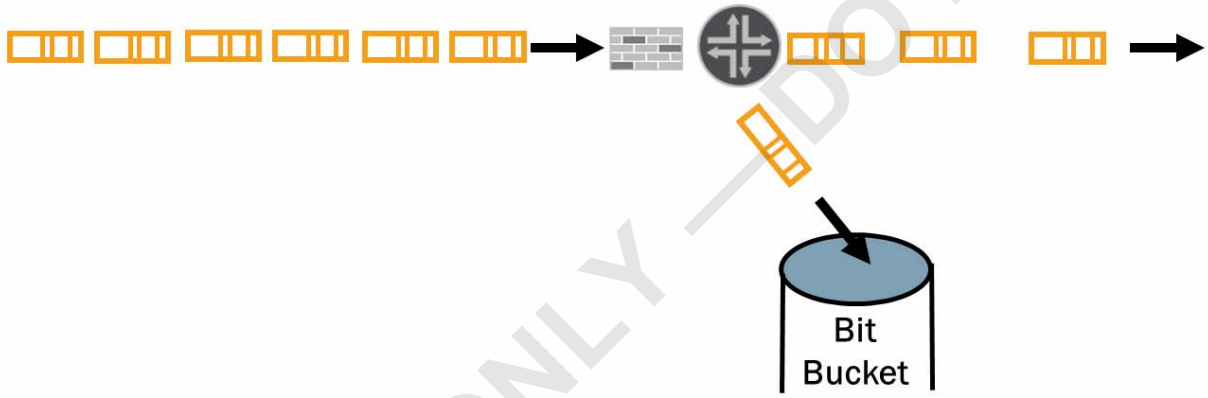
```
[edit policy-options]
user@router# show
prefix-list trusted {
    172.27.102.0/24;
}
```

A second term named `ssh-reject` discards all other SSH traffic not sourced from the `trusted` prefix list. A third term permits all other traffic. Your firewall filters must account for all management and protocol traffic destined to the control plane. In our example, we have accomplished this accounting through the use of the `else-accept` term.

If the `else-accept` term was not included in the filter, the software would discard all control and management traffic not specifically allowed in the other terms. This process could cause quite a disturbance in environments that use dynamic routing protocols, such as OSPF and BGP, as well as management protocols like SNMP or NTP.

Policing

- Policing (rate-limiting) enables you to limit the amount of traffic that passes into or out of an interface:
  - Works with firewall filters to thwart DoS attacks
    - Common actions include discard and setting loss-priority level
  - Uses average bandwidth and maximum burst size



Policing

In addition to dropping or accepting packets, firewall filters can also 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 forth, 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, the Junos OS 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 virtual private network (VPN) traffic as well as the MPLS and IPv6 families, and they operate without the need for a calling firewall filter. Actual policer support might vary between Junos devices. Refer to the documentation for your specific product for support information.

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 defines the total number of bytes the system allows in bursts of data that exceed the given bandwidth limit.

Continued on the next page.

Policing (contd.)

The preferred method for determining the maximum burst size is to multiply the speed of the interface by the amount of time bursts that you want to allow 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 per sec) x allowable burst time (5/1000s)

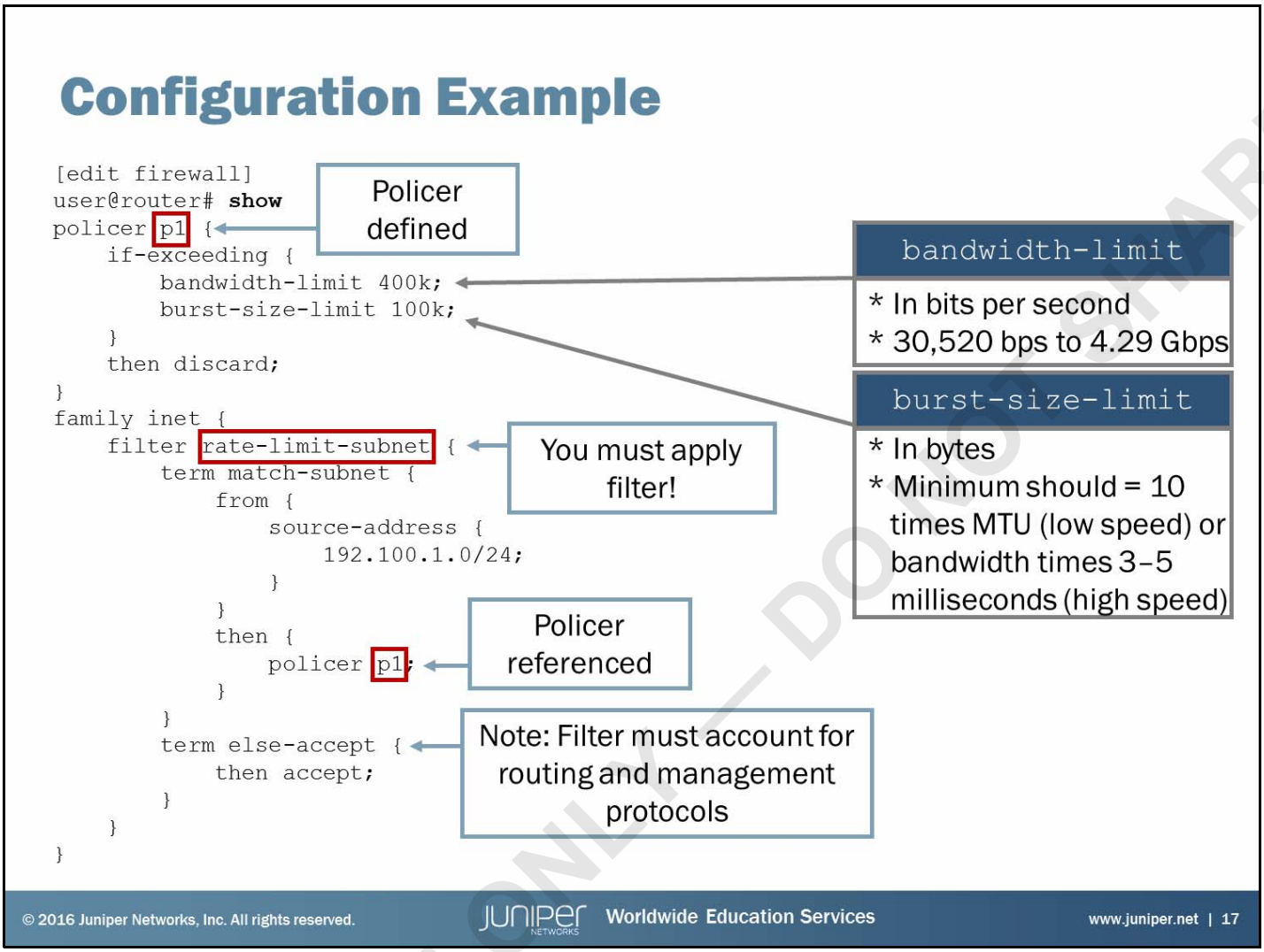
This calculation yields a burst size of 500,000 bits. You can divide this number by 8 to convert it to bytes, which gives you a burst size of 62500 bytes.

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

When a packet matches a term that has a policer in the *then* clause, the system first determines if the packet exceeds the policer. If the packet does not exceed the policer, the system performs the actions in the firewall filter's *then* clause as if you left the policer out of the configuration. If the packet does exceed the policer, the system takes the actions in the policer's *then* clause. If the policer's *then* clause does not result in the software discarding the packet, the system takes the remainder of the actions in the firewall filter's *then* clause. In cases where the specified rate limit has been exceeded and both the policer's *then* clause and the firewall filter's *then* clause define action modifiers, the system 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, whereas it places traffic that conforms to these limits 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;
                }
            }
        }
    }
}
```

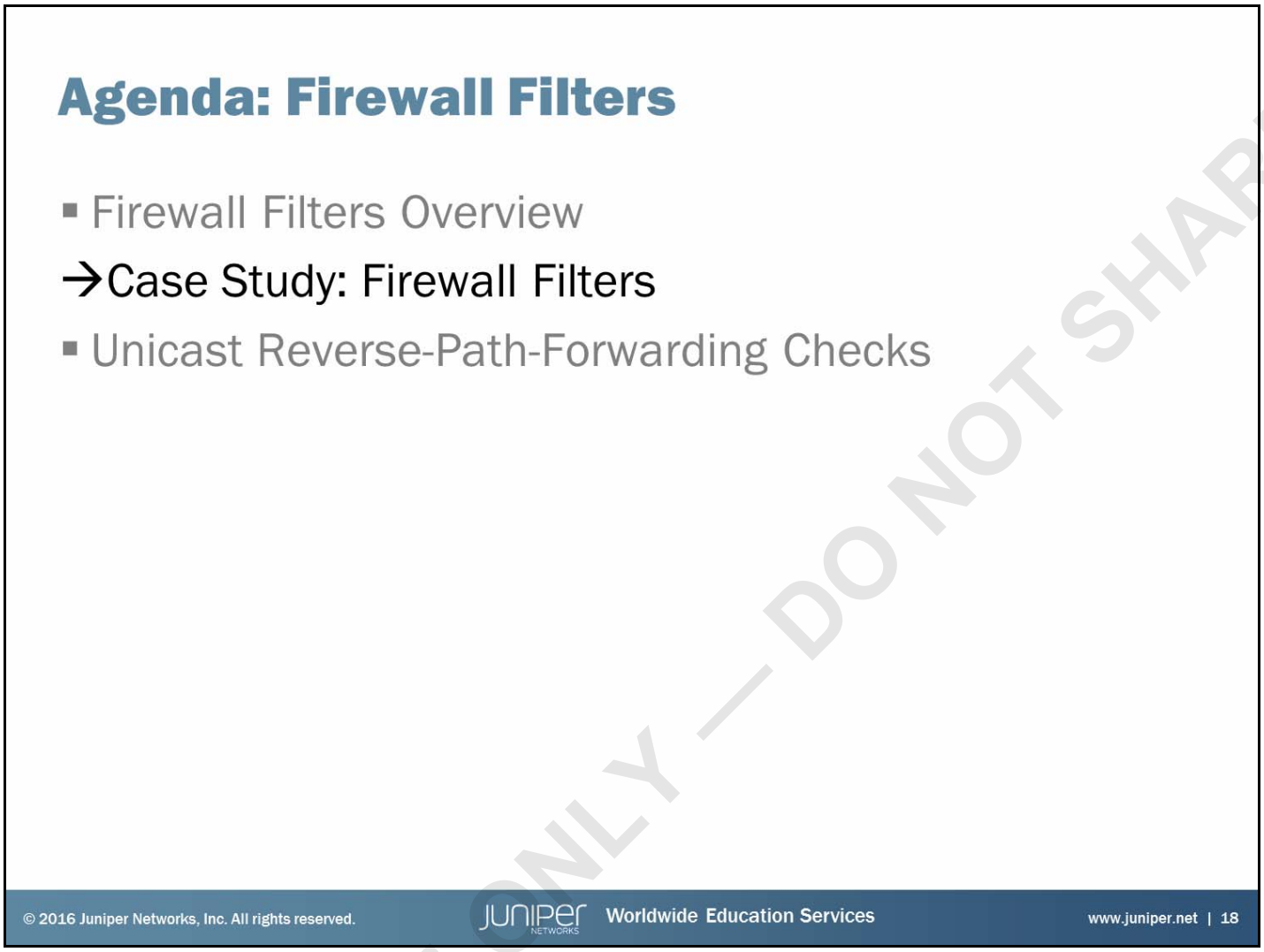


Policing Example

In the example on the slide, we define a policer named *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, devices running the Junos OS treat each invocation of the policer separately, and track statistics separately for each term that references the policer. You can think of the policer definition as simply defining a set of parameters that we can choose to reference in any firewall filter term.

The filter *rate-limit-subnet* polices traffic from the specified subnet. If the traffic sourced from the specified subnet exceeds the limits, the system discards it. If the traffic does *not* exceed the specified limits, the system accepts it.

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



Case Study: Firewall Filters

The slide highlights the topic we discuss next.

Case Study: Objectives and Topology

Case Study Objectives

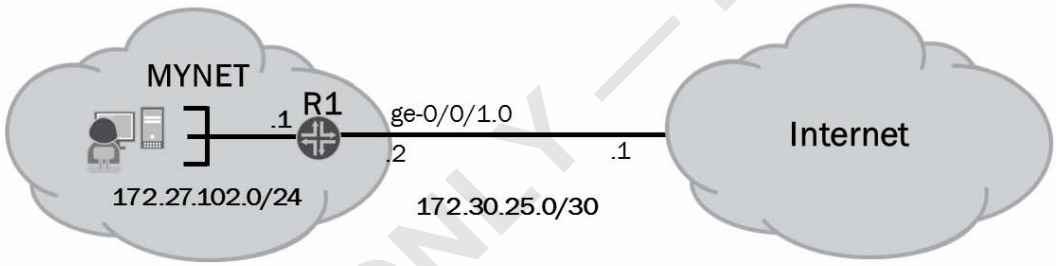
Outbound:

1. Discard and log all outbound traffic with any source address not belonging to 172.27.102.0/24.
2. Allow and count all other traffic.

Inbound:

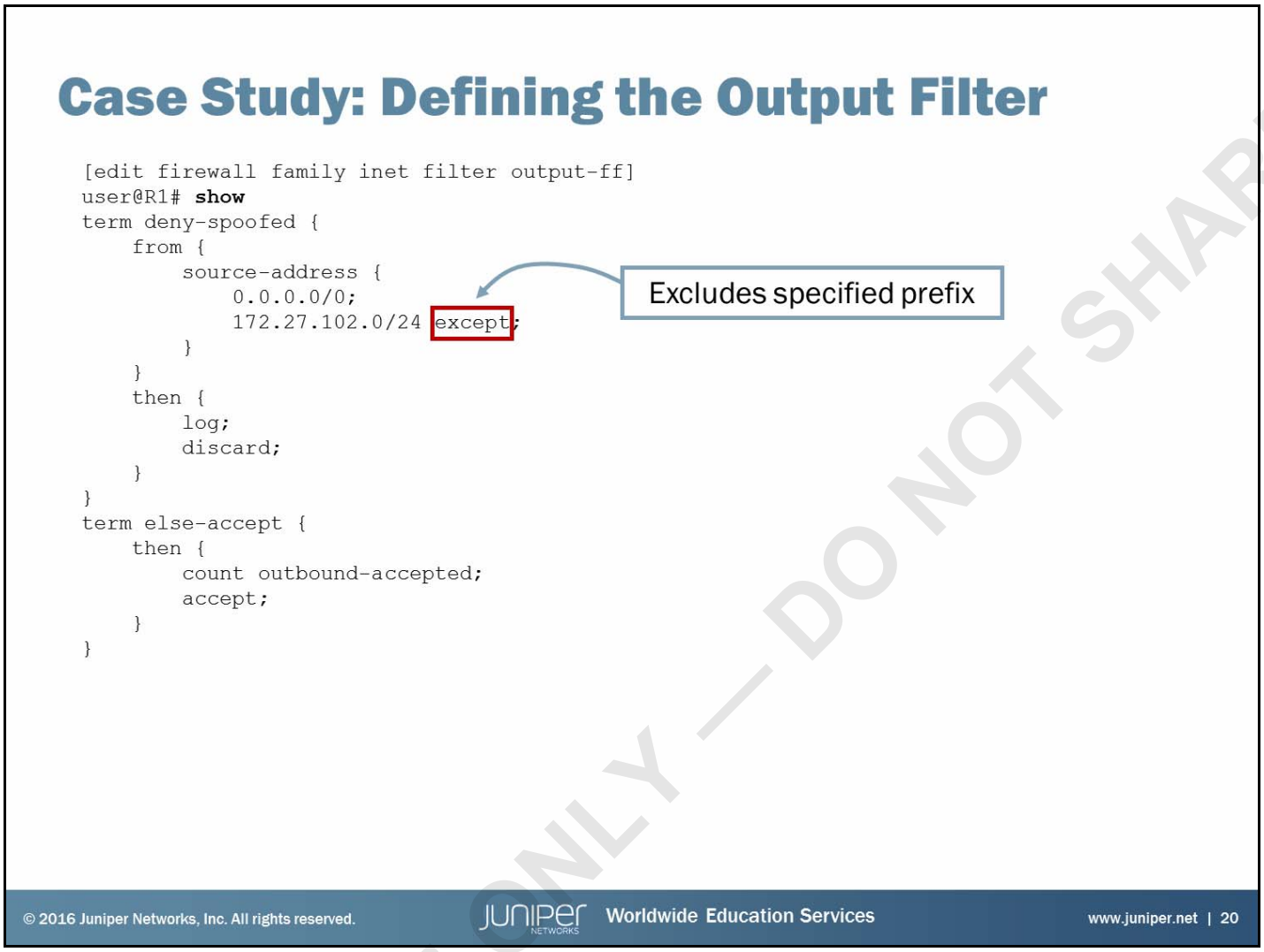
1. Discard and log all inbound traffic with a source address belonging to 172.27.102.0/24.
2. Allow all return traffic from the Internet for TCP sessions initiated from MYNET.
3. Allow ICMP traffic that includes echo replies, time exceeded, and destination unreachable messages.
4. Discard and count all other traffic.

Note: Apply all filters to the ge-0/0/1.0 interface



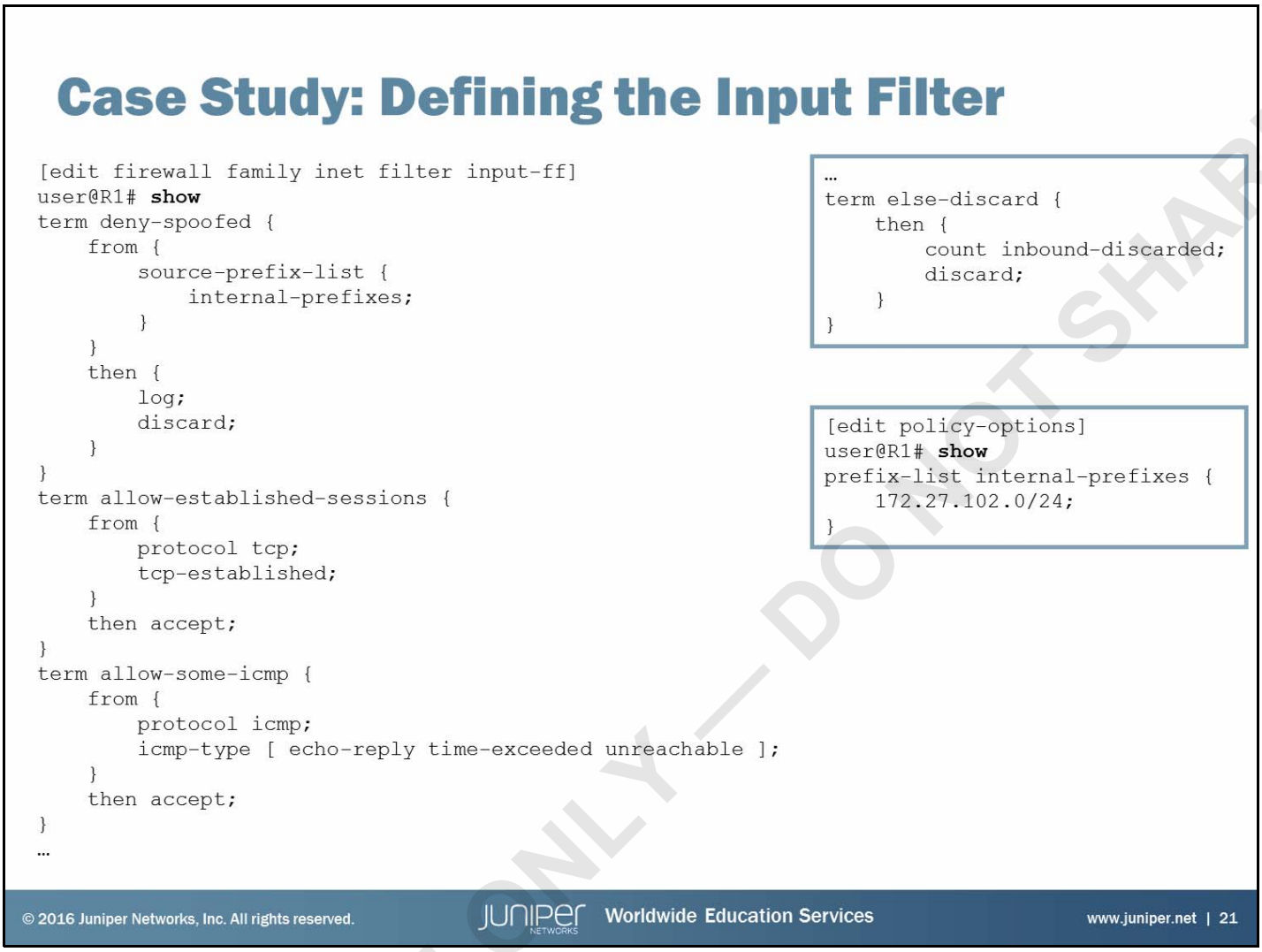
Case Study: Objectives and Topology

The slide introduces the objectives and topology for a firewall filter case study.



Case Study: Defining the Output Filter

The slide illustrates the sample output filter used to meet part of the objectives.

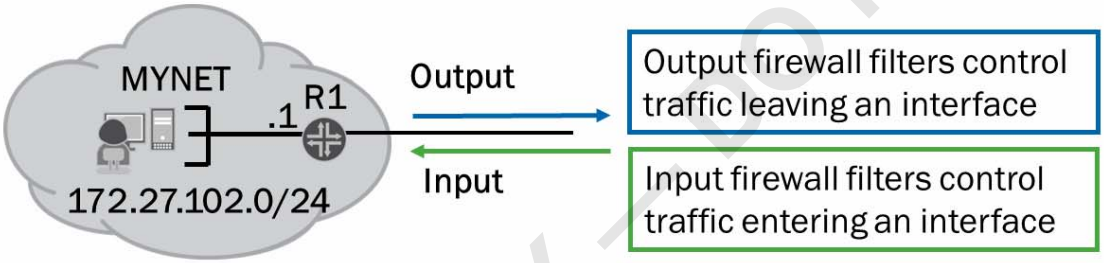


Case Study: Defining the Input Filter

The slide illustrates the sample input filter used to meet the remainder of the objectives.

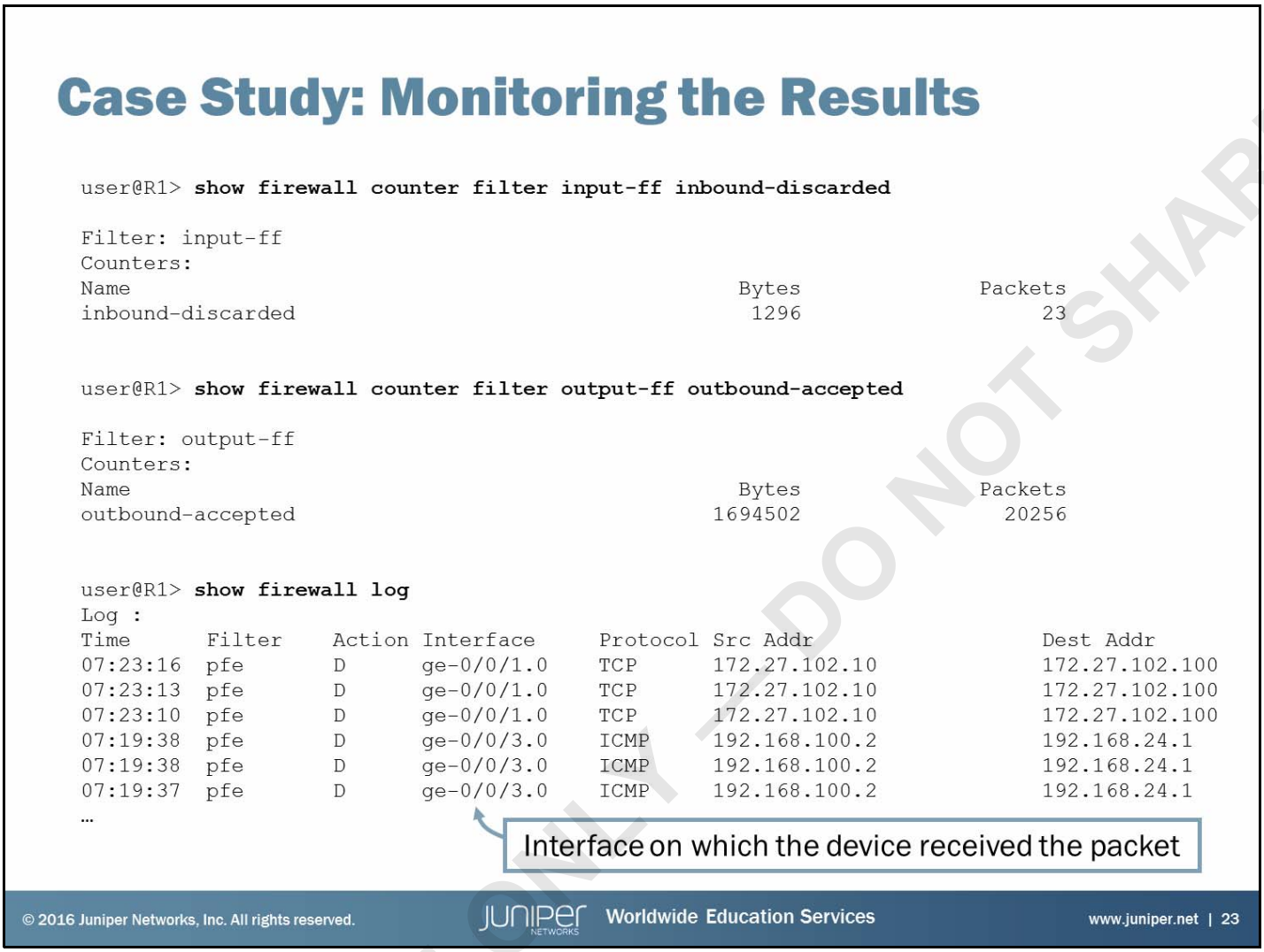
Case Study: Applying the Filters

```
[edit interfaces ge-0/0/1]
user@R1# show
unit 0 {
    family inet {
        filter {
            input input-ff;
            output output-ff;
        }
        address 172.30.25.2/30;
    }
}
```



Case Study: Applying the Filters

The slide displays the application of the firewall filters.



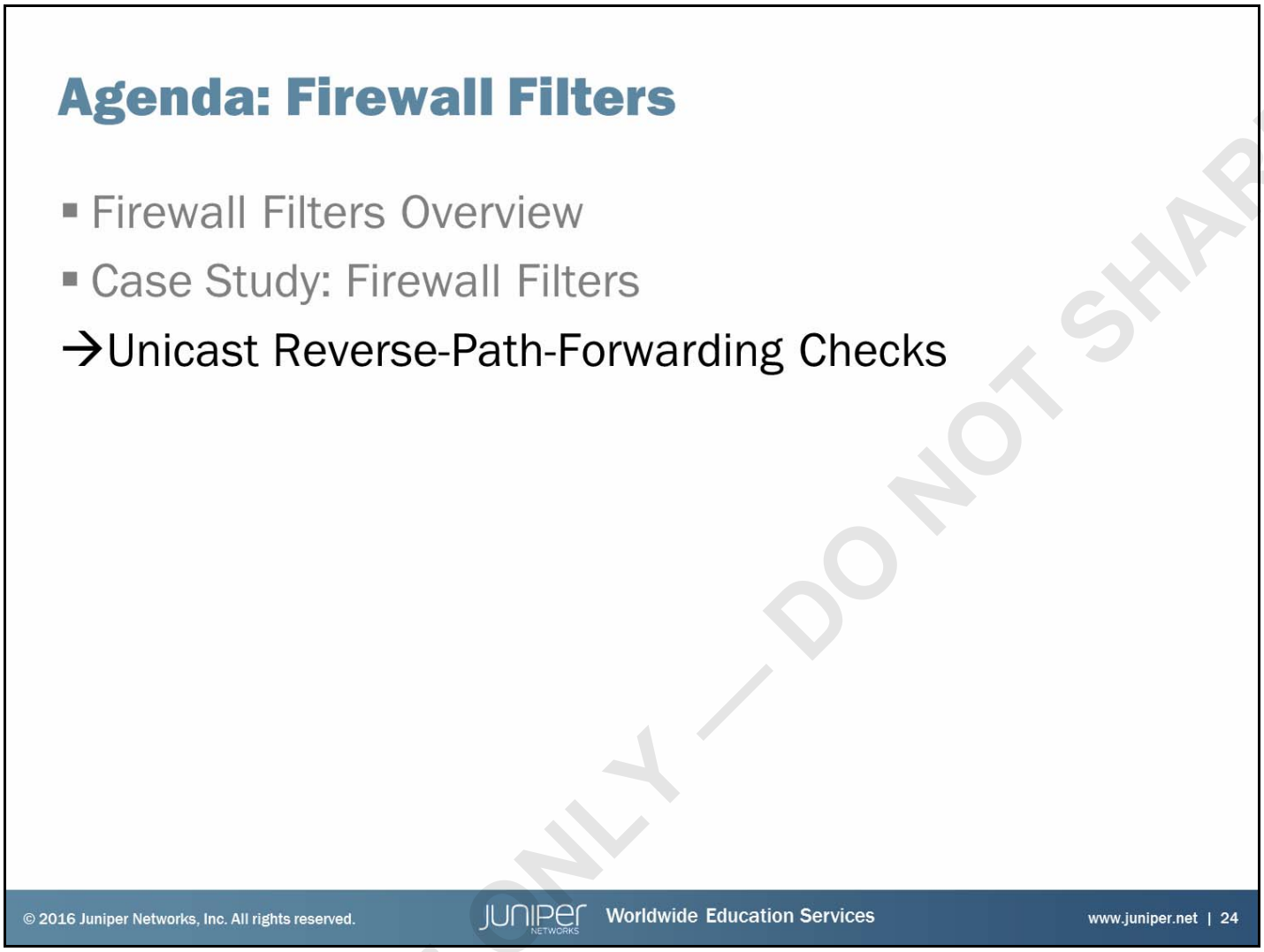
Case Study: Monitoring the Results

The slide illustrates some common **show firewall** commands used to monitor filters. In the configuration for this case study we used the `count` and `log` action modifiers.

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

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

As shown on the slide, you can display the logged packets using the **show firewall log** CLI command. A filter name or a blank space appears if the RE handles the packet. Otherwise, a dash (`-`) or `pfe` appears instead of the filter name to indicate that the packet was handled by the PFE. The contents in the firewall log clear when the system reboots.

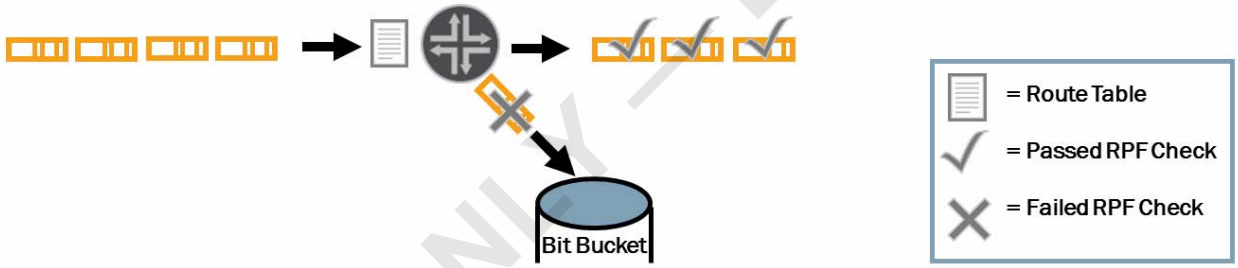


Unicast Reverse-Path-Forwarding Checks

The slide highlights the topic we discuss next.

Unicast RPF Check

- Automates antispoofing filters based on the routing table
- Two modes:
  - Strict (default)—accept packet if:
    - The packet's source address matches an active route
    - The next hop of the active route uses the interface on which the packet arrived
  - Loose—accept packet if:
    - The packet's source address matches a prefix in the routing table
    - If the default route is present, packets always match *loose* mode



Automated Antispoofing Filters

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

For example, if a device running the Junos OS receives a packet with a source address of 10.10.10.10 on interface ge-0/0/1.0 and you configured the device 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 ge-0/0/1.0, the packet passes the unicast RPF check and is accepted. You can combine both unicast RPF and firewall filters on the same interface.

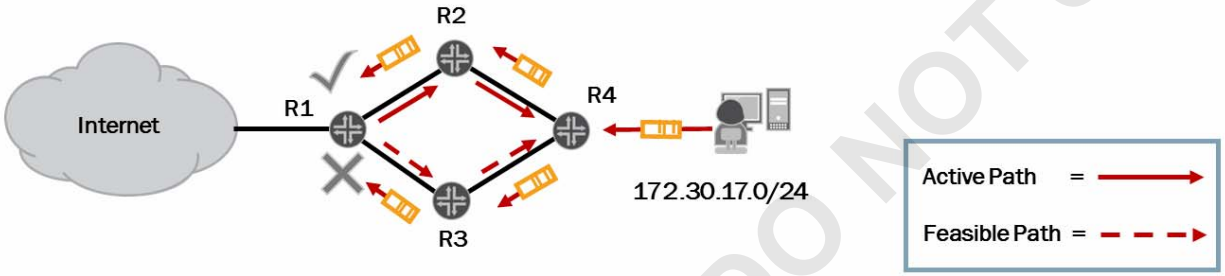
The Junos OS accomplishes unicast RPF checks by downloading additional information to the PFE. Therefore, activating this feature increases PFE memory usage.

Strict Versus Loose

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

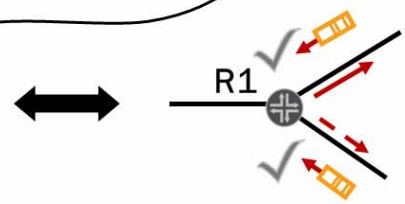
Unicast RPF Caveats (1 of 2)

- Active versus feasible paths (strict mode):
  - By default, the software checks only active paths to a prefix, which can cause drops when multiple paths exist:



  - Enable the option to consider all feasible paths:

```
routing-options {
    forwarding-table {
        unicast-reverse-path feasible-paths;
    }
}
```

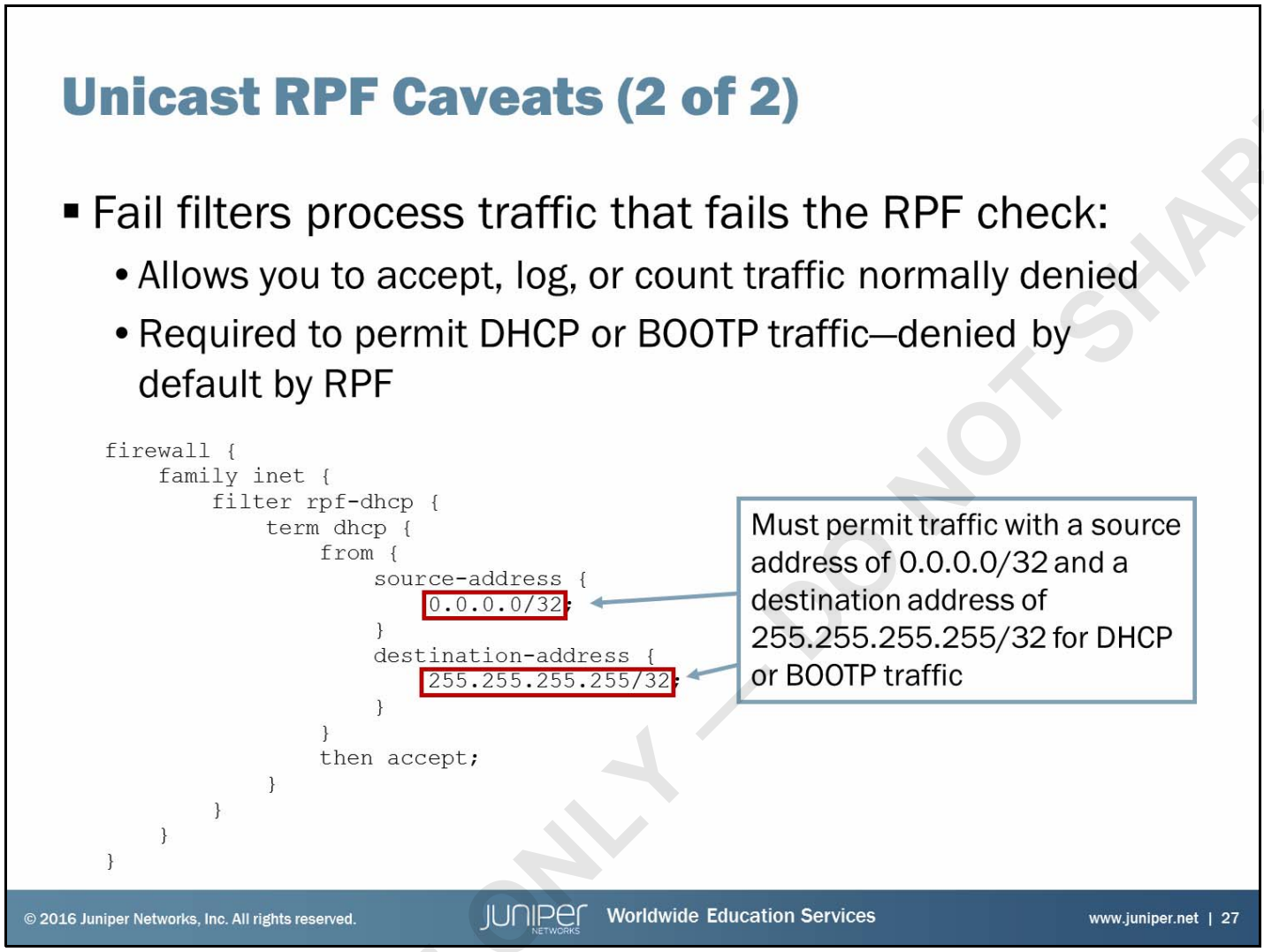


Active Versus Feasible Paths

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

You do not need to implement RPF checking on all devices within your network. Typically, you configure only the edge device to perform RPF checking because all inbound and outbound spoofing passes through that device. In the sample network shown on the slide, R1 should be configured to perform RPF checks on all three interfaces.

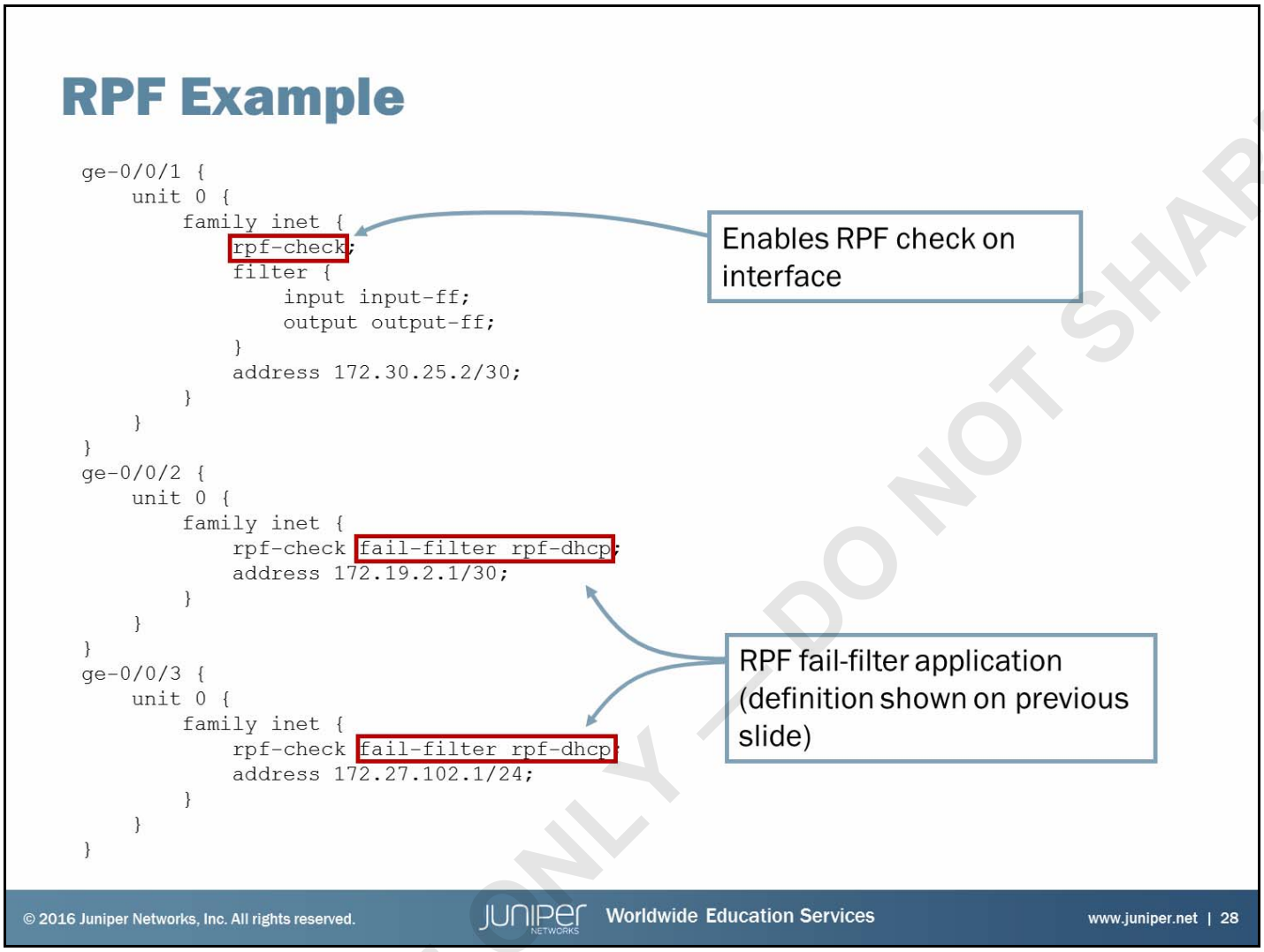
Unicast RPF configuration options vary between Junos devices. Check your product specific documentation for detailed support information.



Fail Filters

When a device running the Junos OS decides that a packet has failed the RPF check, it discards it by default. However, if you specify an optional fail filter, the device 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, the software does not log them. To log these packets, you must log them in an RPF fail filter.)

On most devices running the Junos OS, DHCP and Bootstrap Protocol (BOOTP) requests 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 a destination address of 255.255.255.255. The slide shows a sample fail filter to include DHCP or BOOTP requests.



RPF Example

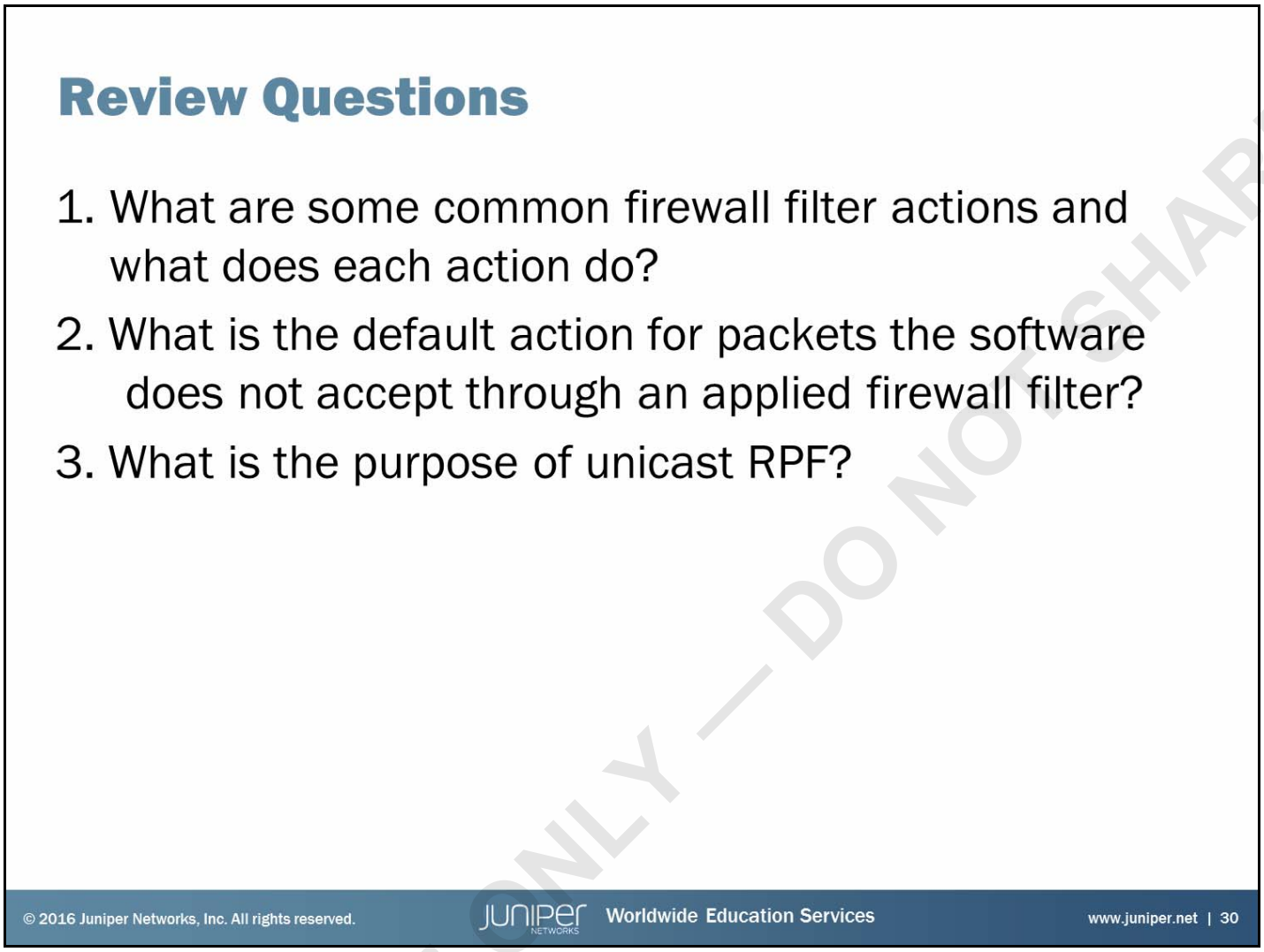
In the example on the slide, we enabled RPF in strict mode on all interfaces, and a Junos device considers only the active paths to any prefix. The fail filter named `rpf-dhcp` applies to the ge-0/0/2 and ge-0/0/3 interfaces. As you might remember, the configuration defines the `rpf-dhcp` fail-filter on the previous slide and permits DHCP and BOOTP requests. Now that you enabled RPF on all interfaces, you do not need to include antispoofing terms within the firewall filters.

Summary

- In this content, we:
  - Described the framework for firewall filters
  - Explained how the software evaluates firewall filters
  - Identified situations where you might use firewall filters
  - Wrote and applied a firewall filter
  - Described the operation and configuration for unicast RPF

We Discussed:

- The framework of firewall filters;
- Firewall filter evaluation;
- Typical usage scenarios for firewall filters;
- Configuration and application of firewall filters; and
- Unicast RPF.

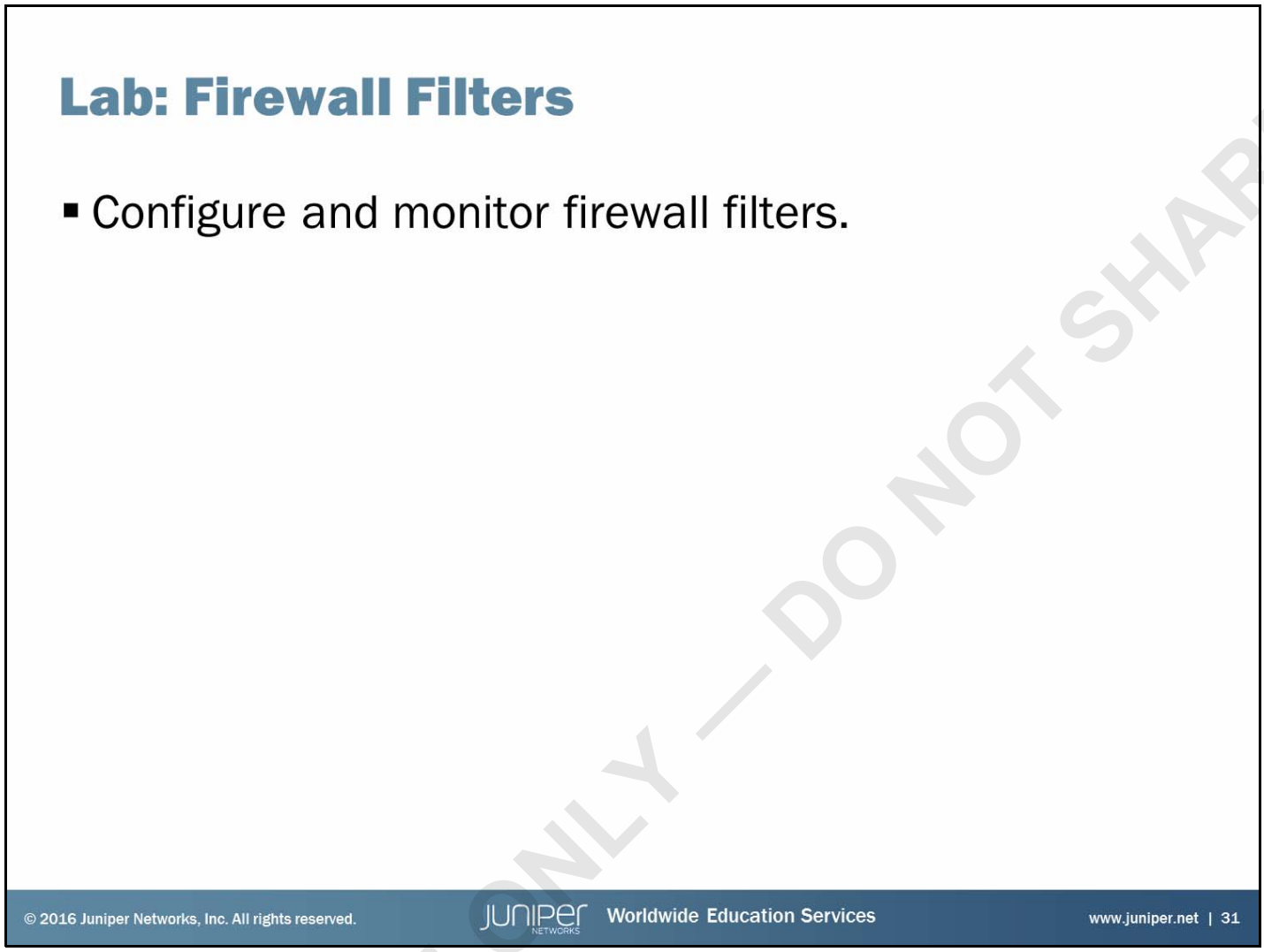


Review Questions

1.

2.

3.



Lab: Firewall Filters

The slide provides the objective for this lab.

Answers to Review Questions

1.

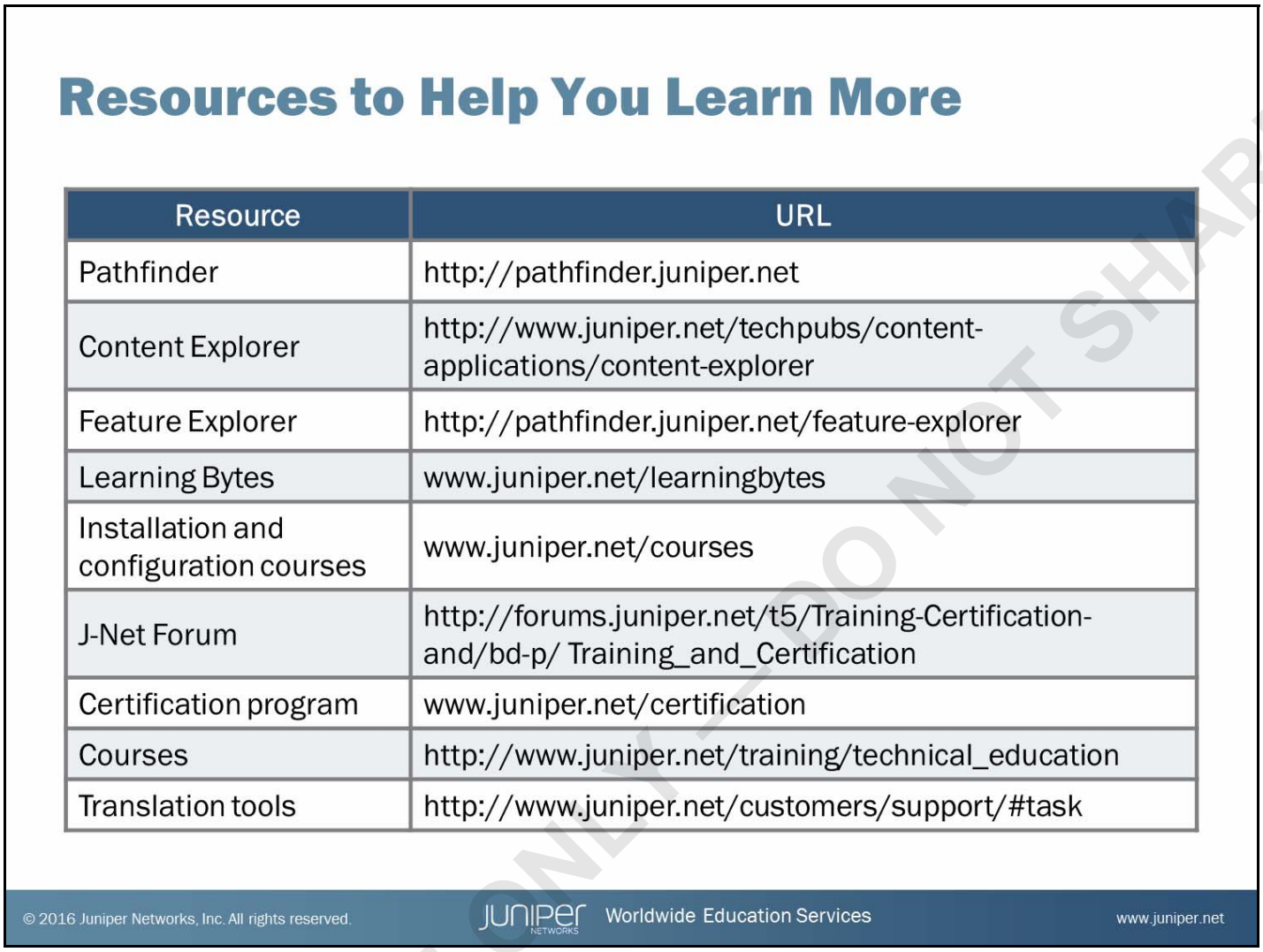
Some common firewall filter actions are `accept`, `discard`, `reject`, and `next term`. The accept action accepts the packet and continues the input or output processing of the packet. The discard action silently rejects the packet, in contrast to the reject action that drops the packet and sends an ICMP message to the source address. The next term action causes the Junos OS to evaluate the next term and is usually used when using a policer and still want the traffic to be evaluated by the rest of the filter.

2.

The default action for packets not explicitly permitted through a firewall filter is `discard`.

3.

Unicast RPF automates antispoofing on a device running the Junos OS.



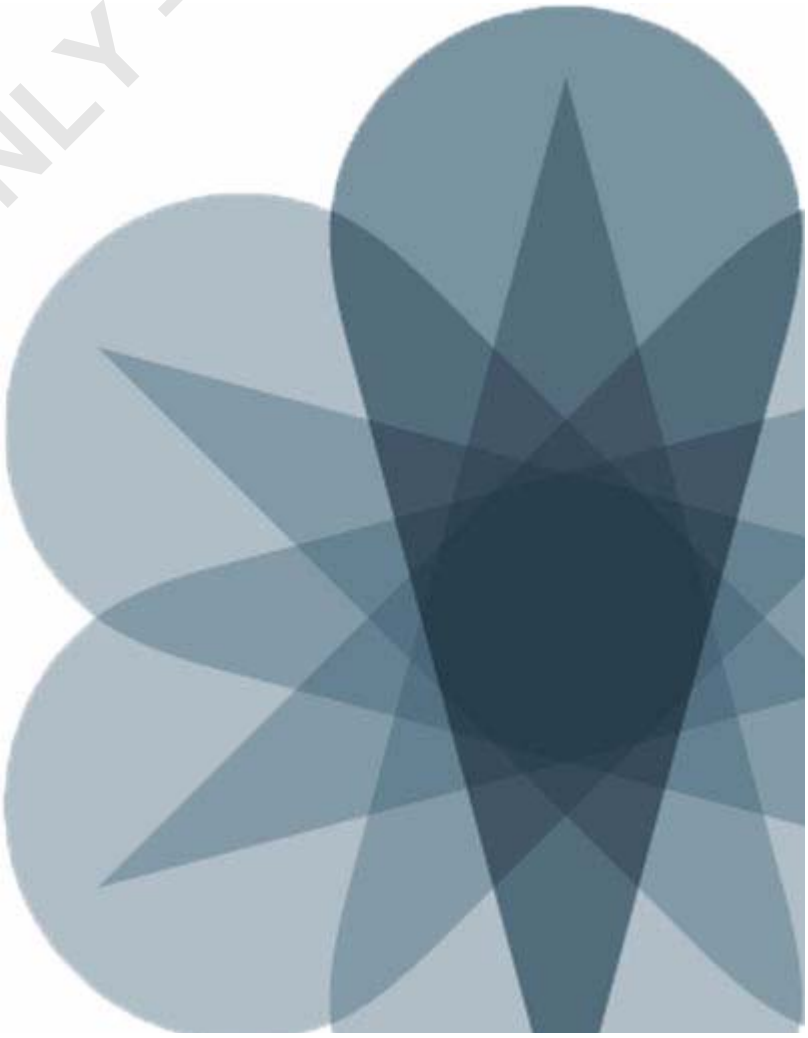
Resources to Help You Learn More

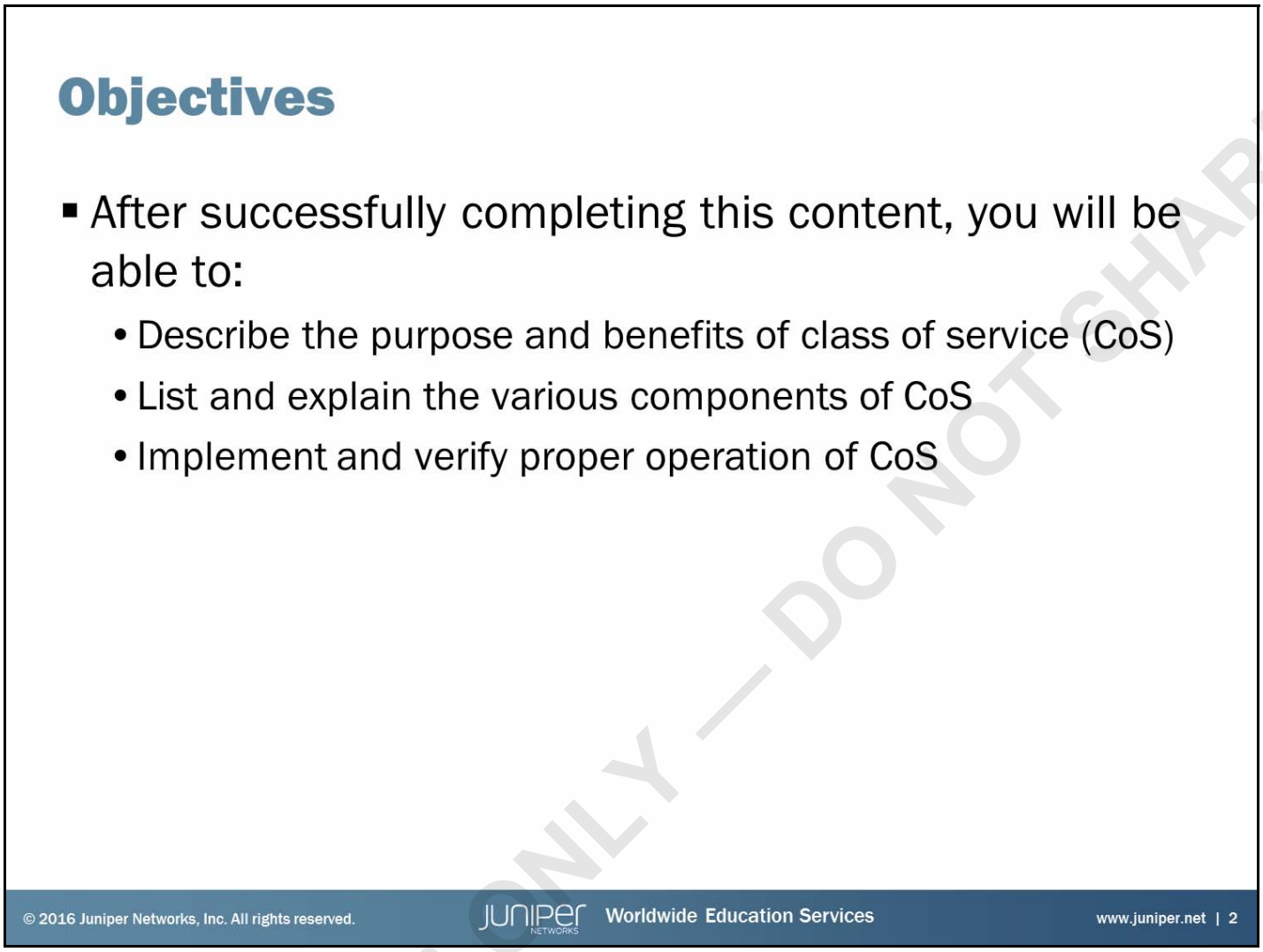
The slide lists online resources available to learn more about Juniper Networks and technology. These resources include the following sites:

- *Pathfinder*: An information experience hub that provides centralized product details.
- *Content Explorer*: Junos OS and ScreenOS software feature information to find the right software release and hardware platform for your network.
- *Feature Explorer*: Technical documentation for Junos OS-based products by product, task, and software release, and downloadable documentation PDFs.
- *Learning Bytes*: Concise tips and instructions on specific features and functions of Juniper technologies.
- *Installation and configuration courses*: Over 60 free Web-based training courses on product installation and configuration (just choose eLearning under Delivery Modality).
- *J-Net Forum*: Training, certification, and career topics to discuss with your peers.
- *Juniper Networks Certification Program*: Complete details on the certification program, including tracks, exam details, promotions, and how to get started.
- *Technical courses*: A complete list of instructor-led, hands-on courses and self-paced, eLearning courses.
- *Translation tools*: Several online translation tools to help simplify migration tasks.

Junos Routing Essentials

Appendix A: Class of Service





We Will Discuss:

- The purpose and benefits of class of service (CoS);
- Components used with CoS; and
- Implementation and verification of CoS components.

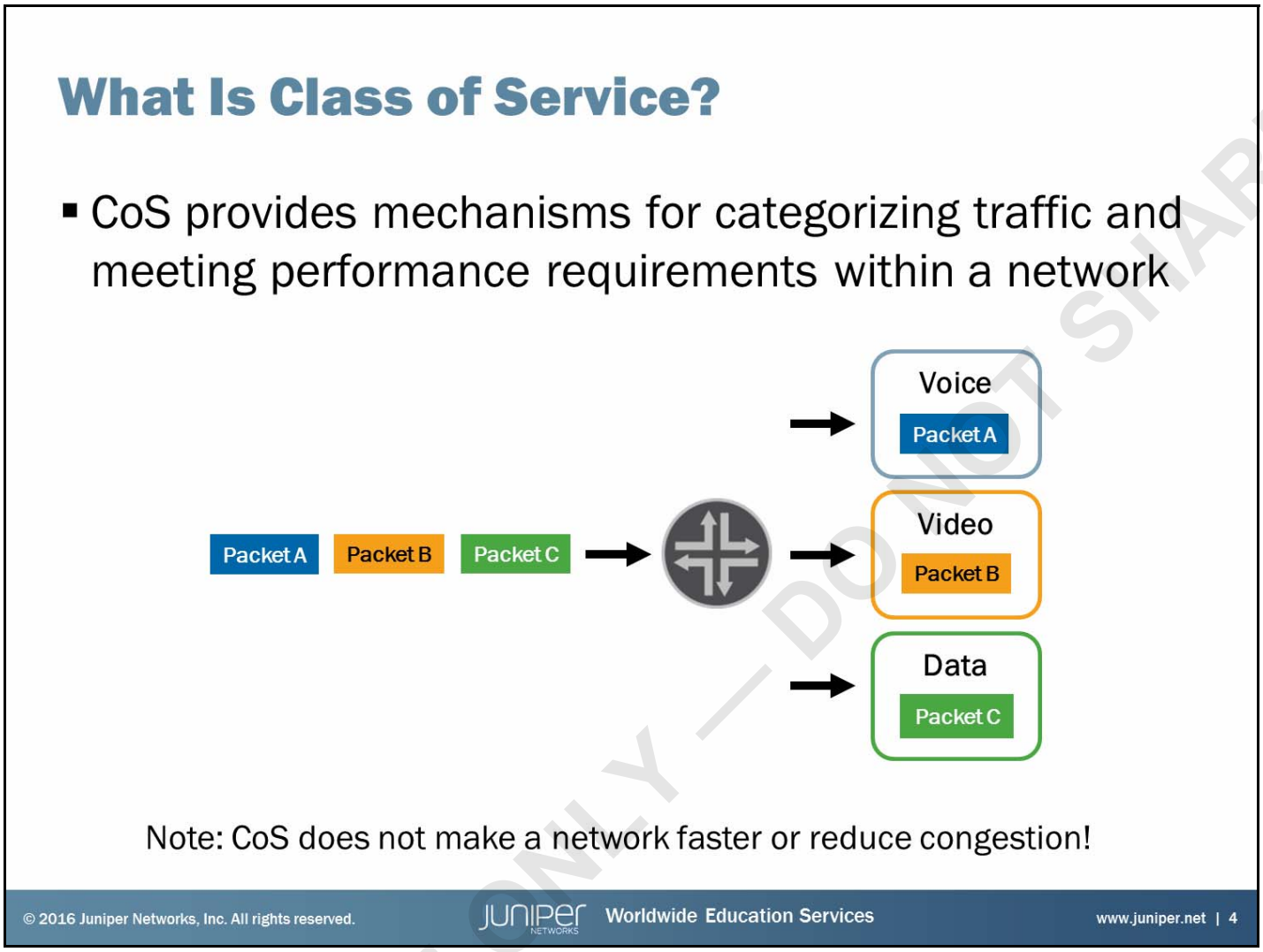
Agenda: Class of Service

→CoS Overview

- Traffic Classification
- Traffic Queuing
- Traffic Scheduling
- Case Study: CoS

CoS Overview

The slide lists the topics we will discuss. We discuss the highlighted topic first.



Uses of Class of Service

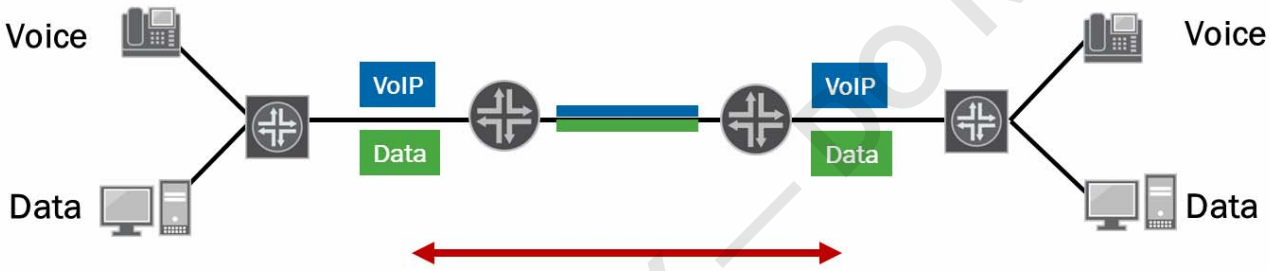
By default, devices running the Junos operating system treat all transit traffic equally. The software handles all traffic entering the device on a first-come, first-served basis. The device mixes together all traffic transiting the system and places it in the same input and output queues, which means the traffic is subject to the same potential for delays and drops. We refer to this method as *best-effort* traffic processing.

The CoS features available to devices running the Junos OS allow differentiated services to network traffic where best-effort traffic processing is insufficient. Several components to the CoS tool kit exist. First, tools exist that allow the system to place traffic into different categories (named *forwarding classes*) where the system provides the same services. Second, certain components allow the system to treat traffic for each forwarding class in a unique manner. Finally, additional tools allow the system to mark packets with their category so that other devices in the network know how to categorize them.

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. CoS cannot eliminate congestion within a network. CoS can, however, help you control how this congestion affects different types of traffic.

Meeting Performance Requirements

- CoS meets a network's performance requirements by:
  - Prioritizing latency-sensitive traffic such as VoIP
  - Controlling congestion to ensure service level agreement maintenance
  - Allocating bandwidth for different classes of traffic



Devices should treat traffic consistently throughout the entire network

Meeting the Requirements

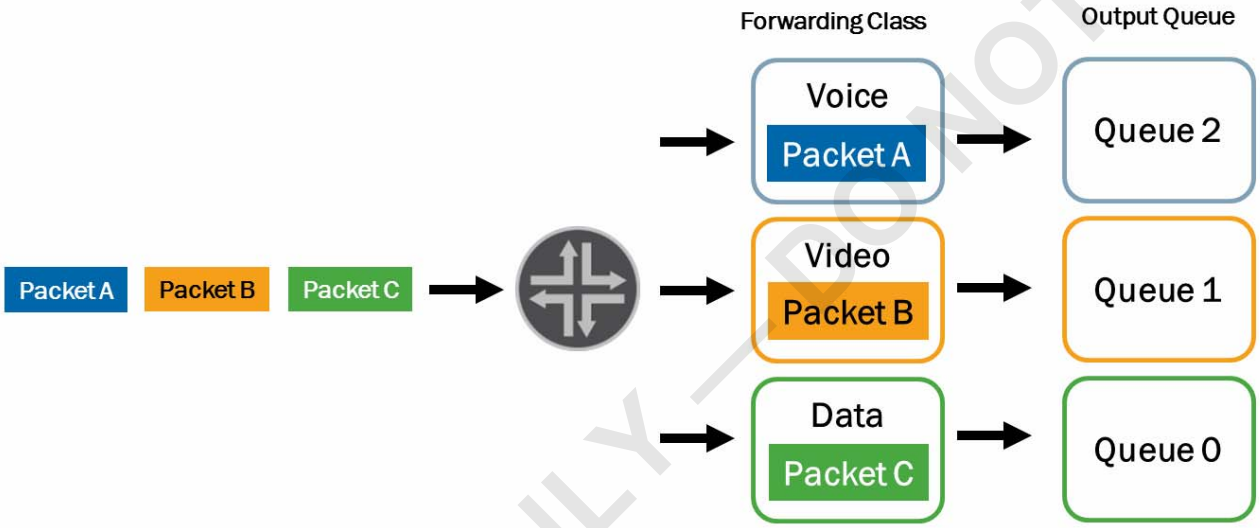
You can use CoS to control the order in which traffic is forwarded through devices running the Junos OS. Even in networks that are not fully utilizing the capacity on all their network links, instantaneous contention for transmission to the wire can occur. If two packets arrive at an output interface at the same time, the system transmits one packet and adds the other to a queue. 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 transmits before other traffic. Prioritization also controls the way that extra bandwidth is allocated after all bandwidth guarantees have been met.

Most devices running the Junos OS use the random early detection (RED) algorithm to avoid congestion. Unlike the tail dropping method, the RED algorithm selectively drops random packets before congestion becomes critical. Using the RED algorithm causes the affected TCP sessions to go into slow-start mode, reducing the total amount of traffic clients attempt to transmit through the congested link. Because of the algorithm used, higher-bandwidth data streams are the most likely to be affected, whereas lower-bandwidth streams (including most interactive sessions) are the least likely to be affected. Because the RED algorithm monitors the queue and drops packets based on statistical probabilities rather than on when the queue is full, TCP global synchronization is avoidable.

You can also configure devices running the Junos OS to guarantee certain levels of bandwidth to traffic classes. This configuration ensures that the device meets minimum bandwidth guarantees 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.

Forwarding Classes

- Forwarding classes:
  - Identify traffic that should receive common treatment
  - Used to assign traffic to output queues



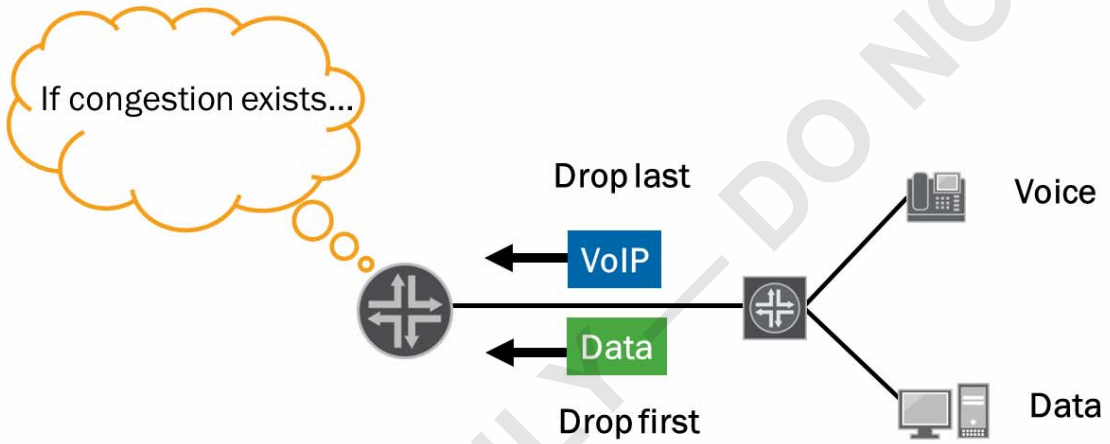
Forwarding Classes

A forwarding class is an abstract concept devices running the Junos OS use to identify traffic that should receive common treatment. The device associates traffic with a forwarding class during the classification process. During output, the system assigns traffic to a particular output queue based on forwarding class and rewrites behavior aggregate markers based on forwarding class.

Thinking of forwarding classes as output queues is tempting (and sometimes helpful) because a one-to-one mapping of forwarding classes and output queues usually exists. However, that definition is not necessarily true, because some platforms support more forwarding classes than queues. In these cases, combining the concepts of forwarding classes and output queues can be confusing.

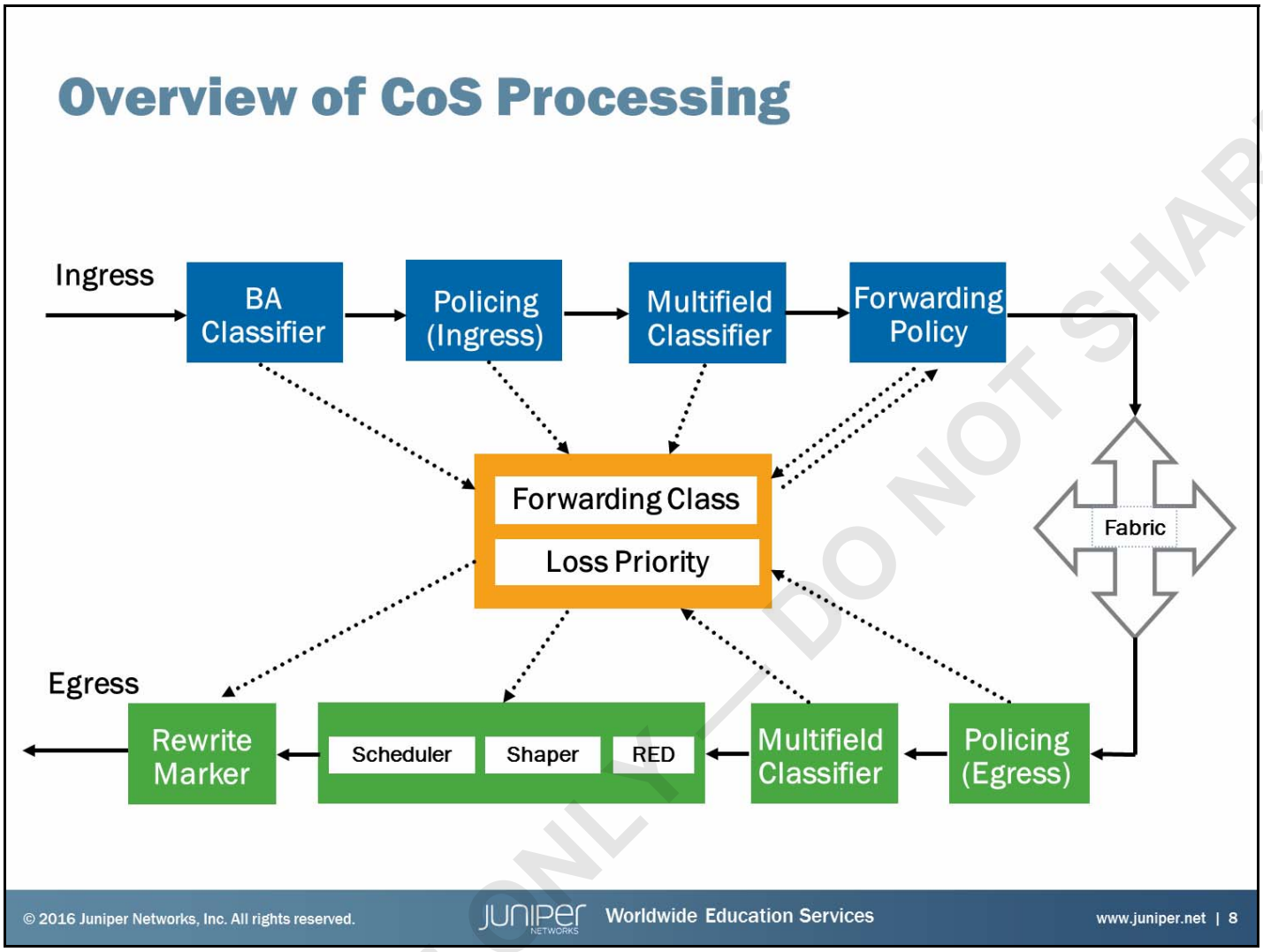
Loss Priority

- Loss priority:
  - Identifies the priority a system should give to dropping a packet
  - Used to select the drop profile used in the RED process



Loss Priority

You can associate a loss priority with a packet to tell the system which 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 OS. 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 and loss priority box. If the process uses the forwarding class and loss priority as input values, the arrow points away from the forwarding class and loss priority box.The CoS processing chart illustrated on the slide relates to some packet-based Junos routing devices. The actual processing flow varies between platforms. Check your product-specific documentation for details.

You can configure devices running the Junos OS 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. Edge devices usually set these fields, which indicate the category of traffic. In this way, a device in the core does not need to recategorize traffic it receives from an edge device. Because these fields indicate the category of traffic to which the packet belongs, we refer to this kind of classification as a *behavior aggregate* (BA) classifier.

You can also configure Junos devices to set the forwarding class and loss priority with multifield classifiers on both ingress and egress. You implement multifield classifiers using firewall filters. Multifield classifiers allow you to 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.

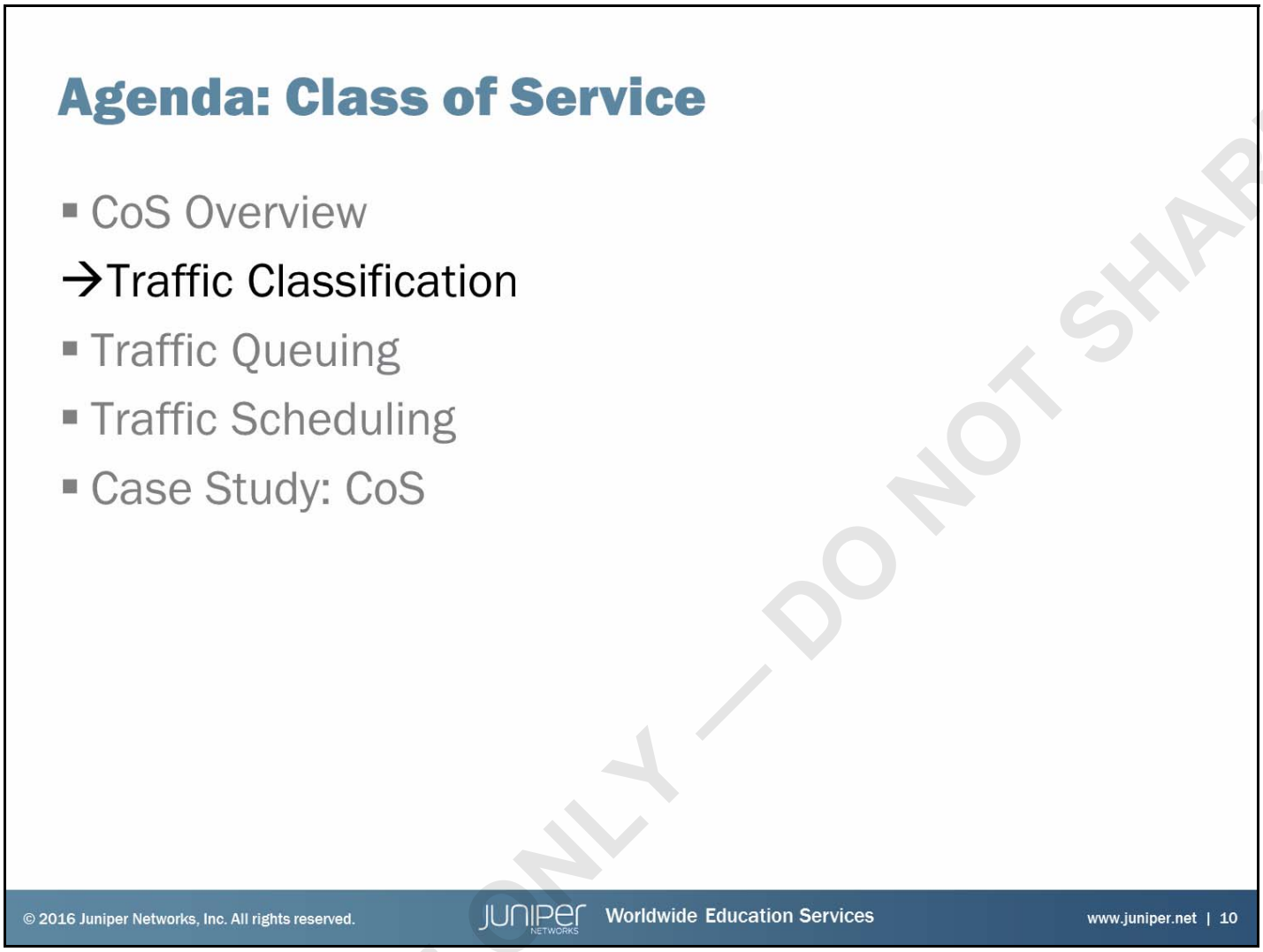
Continued on the next page.

CoS Processing (contd.)

You can use forwarding policy options to implement CoS-based forwarding. When multiple equal-cost paths to a destination exist, you can use CoS-based forwarding to specify which of the equal-cost paths to use 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.

Devices running the Junos OS use the forwarding class and loss priority on egress to assign traffic to queues, implement the RED algorithm, and rewrite BA headers.

Note that CoS support varies between platforms running the Junos OS. For detailed support information, consult the documentation for your specific platform.

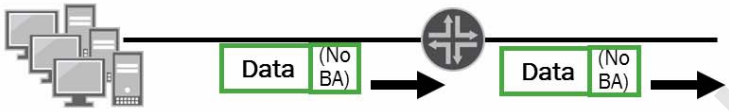


Traffic Classification

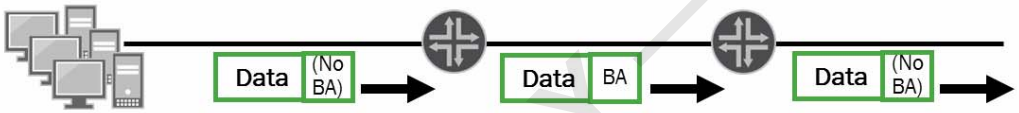
The slide highlights the topic we discuss next.

CoS Deployment Models

- Two primary CoS deployment models exist:
  - In-the-box model uses a multifield classifier only



  - Across-the-network model uses a multifield classifier at edge and BA rewrite and classifier in the core



Devices should treat traffic consistently throughout the entire network.

CoS Deployment Models

Two models for CoS deployment exist. In some networks, the only use for CoS is to provide differentiated forwarding services through a single device. In other networks, the CoS deployment provides differentiated services across multiple devices within a network.

With a deployment limited to a single device, the system typically classifies packets as they pass through using a multifield classifier, then uses the classification to provide the configured service level, and forwards the packets without any BA markings (DSCP, IP precedence, and so forth). However, when the deployment crosses multiple devices, the classification usually occurs differently.

When you configure multiple devices in a network to use CoS to provide differentiated services, it is usually beneficial to initially perform the classification on the edge device and then transmit that classification within the network using a BA. In this model, the edge device classifies traffic using a multifield classifier as traffic enters the network. When the edge device transmits the packet into the network, it marks the packets with a BA. The remainder of the devices in the network read the BA and automatically set the correct forwarding class and loss priority without a multifield classifier.

The use of 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 system. Without BAs, each device must have a multifield classifier capable of classifying all traffic entering the network, and you would need to replicate any classification change on all other devices. 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, you do not add additional overhead by setting a Layer 3 BA.

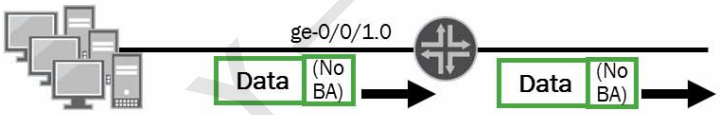
Multifield Classifiers

Definition

```
[edit firewall family inet]
user@router# show 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;
}
```

Application

```
[edit interfaces]
user@router# show ge-0/0/1
unit 0 {
    family inet {
        filter {
            input apply-cos-markings;
        }
        address 192.168.200.1/24;
    }
}
```



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 Junos devices apply multifield classifiers after BA classifiers, multifield classifiers always override the forwarding class and loss priority assigned by the BA.

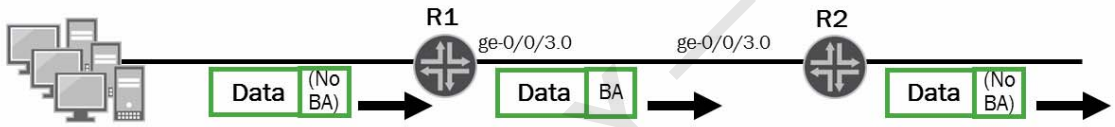
Behavior Aggregates

Behavior Aggregate Rewrite

```
[edit class-of-service]
user@R1# show
interfaces {
    ge-0/0/3 {
        unit 0 {
            rewrite-rules {
                inet-precedence default;
            }
        }
    }
}
```

Behavior Aggregate Classifier

```
[edit class-of-service]
user@R2# show
interfaces {
    ge-0/0/3 {
        unit 0 {
            classifiers {
                inet-precedence default;
            }
        }
    }
}
```



Behavior Aggregates

You configure Junos devices to apply BA markers to packets leaving an interface by applying a rewrite rule to the outbound interface. By default, the Junos OS does not modify a packet's Layer 3 BA header fields as it is sent through the device, so setting these fields is necessary only once, usually on the device at the edge of the network. However, the software does not preserve Layer 2 fields (such as the MPLS EXP and IEEE 802.1p fields), so you must configure the system 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. For example, to apply the default IP precedence markings, 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 the next page.

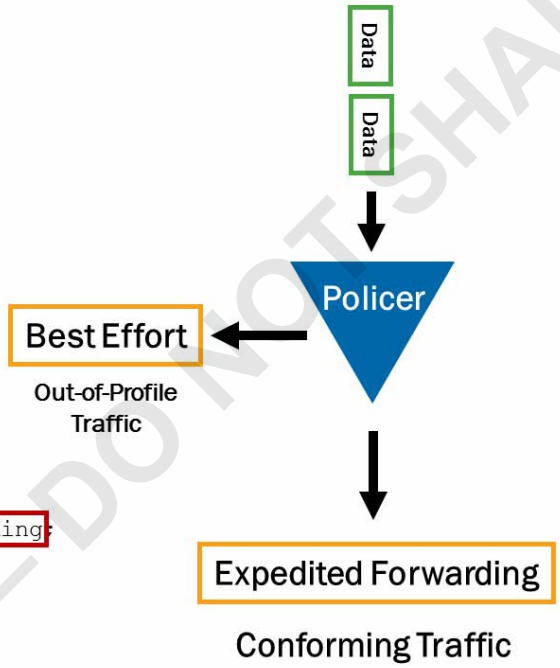
Behavior Aggregates (contd.)

Once a packet is marked with a BA, you can have downstream devices read those markers and automatically assign packets the correct forwarding class and loss priority. To initiate this process, you apply a BA classifier to the inbound interface. You apply BA classifiers to interfaces under the `[edit class-of-service interfaces]` hierarchy. For example, to apply the default IP precedence BA classifier, 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 maps 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 devices 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;
            }
        }
    }
}
```

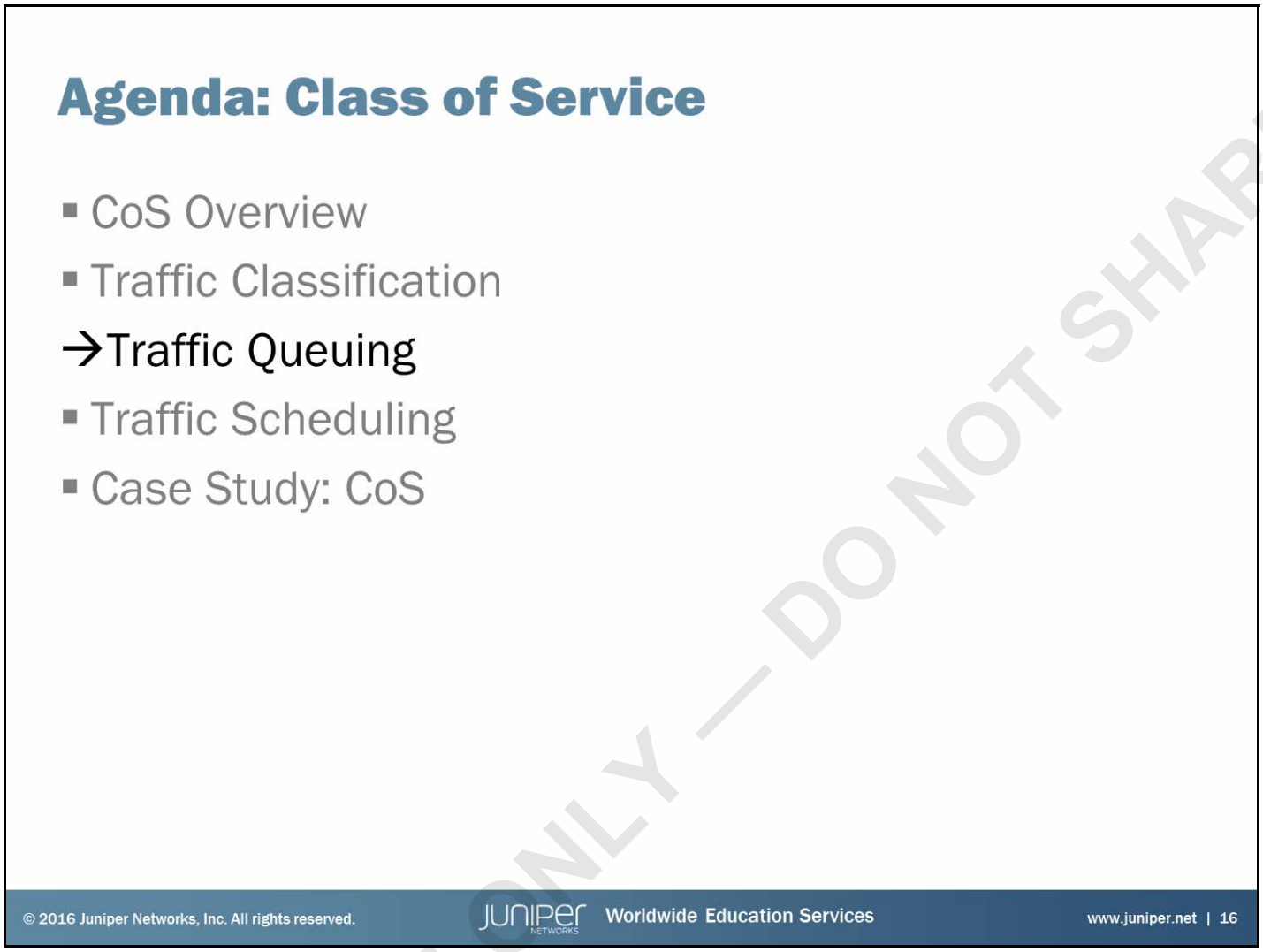


Policers

Policers allow you to limit certain kinds of traffic to a specified bandwidth and burst size. You can configure Junos devices to assign a different forwarding class or loss priority to traffic exceeding the configured limits. You can configure these policers 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 policer, shown on the slide, assigns traffic within the specified rate limit to the `expedited-forwarding` class. The policer assigns traffic that exceeds the specified rate limit to the `best-effort` forwarding class. Although not illustrated on the slide, you must apply the *`apply-cos-markings`* firewall filter to the proper interface for the traffic classification to function properly.

Note that some platforms running the Junos OS support only a subset of the available policer actions. Check the current documentation for your product for support information.

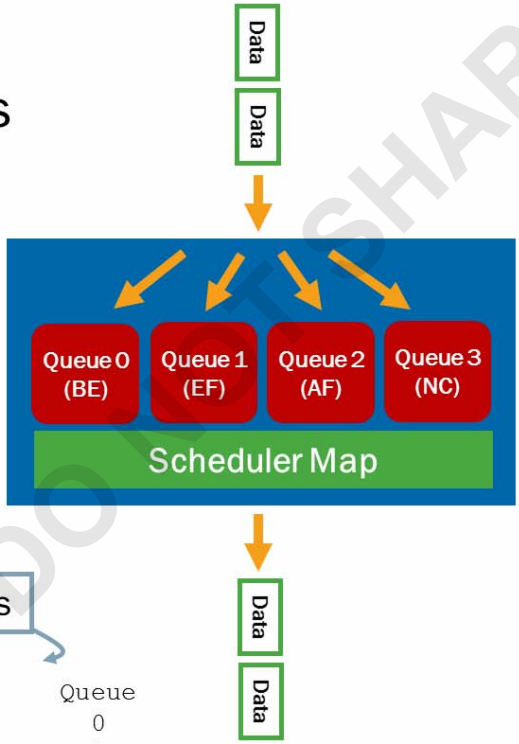


Traffic Queuing

The slide highlights the topic we discuss next.

Queuing

- Forwarding classes map to queues
  - Default queue and forwarding class mappings for most devices running Junos OS:
    - 0: best-effort
    - 1: expedited-forwarding
    - 2: assured-forwarding
    - 3: network-control



Displays current queue and forwarding class mappings

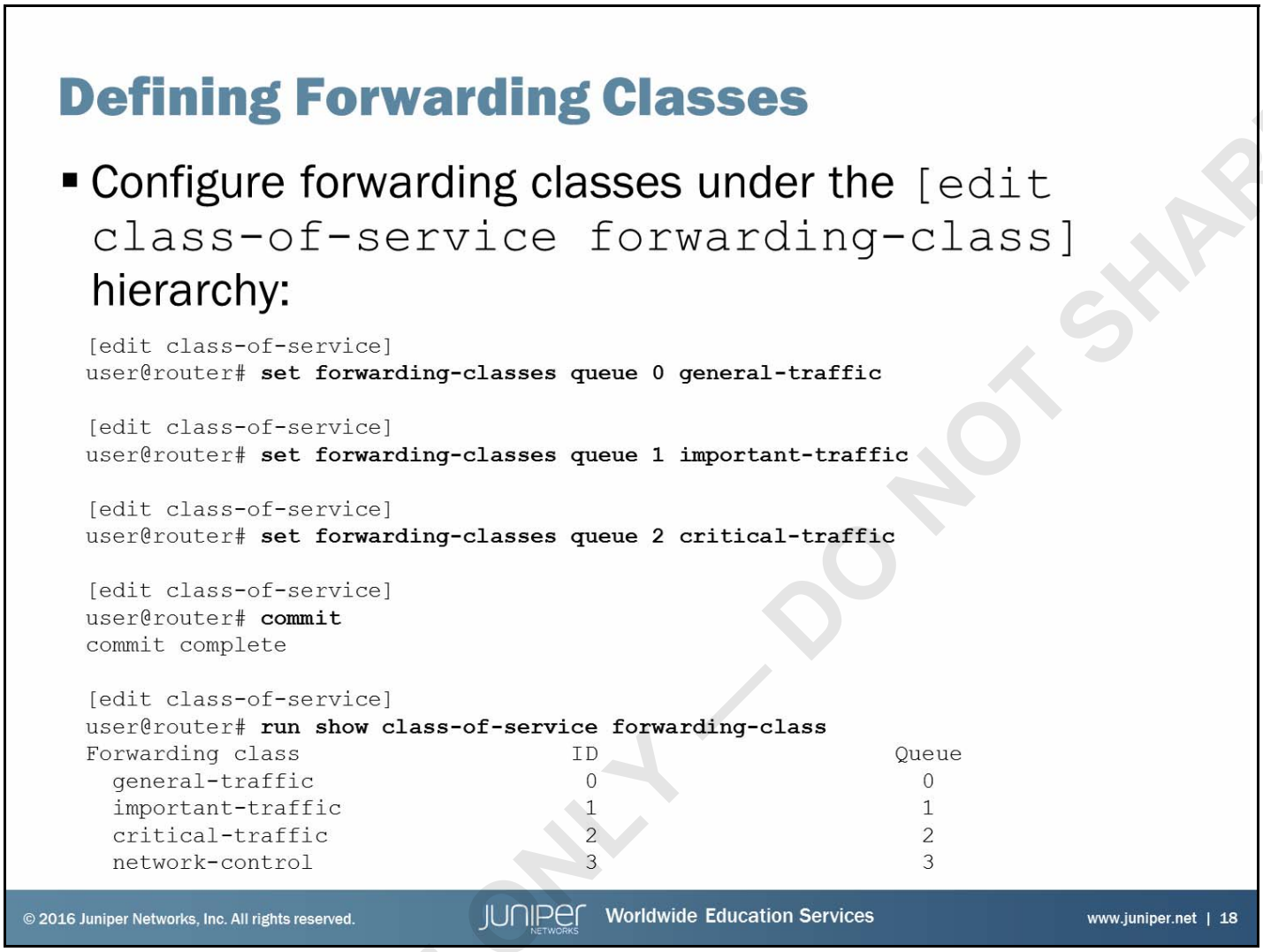
```
user@router> show class-of-service forwarding-class
Forwarding class                  ID                      Queue
  best-effort                      0                      0
  expedited-forwarding             1                      1
  assured-forwarding               2                      2
  network-control                  3                      3
```


Queuing Overview

When traffic reaches the outbound interface, the system places traffic associated with each forwarding class in its own queue. The number of queues supported on outbound interfaces is hardware dependent.

Devices running the Junos OS associate each forwarding class with a queue number. The slide shows the default forwarding classes for most platforms running the Junos OS and their associated queue numbers. By default, the Junos OS sends certain network control traffic (routing protocol messages, keepalives, and so forth) to the queue associated with the network-control forwarding class (typically Queue 3), while all other traffic is sent to the queue associated with the best-effort forwarding class (typically Queue 0). You must configure some form of classifier to send traffic to any other queue.

After the system places traffic in the correct queues, a scheduler defines how the interface should process traffic from each queue.

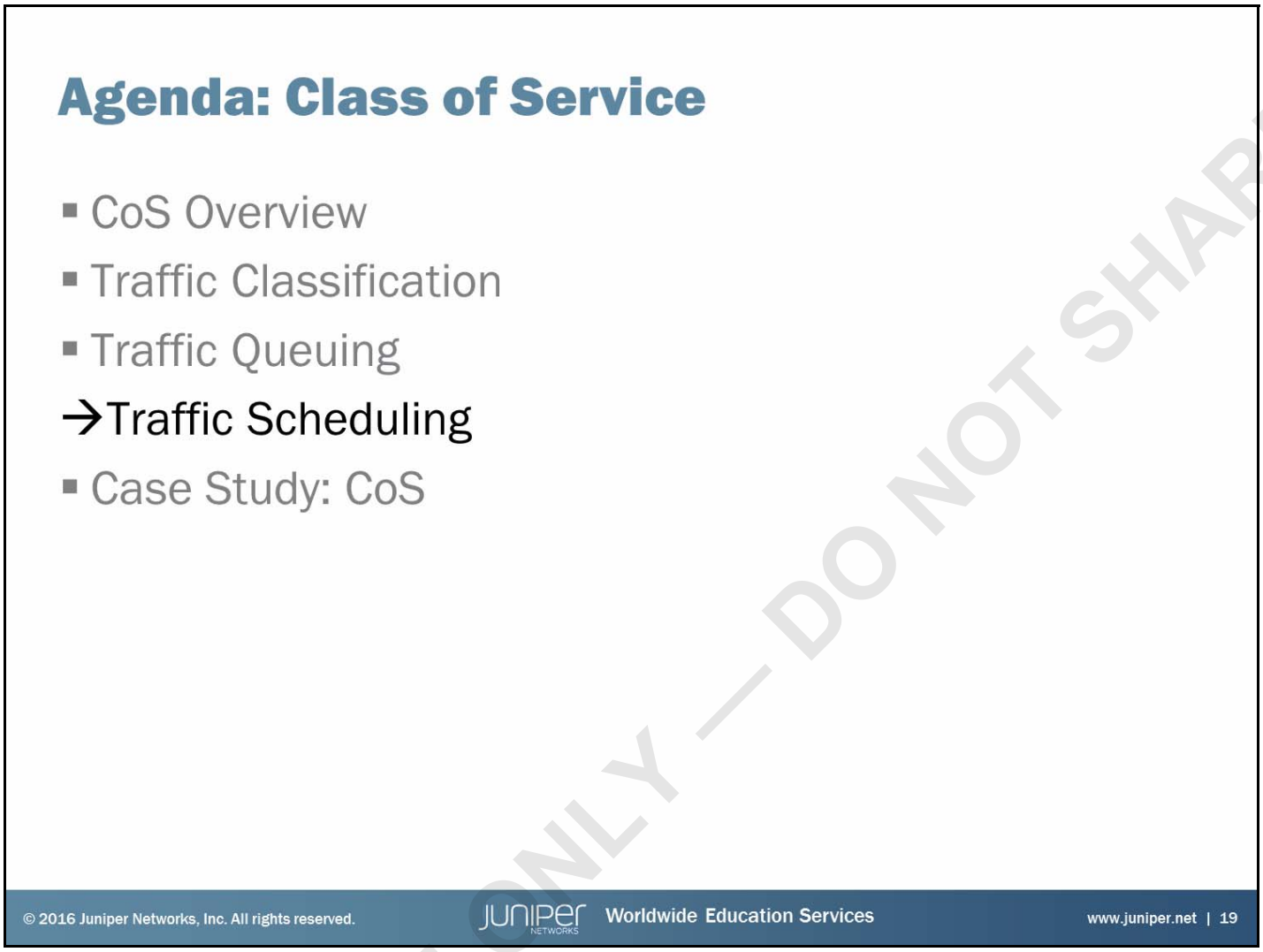


Forwarding Class Definition

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 system 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. On most platforms running the Junos OS the default forwarding class names are associated with Queues 0–3.

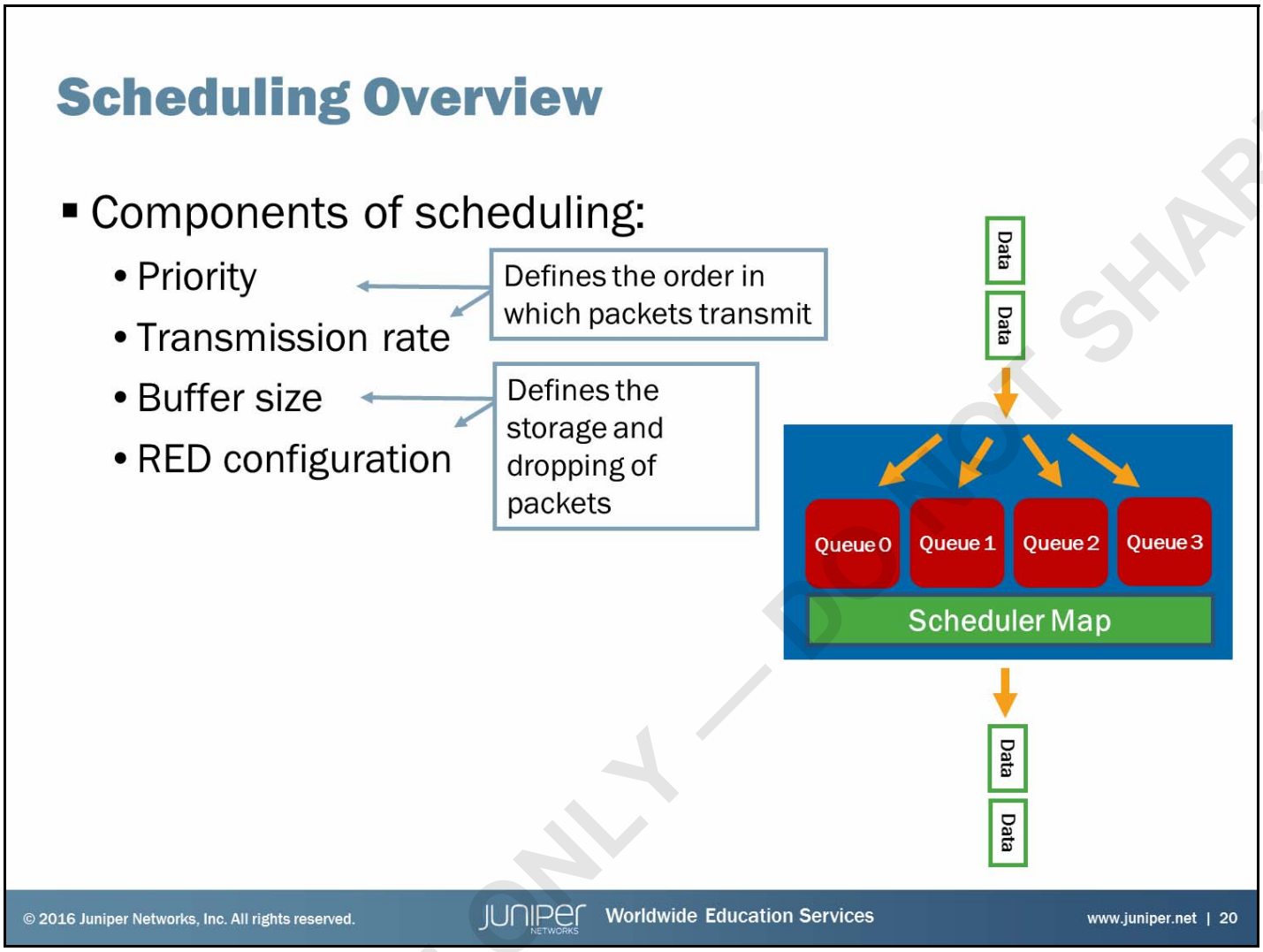
As the slide indicates, you can change the default forwarding class names. If you rename the default forwarding class names, the software still sends traffic to the corresponding queues. Additionally, the default BA classifiers and rewrite rules map traffic to and from the associated queues (again, typically Queues 0–3), regardless of the forwarding class names associated with those queues. In addition to changing the default forwarding class names, you can also assign forwarding class names to nondefault queues.

Forwarding classes are important because all other CoS rules reference forwarding classes, rather than queues. The software maintains the mapping between forwarding class names and queue numbers only in the forwarding class definitions in the `[edit class-of-service forwarding-classes]` hierarchy.



Traffic Scheduling

The slide highlights the topic we discuss next.



Scheduling Overview

You control the way the system services queues by configuring schedulers and scheduler maps. A scheduler contains parameters that describe how to service a queue. A scheduler is associated with a particular queue and forwarding class through a scheduler map.

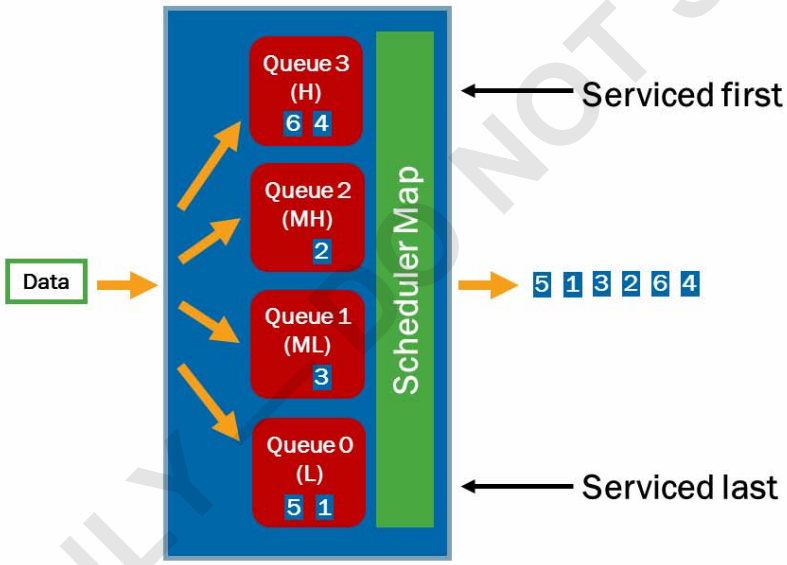
You define the order in which packets should transmit by defining a priority and a transmission rate for a forwarding class. Devices running the Junos OS typically transmit packets from forwarding classes or queues with a higher-priority before those with a lower-priority. The default priority values might vary between Junos devices.

You can configure a transmission rate for each forwarding class. The transmission rate controls how much bandwidth the traffic associated with a given forwarding class can consume. By default, the software assigns 95 percent of the bandwidth to the queue associated with the best-effort forwarding class (typically Queue 0) and 5 percent of the bandwidth to the queue associated with the network-control forwarding class (typically Queue 3). The software assigns all other queues 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.

The configured buffer size determines the size of each queue. As previously mentioned, by default, the software assigns buffer space only to the queues associated with the best-effort and network control forwarding classes. By default, all other queues have 0 percent of the available buffer space; therefore, if you use queues other than those associated with the best-effort and network-control forwarding classes, you should assign buffers to those queues. As the buffer fills, the likelihood that the RED algorithm will drop packets increases. 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 only then drop packets. Note that not all platforms running the Junos OS support RED. Check your product-specific documentation for CoS support information.

Queue Priority

- Queues receive service according to their assigned priority; common priorities include:
  - High
  - Medium-high
  - Medium-low
  - Low

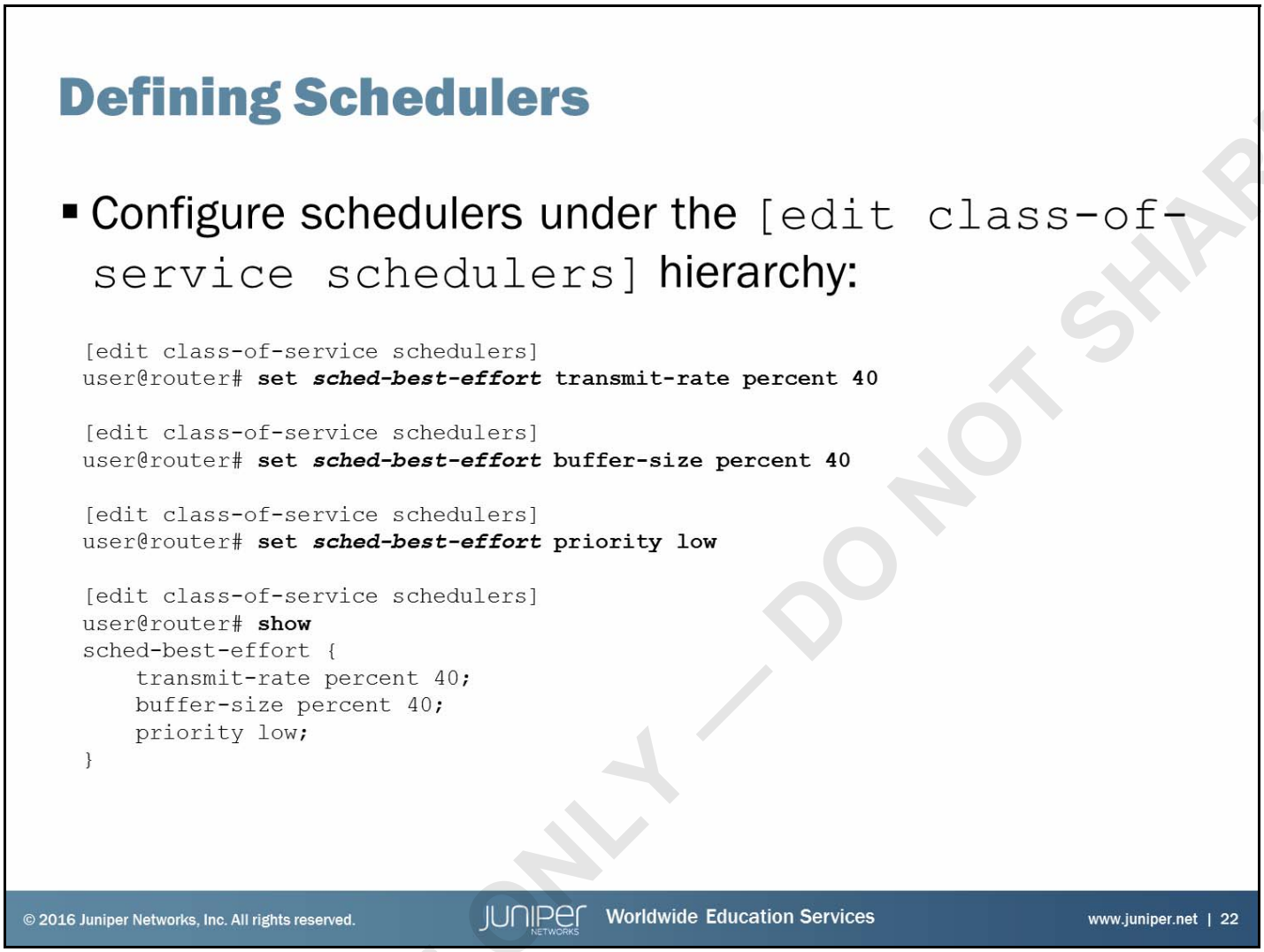


Queue Priority

The Junos OS supports multiple levels of transmission priority, which in order of increasing priority are low, medium-low, medium-high, high, and strict-high. Using this priority structure allows the Junos OS to service higher-priority queues before lower-priority queues. Priority scheduling determines the order in which an output interface transmits traffic from the queues, thus ensuring that queues containing important traffic receive better access to the outgoing interface. This procedure involves the software examining the priority of the queue. In addition, the software determines if the individual queue is within its defined bandwidth profile.

Higher-priority queues transmit packets ahead of lower-priority queues as long as the higher-priority forwarding classes retain enough bandwidth credit. When you configure a higher-priority queue with a significant fraction of the transmission bandwidth, the queue might lock out (or starve) lower-priority traffic. Strict priority queuing works differently on different platforms. For platform-specific details and behavior, refer to the documentation for your specific product.

We provide a sample configuration showing a defined bandwidth profile on the next slide.



Configuring Schedulers

You configure schedulers under the `[edit class-of-service schedulers]` hierarchy. The name of the scheduler has no special significance. The Junos OS associates schedulers with individual forwarding classes and queues using scheduler maps. In the example on the slide, the software should use the scheduler 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 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 practice should perform acceptably under many circumstances; however, cases exist 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*.

Defining Scheduler Maps

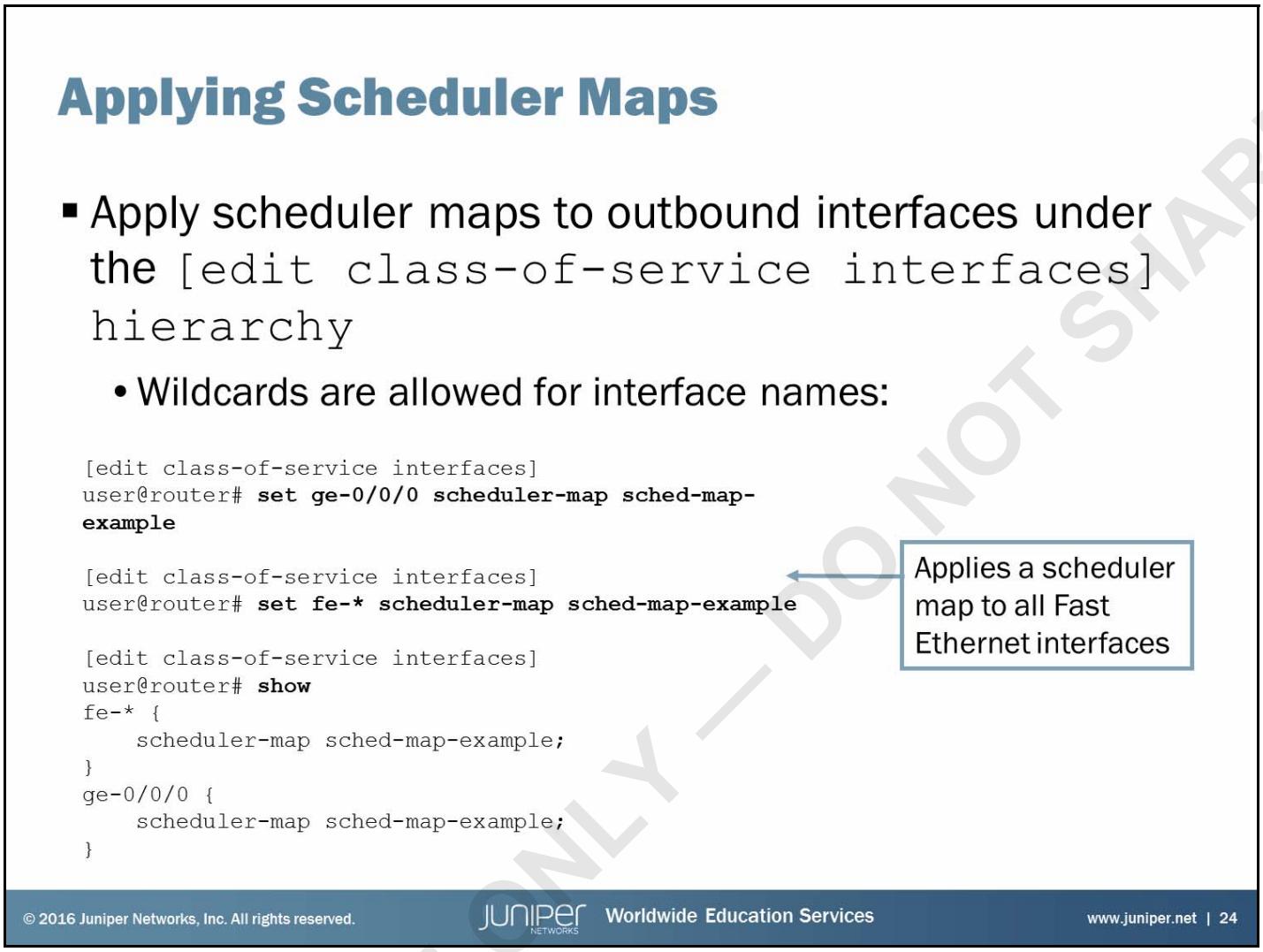
- Scheduler maps associate schedulers with forwarding classes and queues

```
[edit class-of-service scheduler-maps]
user@router# set sched-map-example forwarding-class best-effort scheduler sched-BE

[edit class-of-service scheduler-maps]
user@router# set sched-map-example forwarding-class expedited-forwarding scheduler sched-
EF

[edit class-of-service scheduler-maps]
user@router# set sched-map-example forwarding-class assured-forwarding scheduler sched-AF

[edit class-of-service scheduler-maps]
user@router# set sched-map-example forwarding-class network-control scheduler sched-NC

[edit class-of-service scheduler-maps]
user@router# show
sched-map-example {
    forwarding-class best-effort scheduler sched-BE;
    forwarding-class expedited-forwarding scheduler sched-EF;
    forwarding-class assured-forwarding scheduler sched-AF;
    forwarding-class network-control scheduler sched-NC;
}
```


Creating Scheduler Maps

Scheduler maps associate schedulers with particular forwarding classes and their respective queues. The Junos OS references queues by the name of the forwarding class associated with them. You configure scheduler maps under the `[edit class-of-service scheduler-maps]` hierarchy.

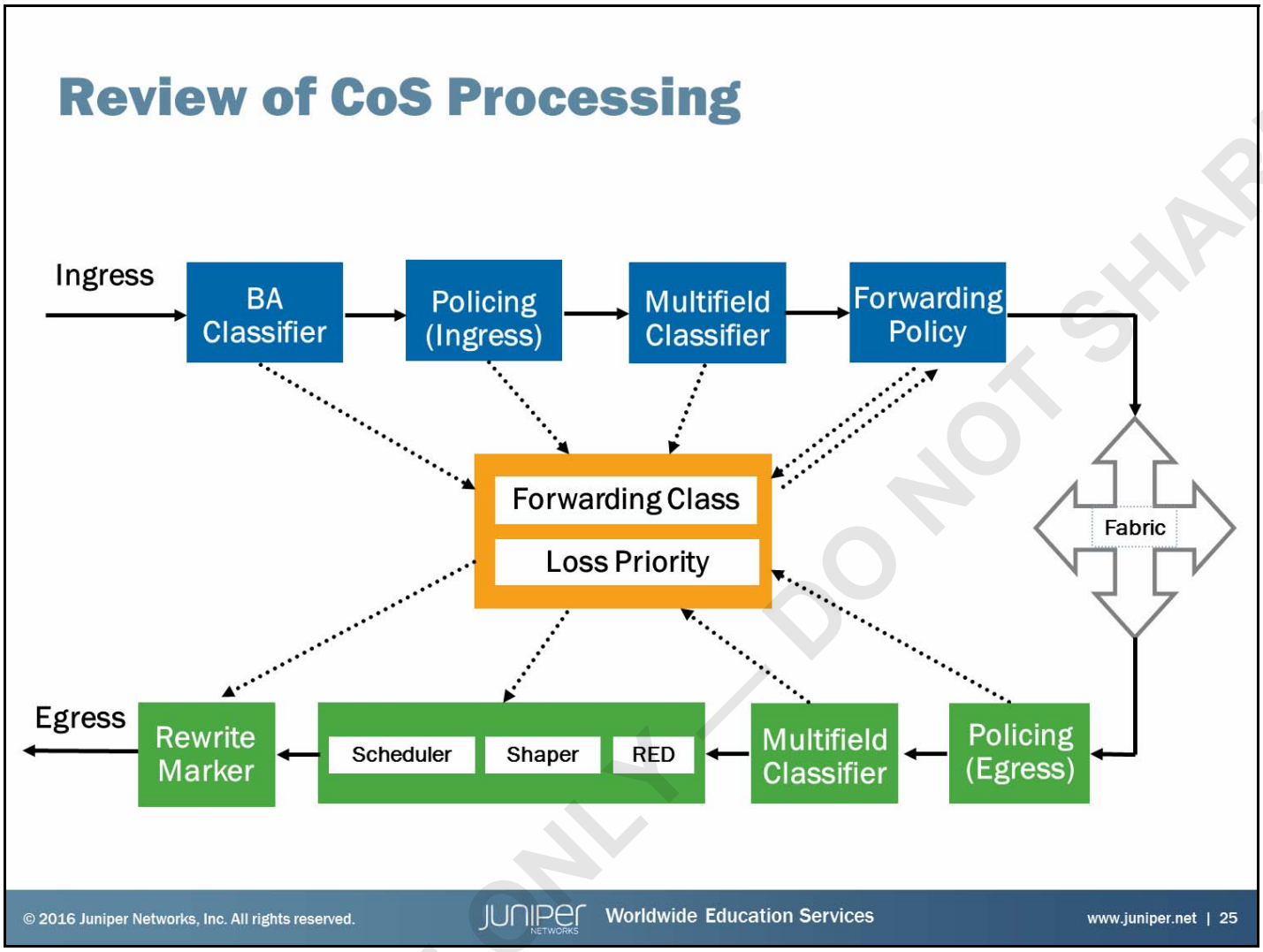
In the example on the slide, we associate four schedulers with the four default queues.



Applying a Scheduler Map to an Interface

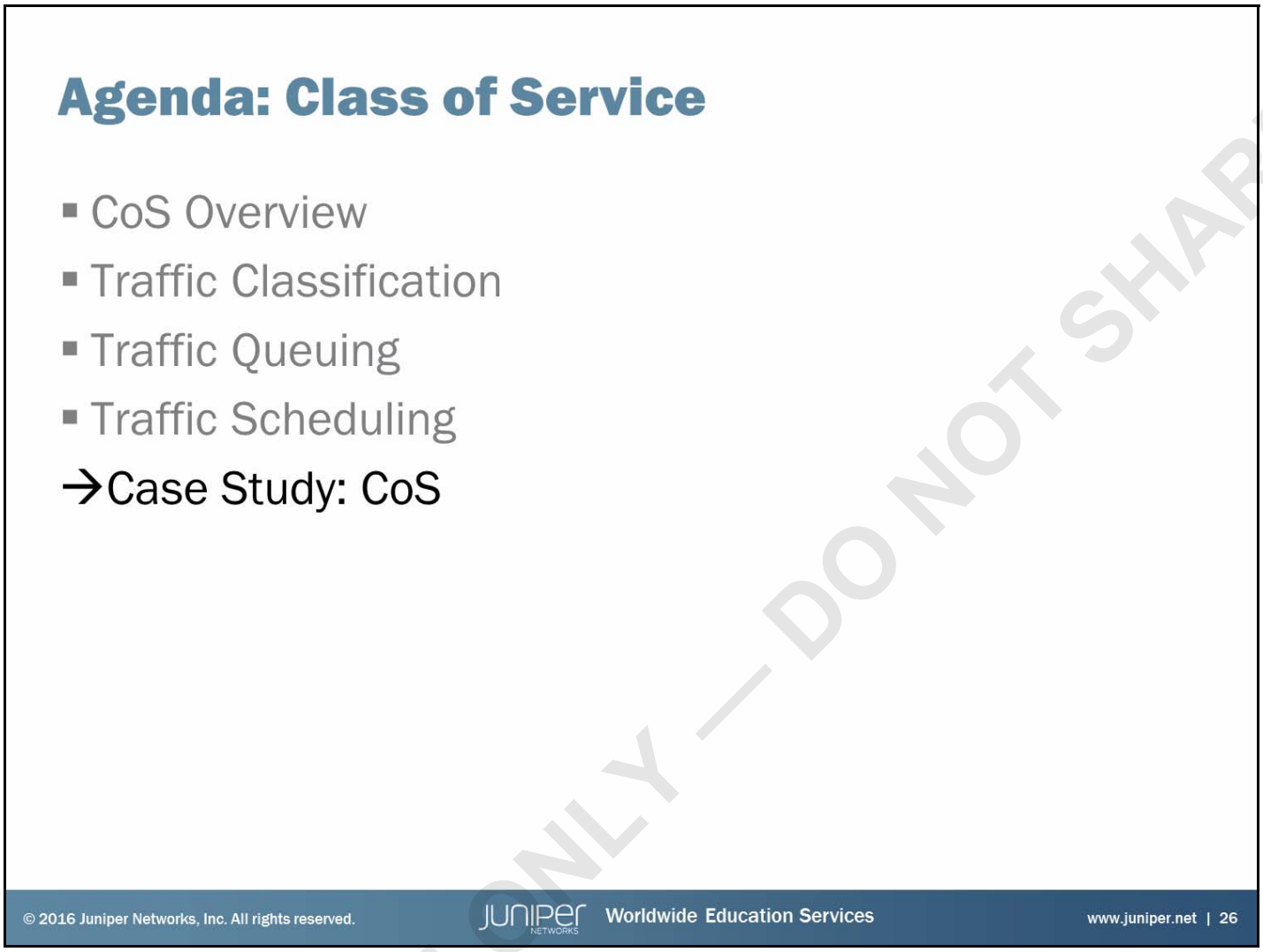
The final step in configuring a device running the Junos OS to service its queues is to associate the scheduler map with an outbound interface. When packets arrive at the output interface, the system places the packets in the appropriate queue for the forwarding class of the traffic. The system determines the order in which to transmit packets from the queues by referring to the parameters configured in the scheduler associated with a given queue. The scheduler map tells the system 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 you configure `per-unit-scheduler` for the interface under the `[edit interfaces]` hierarchy. In that case, the system schedules packets on a per-unit basis, and you can also define different scheduler maps for each unit. Support for per-unit schedulers varies between platforms running the Junos OS.



Review of CoS Processing

We introduced the diagram shown on the slide earlier in this appendix. Now that we have covered the concepts, we review the diagram to ensure that you understand how these concepts fit into the big picture.



Case Study: CoS

The slide highlights the topic we discuss next.

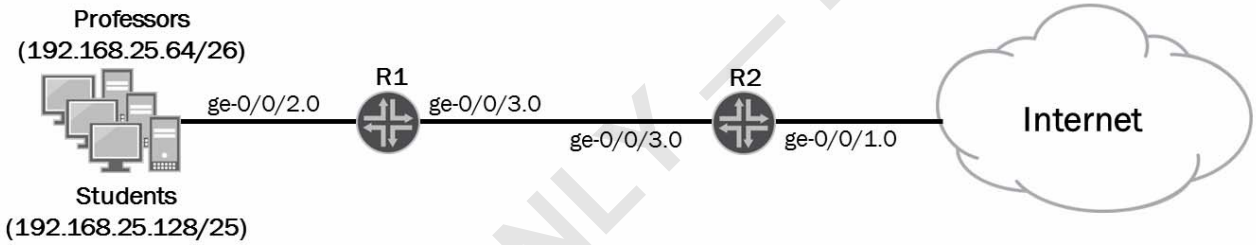
Case Study: Objectives and Topology

Case Study Objectives

Classification rules:
-Place traffic to and from professors (192.168.25.64/26) in the professors forwarding class.
-Place traffic to and from students (192.168.25.128/25) in the students forwarding class, provided the traffic from the students does not exceed 100 Mbps.
-Place traffic from the students in excess of 100 Mbps in the best-effort forwarding class.

Scheduling rules:
-Give network-control traffic high priority and 5% of the available bandwidth.
-Give traffic from professors medium-high priority and 45% of the bandwidth.
-Give traffic from students medium-low priority and 40% of the bandwidth.
-Give best-effort traffic low priority and 10% of the bandwidth. Prohibit use of extra bandwidth.

Forwarding class and queue mappings:
Queue 0 = best effort, Queue 1 = students, Queue 2 = professors, Queue 3 = network control



Objectives and Topology

The example on the slide shows a fairly simple topology with an across-the-network CoS application. We show the traffic classification and scheduling rules, as well as the forwarding classes and queue mappings.

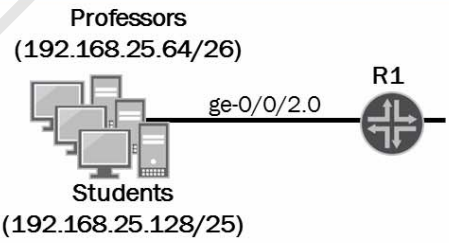
For simplicity, we use the same scheduling rules for all traffic on all interfaces. Note that once traffic is associated with queues other than Queue 0 and Queue 3, you must configure a scheduler for those queues on *all* interfaces. Failure to do so might lead to serious performance problems for traffic that exits an interface through a queue that has no defined scheduler.

Remember that CoS support varies between platforms running the Junos OS. In this example, R1 and R2 are J Series Service Routers.

Case Study: R1 Ingress Multifield Classifier

```
[edit firewall]
user@R1# show 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 {
            192.168.25.128/25;
        }
    }
    then {
        policer student-policer;
        forwarding-class students;
        accept;
    }
}
term default {
    then accept;
}
```

```
[edit firewall]
user@R1# show policer student-policer
if-exceeding {
    bandwidth-limit 100m;
    burst-size-limit 625k;
}
then forwarding-class best-effort;


[edit interfaces]
user@R1# show ge-0/0/2
unit 0 {
    family inet {
        filter {
            input apply-cos-markings;
        }
        address 192.168.25.1/24;
    }
}
```



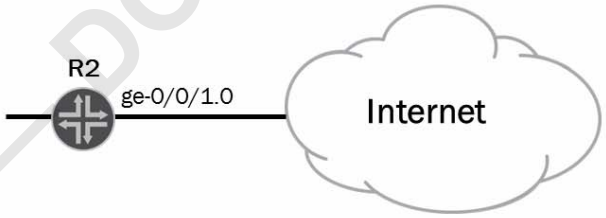
R1 Ingress Multifield Classifier Configuration

The slide shows the configuration necessary to implement the defined classification rules on R1.

Case Study: R2 Ingress Multifield Classifier

```
[edit firewall]
user@R2# show 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 {
            192.168.25.128/25;
        }
    }
    then {
        forwarding-class students;
        accept;
    }
}
term default {
    then accept;
}
```

```
[edit interfaces]
user@R2# show ge-0/0/1
unit 0 {
    family inet {
        filter {
            input apply-cos-markings;
        }
        address 172.22.13.2/30;
    }
}
```



R2 Ingress Multifield Classifier Configuration

The slide shows the configuration necessary to implement the specified classification rules on R2.

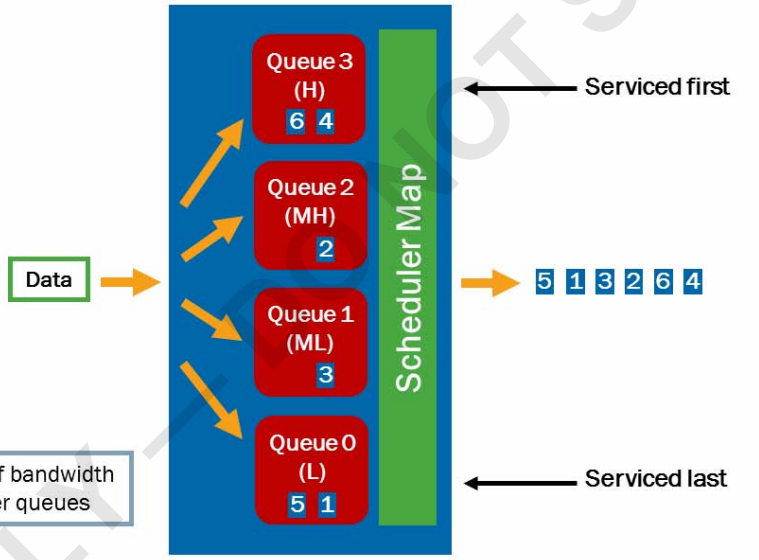
Case Study: Forwarding Class and Scheduler

```
[edit class-of-service]
user@R1# show forwarding-classes
queue 1 students;
queue 2 professors;
```

```
[edit class-of-service]
user@R1# show schedulers
sched-network-control {
    transmit-rate percent 5;
    buffer-size percent 5;
    priority high;
}
sched-professors {
    transmit-rate percent 45;
    buffer-size percent 45;
    priority medium-high;
}
sched-students {
    transmit-rate percent 40;
    buffer-size percent 40;
    priority medium-low;
}
sched-best-effort {
    transmit-rate percent 10 exact;
    buffer-size percent 10;
    priority low;
}
```

Prohibits the use of bandwidth assigned to other queues

```
[edit class-of-service]
user@R1# show 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;
}
```



Note: Forwarding class and scheduler configuration is identical on R1 and R2.

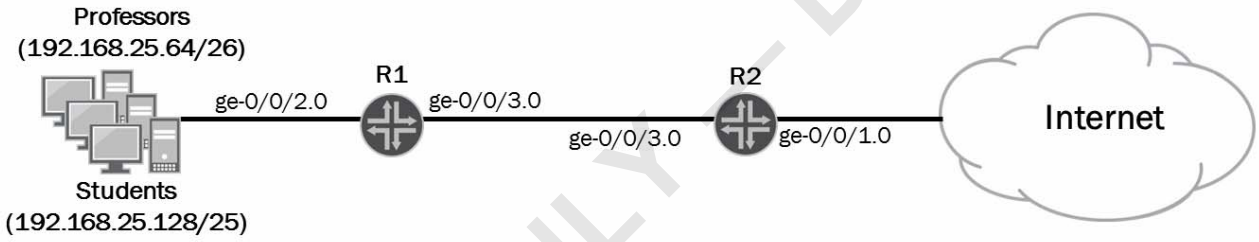
Forwarding Class and Scheduler Configuration

The slide shows the forwarding class, scheduler, and scheduler map configuration (for both routers).

Case Study: BA and Scheduler Application

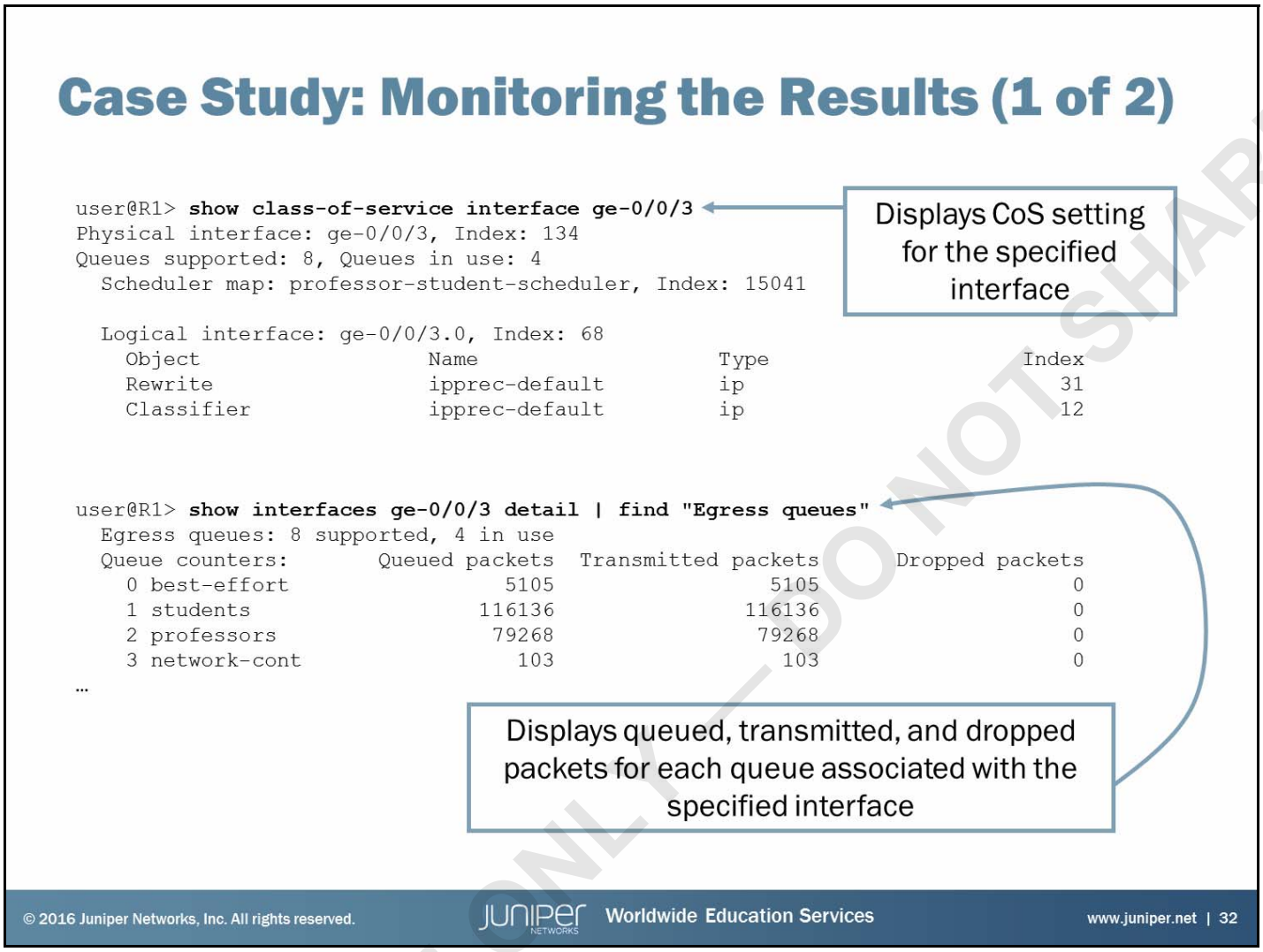
```
[edit class-of-service]
user@R1# show interfaces
ge-0/0/2 {
    scheduler-map professor-student-scheduler;
}
ge-0/0/3 {
    scheduler-map professor-student-scheduler;
    unit 0 {
        classifiers {
            inet-precedence default;
        }
        rewrite-rules {
            inet-precedence default;
        }
    }
}
```

```
[edit class-of-service]
user@R2# show interfaces
ge-0/0/1 {
    scheduler-map professor-student-scheduler;
}
ge-0/0/3 {
    scheduler-map professor-student-scheduler;
    unit 0 {
        classifiers {
            inet-precedence default;
        }
        rewrite-rules {
            inet-precedence default;
        }
    }
}
```



BA and Scheduler Application

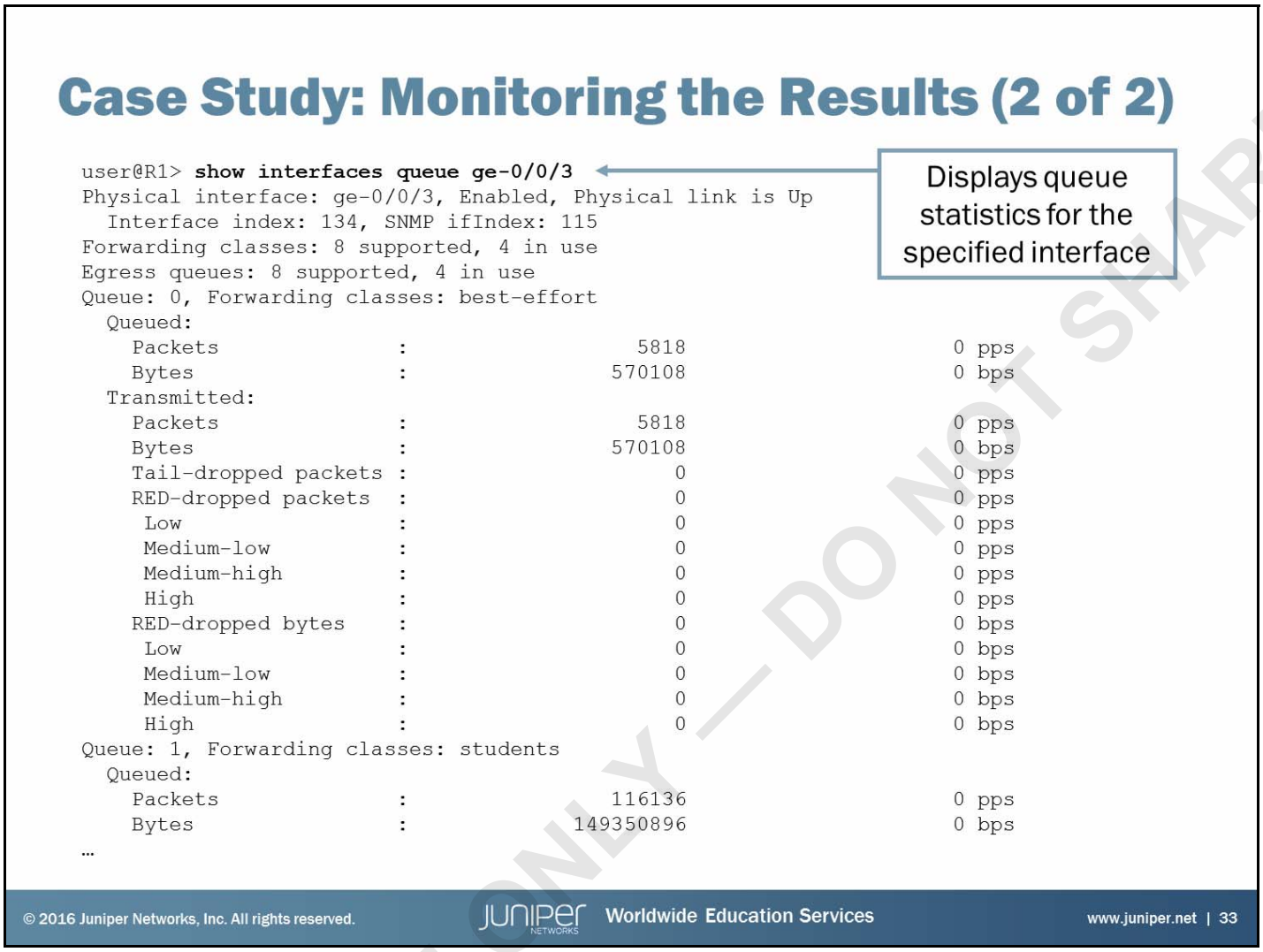
The slide shows the BA classifier and rewrite application as well as the scheduler map application (for both routers).



Monitoring the Results: Part 1

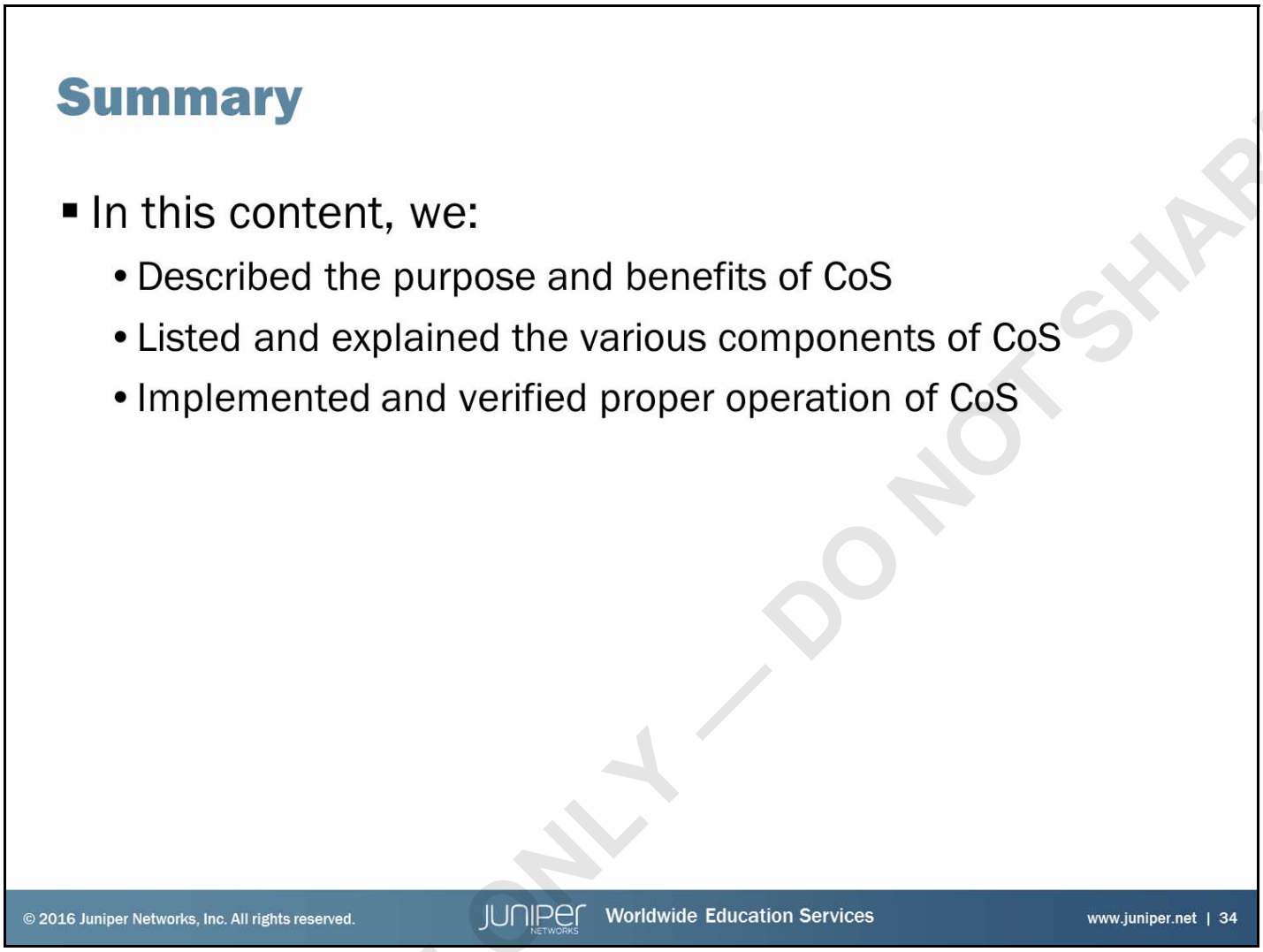
You can quickly see most of the CoS configuration for an interface using the **show class-of-service interface *interface*** command. This command displays the active configuration, including default values. You can use other **show class-of-service** commands to display details about various components 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 ensure that traffic is 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.



Monitoring the Results: Part 2

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. Based on this slide and the previous slide, we can see that traffic is being queued correctly.



We Discussed:

- The purpose and benefits of CoS;
- Components used with CoS; and
- Implementation and verification of CoS components.

Review Questions

1. How can CoS meet the performance requirements of a network?
2. What is the difference between a multifield classifier and a behavior aggregate classifier? When should you use each?
3. How are forwarding classes related to queues?
4. What is the difference between a scheduler and a scheduler map?

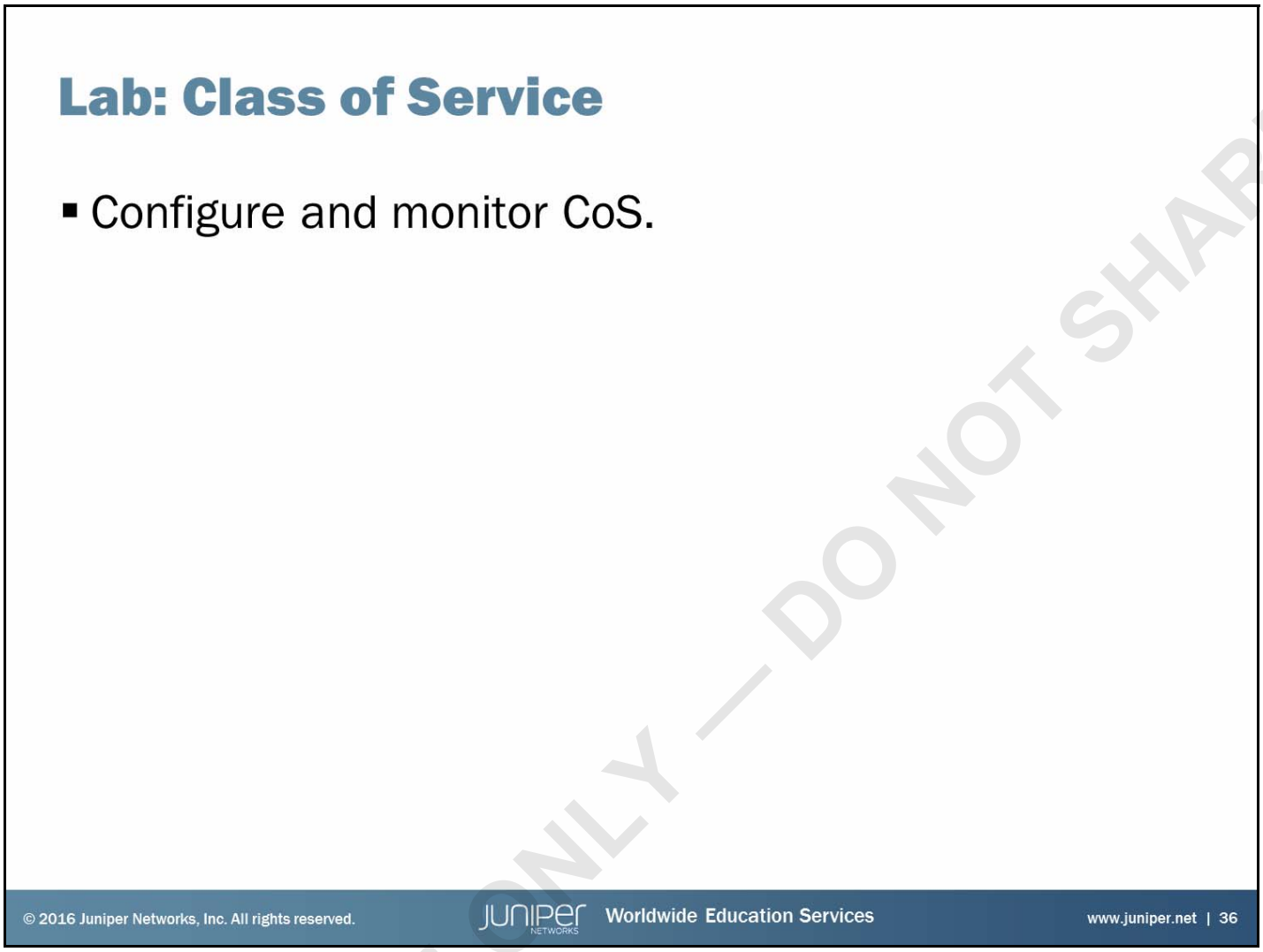
Review Questions

1.

2.

3.

4.



Lab: Class of Service

The slide lists the objective for this lab.

Answers to Review Questions

1.

CoS can meet the performance requirements of a network by prioritizing latency-sensitive traffic, such as VoIP; controlling congestion to ensure SLA maintenance; and allocating bandwidth for different classes of traffic.

2.

You implement a multifield classifier in a firewall filter and it can classify traffic based on any of the match conditions available in firewall filters. A behavior aggregate classifier classifies traffic based on specific behavior aggregate markings in packet headers. You typically use multifield classifiers on the edge of the network to initially classify traffic; you typically use behavior aggregate markings to maintain classification within the network.

3.

As the Junos OS processes traffic, it classifies the traffic into a forwarding class. When the traffic arrives at the output interface, the software places it into an appropriate queue based on the forwarding class. On some hardware, a many-to-one relationship can exist between forwarding classes and queues. Therefore, keeping the two concepts distinct is important.

4.

A scheduler defines CoS parameters for queue servicing. A scheduler map associates these parameters with a particular queue. You can then configure the Junos device to use scheduler maps for particular interfaces or units.

Acronym List

ACL	access control list
AS	autonomous system
BA	behavior aggregate
BOOTP	Bootstrap Protocol
CLI	command-line interface
CoS	class of service
DSCP	DiffServ code point
EGP	exterior gateway protocol
FBF	filter-based forwarding
GUI	graphical user interface
ICMP	Internet Control Message Protocol
IGP	interior gateway protocol
JNCP	Juniper Networks Certification Program
LSA	link-state advertisement
LSDB	link-state database
MBGP	multicast Border Gateway Protocol
MSDP	Multicast Source Discovery Protocol
OSI	Open Systems Interconnection
PFE	Packet Forwarding Engine
RE	Routing Engine
RED	random early detection
rpd	routing protocol process
RPF	reverse path forwarding
SPF	shortest path first
VoIP	voice over IP
VPN	virtual private network