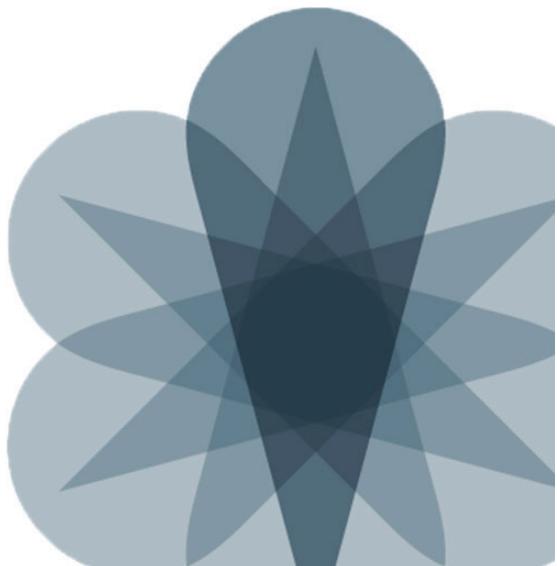


JNCIS-SP Study Guide—Part 1



Worldwide Education Services

1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA
408-745-2000
www.juniper.net



This document is produced by Juniper Networks, Inc.

This document or any part thereof may not be reproduced or transmitted in any form under penalty of law, without the prior written permission of Juniper Networks Education Services.

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

>B 7-GIGD'Gh Xmt i JXYf DUfh%

Copyright © 2013 Juniper Networks, Inc. All rights reserved.

Printed in USA.

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 12.1R1.9. 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.

This study guide is aligned with the related Juniper Networks Certification track and is an approved resource for your exam preparation.

Please go to the Juniper Networks Certification Program webpage (www.juniper.net/certification) to:

- Learn more about the Juniper Networks certification program
- See the certification tracks
- Get detailed exam descriptions
- Find additional exam preparation materials
- Register for an exam



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

Chapter 1: Protocol-Independent Routing	1-1
Chapter 2: Load Balancing and Filter-Based Forwarding	2-1
Chapter 3: Open Shortest Path First	3-1
Chapter 4: Border Gateway Protocol	4-1
Chapter 5: IP Tunneling	5-1
Chapter 6: High Availability	6-1
Appendix A: IPv6	A-1
Appendix B: IS-IS	B-1
Appendix C: Routing Information Protocol	C-1

Overview

Welcome to the JNCIS-SP Study Guide—Part 1. The purpose of this guide is to help you prepare for your JN0-360 exam and achieve your JNCIS-SP credential. The contents of this document are based on the *Junos Intermediate Routing* (JIR) course. This study guide provides students with intermediate routing knowledge and configuration examples. The guide includes an overview of protocol-independent routing features, load balancing and filter-based forwarding, OSPF, BGP, IP tunneling, and high availability (HA) features. The guide covers configuring and monitoring the Junos OS and monitoring device operations. This study guide is based on Junos OS Release 12.1R1.9.

Document Conventions

CLI and GUI Text

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

Style	Description	Usage Example
Franklin Gothic	Normal text.	Most of what you read in the Lab Guide and Student Guide.
Courier New	Console text: <ul style="list-style-type: none">• Screen captures• Noncommand-related syntax GUI text elements: <ul style="list-style-type: none">• Menu names• Text field entry	commit complete Exiting configuration mode 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	No distinguishing variant.	Physical interface:fxp0 , Enabled
Normal GUI		View configuration history by clicking Configuration > History.
CLI Input	Text that you must enter.	lab@San_Jose> show route
GUI Input		Select File > Save, and type config.ini in the Filename field.

Defined and Undefined Syntax Variables

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

Style	Description	Usage Example
<i>CLI Variable</i>	Text where variable value is already assigned.	policy my-peers
<i>GUI Variable</i>		Click <i>my-peers</i> in the dialog.
<i>CLI Undefined</i>	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 <u>policy-name</u> . ping 10.0.<u>x.y</u>
<i>GUI Undefined</i>		Select File > Save, and type <u>filename</u> 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 *JNCIS-SP Study Guide—Part 1* was developed and tested using software Release 12.1R1.9. 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

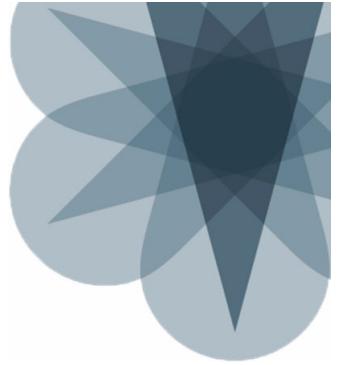
You can print technical manuals and release notes directly from the Internet in a variety of formats:

- 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).



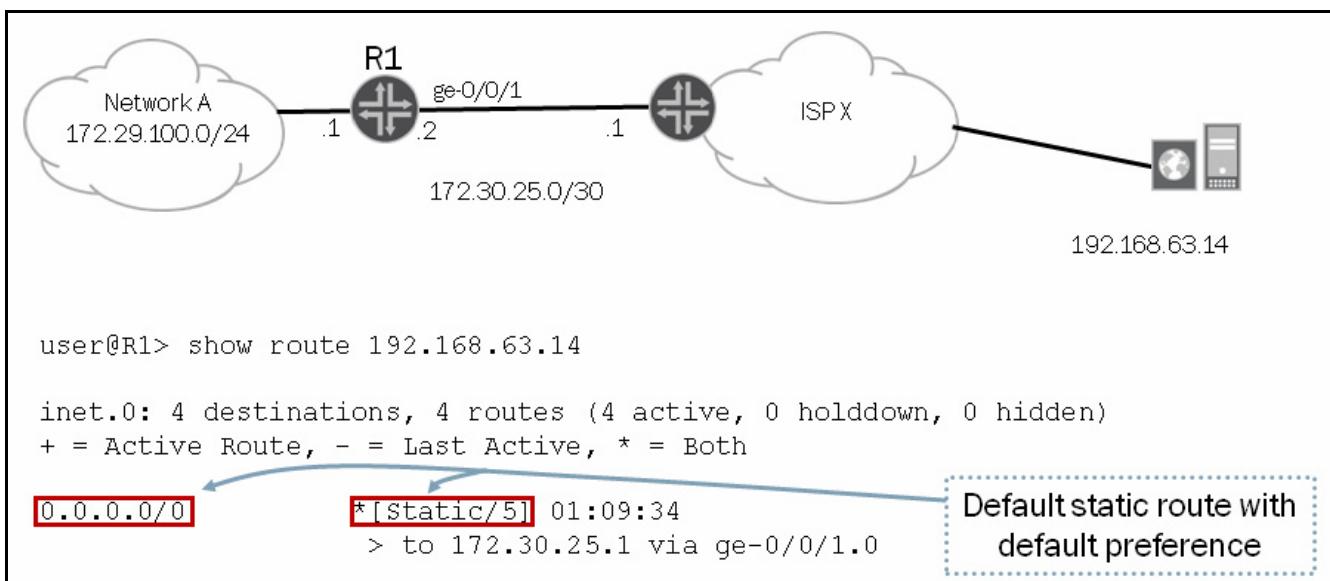
JNCIS-SP Study Guide—Part 1

Chapter 1: Protocol-Independent Routing

This Chapter Discusses:

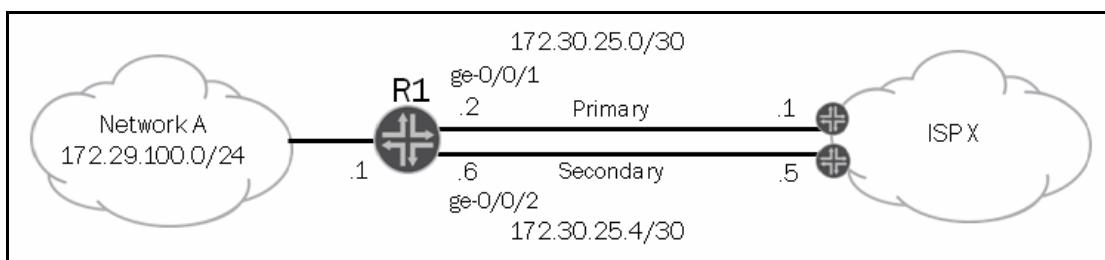
- Static, aggregate, and generated routes;
- Configuration and monitoring of static, aggregate, and generated routes;
- Martian routes and how to add new entries to the default list;
- Routing instances and their typical uses; and
- Configuration and sharing of routes between routing instances.

A Review of Static Routing



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.

Configuring Static Routes



All configuration for static routes occurs at the [edit routing-options] level of the hierarchy. A basic static route consists of a destination prefix and its associated next hop.

Static routes must have a valid next hop defined. Often that next-hop value is the IP address of a directly connected device through which the destination prefix is reachable. On point-to-point interfaces, you can specify the egress interface name rather than the IP address of the remote device as the next-hop value. You must specify an IP address as the next hop if the egress interface is an Ethernet interface.

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 operating system, 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 these two options is in the action the software takes after the drop action. If you specify **reject** as the next-hop value, the software 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 software does not send back an ICMP message; the system drops the packet silently.

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.

By default, the next-hop IP address of static routes must be reachable using a direct route. Unlike with software from other vendors, the Junos OS does not perform recursive lookups of next hops by default. You can override this default behavior using the **resolve** option.

You can include a number of optional parameters when configuring static routes, such as the **qualified-next-hop** and **resolve** options.

The **qualified-next-hop** option allows independent preferences for static routes to the same destination. The sample configuration corresponds with the diagram, which uses the **qualified-next-hop** option.

```
[edit routing-options]
user@R1# show
static {
    route 0.0.0.0/0 {
        next-hop 172.30.25.1;
        qualified-next-hop 172.30.25.5 {
            preference 7;
        }
    }
}
```

In the sample configuration shown, 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 default static route will then use the 172.30.25.5 next hop. Some vendors refer to this implementation as a *floating static route*.

When defining static routes, you can include the **next-table** option with a designated table name. Using the **next-table** option directs the matching traffic to the specified table where a second route lookup performed. The referenced route table must exist for the commit operation to succeed. Note that the **next-table** option is not supported on all Junos devices. Consult your product-specific documentation for support information.

Configuration Example: Static Routing

```
[edit routing-options]
user@R1# show
rib inet6.0 {
    static {
        route 0::/0 {
            next-hop 3001::1;
            preference 250;
        }
    }
    static {
        defaults {
            preference 250;
        }
        route 0.0.0.0/0 {
            next-hop 172.30.25.1;
            qualified-next-hop 172.30.25.5 {
                preference 251;
            }
        }
        route 172.28.102.0/24 {
            next-hop 10.210.11.190;
            no-readvertise;
        }
    }
}
```

Annotations:

- IPv6 default static route**: Points to the first static route definition for IPv6.
- Parameters defined under defaults section are applied to IPv4 static routes that do not include explicit definitions**: Points to the 'defaults' section under the IPv4 static block.
- IPv4 default static route with secondary qualified next hop**: Points to the 'qualified-next-hop' section of the IPv4 static route.
- Restricts route from being advertised through routing policy; highly suggested for static routes used for management traffic**: Points to the 'no-readvertise' option at the end of the IPv4 static route definition.

The inset illustrates the basic configuration syntax for IPv4 and IPv6 static routes. The inset also highlights the **no-readvertise** option, which prohibits the reference route from being redistributed through routing policy into a dynamic routing protocol such as OSPF. We highly suggest that you use the **no-readvertise** option on static routes that direct traffic out the management Ethernet interface and through the management network.

Within the [edit routing-options static] configuration hierarchy, the **defaults** section can contain static route options. Any options configured within this section are applied to all static routes on the device. In the example in the inset, we changed the preference value to 250. The software applies this preference value to all IPv4 static routes that do not explicitly have a preference value defined. Additional options you can assign to static routes include the following:

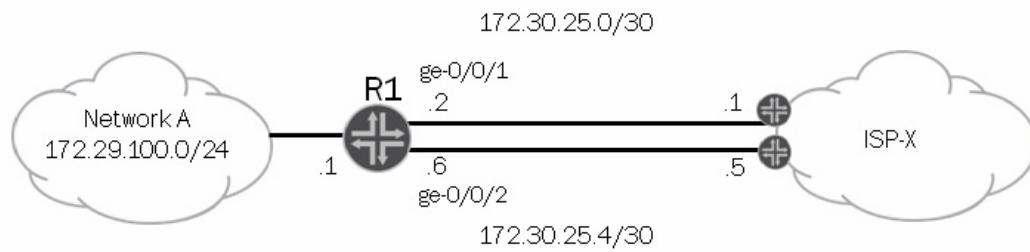
- **as-path**: Used if this route is intended to be redistributed into BGP and you want to add values manually to the AS path attribute.
- **community**: Used if this route is intended for BGP, and you want to add community values to the route for use in your AS.
- **metric**: If multiple routes share the same preference value, the route with the best metric becomes active in the routing table. Use this value to prefer one route over another in this case.
- **preference**: The default preference value of static routes is 5. This preference makes them more likely to be active than OSPF, IS-IS, or BGP for matching prefixes. Use this option to increase the value of the static routes to prefer other sources of routing information.

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

Test Your Knowledge: Part 1

■ Which next hop will be used in the example?

```
[edit routing-options]
user@R1# show
static {
    defaults {
        preference 180;
    }
    route 0.0.0.0/0 {
        next-hop 172.30.25.1;
        qualified-next-hop 172.30.25.5 {
            preference 7;
        }
    }
}
```



The inset illustrates a sample configuration and topology along with a thought-provoking question used to test your knowledge.

As previously mentioned, when a preference value is defined under the [edit routing-options static defaults] hierarchy level, it is assigned to all static routes that do not have an explicitly defined preference value. In our example, the next hop 172.30.25.1 uses a route preference of 180 based on the definition under the [edit routing-options static defaults] hierarchy level, and the qualified next hop 172.30.25.5 uses a route preference of 7, which is explicitly defined. With this knowledge, we know that the qualified next hop 172.30.25.5 will be selected as the next hop based on its lower route preference.

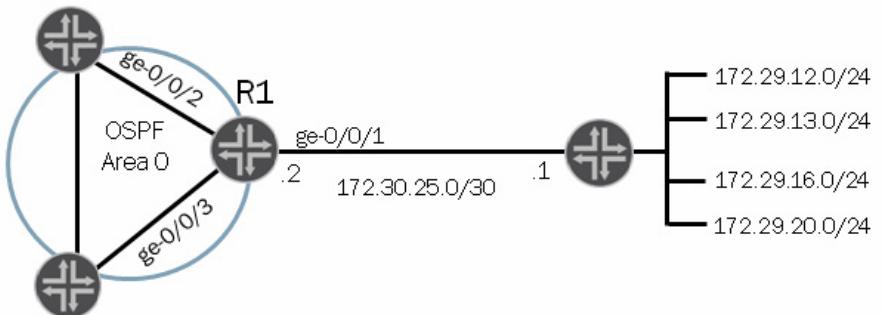
Test Your Knowledge: Part 2

■ Which static routes will be exported into OSPF?

```
[edit routing-options]
user@R1# show
static {
    route 172.29.12.0/24 next-hop 172.30.25.1;
    route 172.29.13.0/24 {
        next-hop 172.30.25.1;
        no-readvertise;
    }
    route 172.29.16.0/24 next-hop 172.30.25.1;
    route 172.29.20.0/24 next-hop 172.30.25.1;
}
```

```
[edit protocols ospf]
user@R1# show
export export2ospf;
area 0.0.0.0 {
    interface ge-0/0/2.0;
    interface ge-0/0/3.0;
    interface lo0.0;
}
```

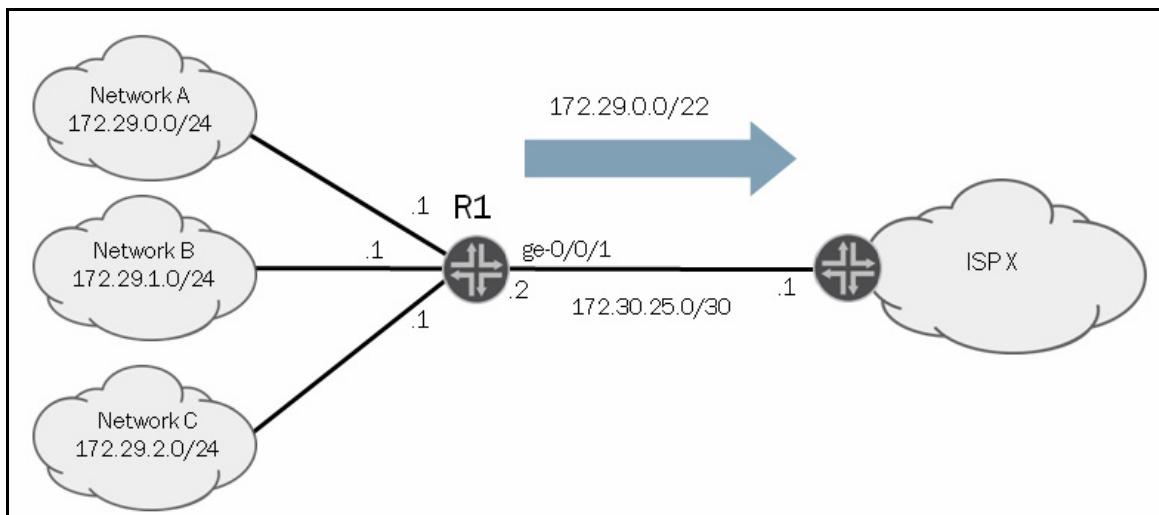
```
[edit policy-options]
user@R1# show
policy-statement export2ospf {
    term match-static-routes {
        from {
            protocol static;
            route-filter 172.29.0.0/16 orlonger;
        }
        then accept;
    }
}
```



The inset illustrates a sample configuration and topology along with a thought-provoking question used to test your knowledge.

As previously mentioned, the software does not allow any static routes with the no-readvertise option to be advertised through an export policy. In our example, the static route for the 172.29.13.0/24 destination uses the no-readvertise option, which excludes it from being exported into OSPF through the referenced export policy even though the route meets the match criteria. The software matches all other static routes in this example, and exports them into OSPF through the export policy.

Aggregate Routes



Aggregate routes allow you to combine groups of routes with common addresses into a single entry. By combining routes into a single entry, you can decrease the number of route advertisements sent by your device, thus decreasing the size of the routing tables maintained by neighboring devices. A second advantage of advertising a single route prefix that represents all other internal route prefixes is that internal routing instabilities can be hidden from external peers.

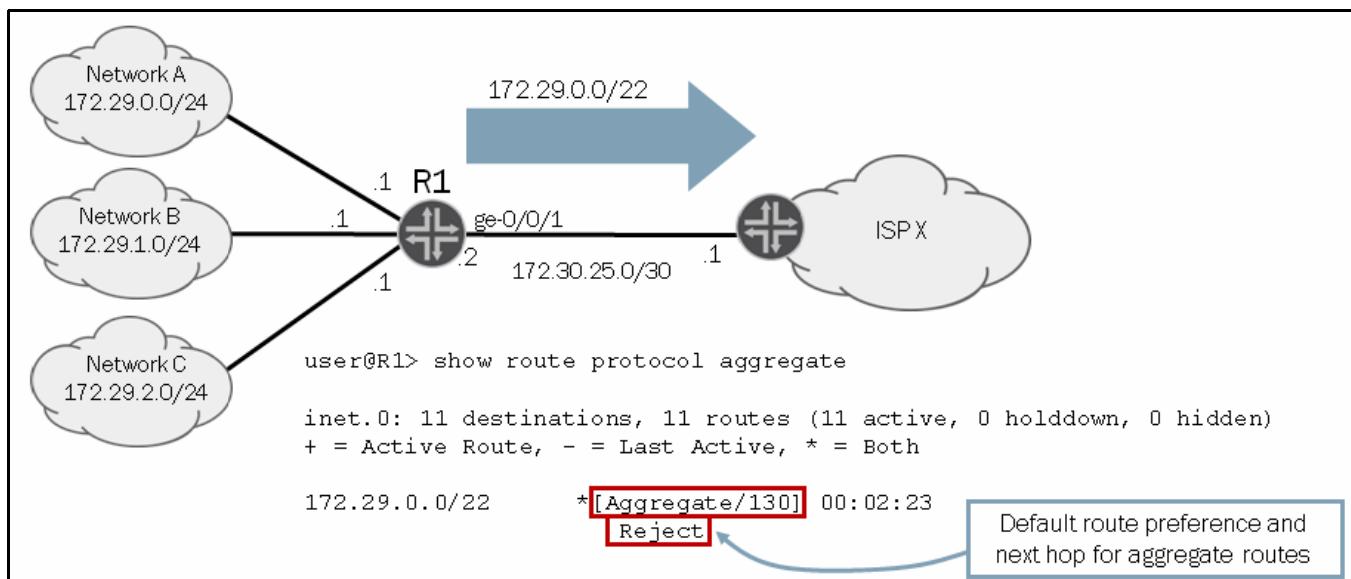
Think About It

- Which of the prefixes are part of the 10.1.0.0/20 aggregate route?
 - 10.1.14.0/24 ✓
 - 10.1.15.0/24 ✓
 - 10.1.16.0/24 ✗
 - 10.1.17.0/24 ✗



This graphic presents a basic route aggregation scenario to identify which prefixes belong to the 10.1.0.0/20 aggregate route. In this example, only the top two prefixes (10.1.14.0/24 and 10.1.15.0/24) belong to the 10.1.0.0/20 aggregate route. If you wanted to include all of the listed prefixes in a single aggregate route, you could use the 10.1.0.0/19 prefix as the aggregate route.

Configuring Aggregate Routes



You configure aggregate routes at the [edit routing-options] hierarchy level. An aggregate route consists of a destination summary prefix that represents more specific routes, also known as contributing routes. Aggregate routes become active in the routing table when at least one of the contributing routes for the aggregate is also active in the routing table.

By default, the software assigns aggregate routes a route preference of 130 and a next-hop value of `reject`. If the system advertising the aggregate route receives a packet destined to an IP address within the aggregate route but a more specific route does not exist, the software drops the packet and sends an ICMP message back to the source device. You can modify the default route preference and next-hop type.

Configuration Example: Aggregate Routes

```
[edit routing-options]
user@R1# show
aggregate {
    defaults {
        community 1:888;
    }
    route 172.29.0.0/22;
    route 172.25.0.0/16 {
        community 1:999;
        discard;
    }
}
```

Parameters defined under `defaults` section are applied to aggregate routes that do not include explicit definitions

`discard` next hop silently discards packets destined to a contributing subnet for which a more specific route entry does not exist

This inset illustrates a sample aggregate route configuration. Within the `[edit routing-options aggregate]` configuration hierarchy, the `defaults` section can contain aggregate route options. Any options configured within this section are applied to all aggregate routes that do not have those options explicitly defined. In the example, the `172.25.0.0/16` aggregate route will use its explicitly defined community value (`1:999`) and the `172.29.0.0/22` aggregate route will use the community value defined under the `defaults` section (`1:888`). Note that you can configure only one aggregate route per prefix.

The default next-hop value for aggregate routes is `reject`. As with static routes, when the `reject` next-hop value is used, the software drops the packet from the network and sends an ICMP network unreachable message back to the source of the IP packet. The other possible next-hop value for aggregate routes is `discard`. When the `discard` option is used, the software drops the packet silently and does not send an ICMP network unreachable message back to the source device.

Some options you can assign to aggregate routes include the following:

- `as-path`: Use this option if this route is intended to be redistributed into BGP, and you want to add values manually to the AS path attribute.
- `community`: Use this option if this route is intended for BGP, and you want to add community values to the route for use in your AS.
- `metric`: If multiple routes share the same preference value, the route with the best metric becomes active in the routing table. Use this value to prefer one route over another in this case.
- `policy`: By default, you can use all possible, more-specific contributing routes to activate an aggregate route. To alter this default, you can use a policy to accept or reject certain routes that should or should not be used.
- `preference`: The default preference value of aggregate routes is 130. Use this option to alter the value of aggregate routes.

Viewing the Contributing Routes

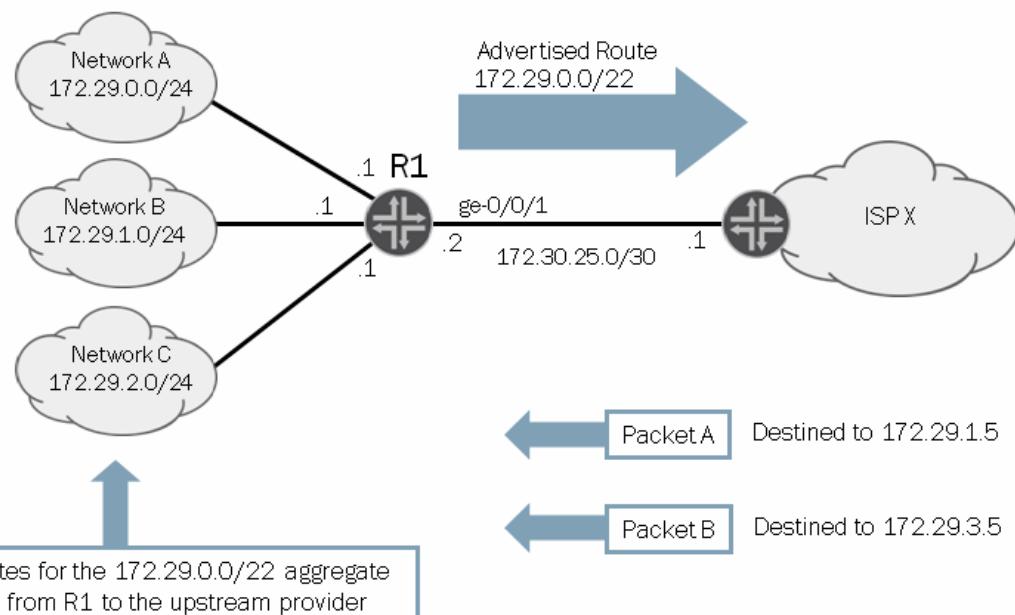
```
user@R1> show route 172.29.0.0/22 exact detail

inet.0: 12 destinations, 12 routes (11 active, 0 holddown, 1 hidden)
172.29.0.0/22 (1 entry, 1 announced)
  *Aggregate Preference: 130
    Next hop type: Reject
    Next-hop reference count: 2
    State: <Active Int Ext>
    Age: 1:39:21
    Task: Aggregate
    Announcement bits (1): 0-KRT
    AS path: I (LocalAgg)
    Flags: Depth: 0 Active
    AS path list:
    AS path: I Refcount: 3
    Contributing Routes (3):
      Contributing routes { 172.29.0.0/24 proto Direct
                           172.29.1.0/24 proto Direct
                           172.29.2.0/24 proto Direct}
```

As shown in the inset, the **show route *prefix* exact detail** command displays the summary route and all currently contributing routes. In addition to listing the prefixes contributing actively, you can also see which protocol placed that contributing route into the routing table. In the example, the aggregate route 172.29.0.0/20 is active in the routing table and has three current contributing routes. Note that other variations of the **show route** command also display similar information.

Test Your Knowledge

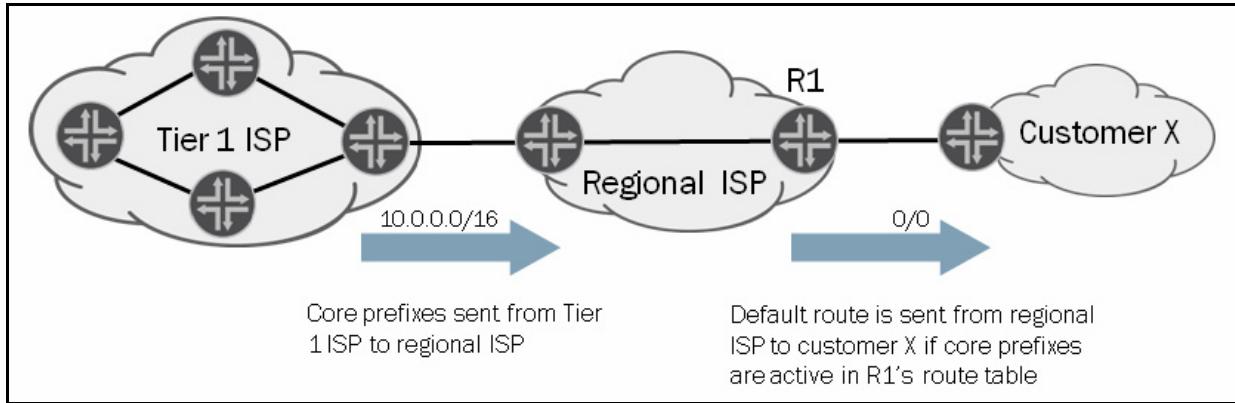
- What action does R1 take when it receives packet A and packet B (assume the default next-hop behavior)?



The graphic illustrates a sample topology and some associated details along with a thought-provoking question used to test your knowledge.

In our sample scenario, the destination address associated with Packet A matches one of the contributing routes (172.29.1.0/24). Packet A is forwarded by the software to its destination using the forwarding table entry related to matching contributing route. Although the destination address associated with Packet B matches the aggregate route, it does not match any of the contributing routes for the aggregate route. Because packet B does not match any of the contributing routes, the software drops the packet and generates and sends ICMP network unreachable messages back to the source device. The described behavior assumes the default next-hop type of reject.

Generated Routes



Generated routes are similar in nature and configuration to aggregate routes. Like aggregate routes, generated routes become active in the routing table when at least one contributing route (more specific route) for the generated route is also active in the routing table.

The output of **show route** commands displays generated routes as aggregate routes, as shown in the following sample output:

```
user@R1> show route 0/0 exact detail

inet.0: 14 destinations, 14 routes (14 active, 0 holddown, 0 hidden)
0.0.0.0/0 (1 entry, 1 announced)
  *Aggregate Preference: 130
    Next hop type: Router, Next hop index: 546
    Next-hop reference count: 4
    Next hop: 172.30.25.1 via ge-0/0/1.100, selected
    State: <Active Int Ext>
    Local AS: 65400
    Age: 1:03:46
    Task: Aggregate
    Announcement bits (2): 0-KRT 2-OSPF
    AS path: I
      Flags: Generate Depth: 0          Active
      Contributing Routes (1):
        10.0.0.0/16 proto BGP
```

The main difference between an aggregate route and a generated route is that a generated route receives the next hop of the primary contributing route. The primary contributing route is the route with lowest route preference that falls within the aggregated range of prefixes. If there are multiple routes that fall within the aggregated range that share the same route preference, the route with the lowest number prefix, not the lowest prefix length, is selected as the primary contributing route.

For a route to qualify as a contributing route for a generated route, the route must have a valid forwarding next hop other than the local device; otherwise the generated route becomes hidden, as shown in the following sample output:

```
user@R1> show route hidden

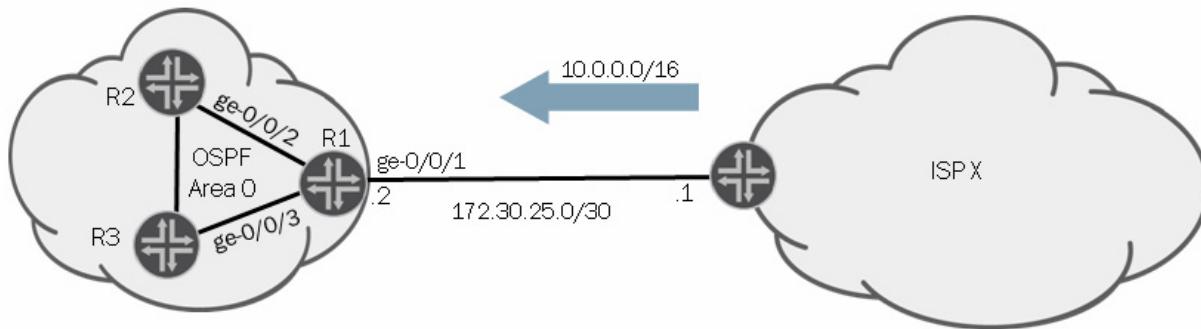
inet.0: 7 destinations, 7 routes (6 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          [Aggregate] 00:17:30
                           Reject
```

A generated route is often referred to as a *route of last resort*. This reference is because of one of the possible uses of generated routes, which is to source a default route when specific conditions are met. You can define the required conditions through routing policy. We show an example of how to use routing policy to define the required conditions for a generated route later in this section.

Case Study: Topology and Objective

- Configure R1 to generate and advertise a default route into OSPF when the 10.0.0.0/16 BGP route is present in its routing table

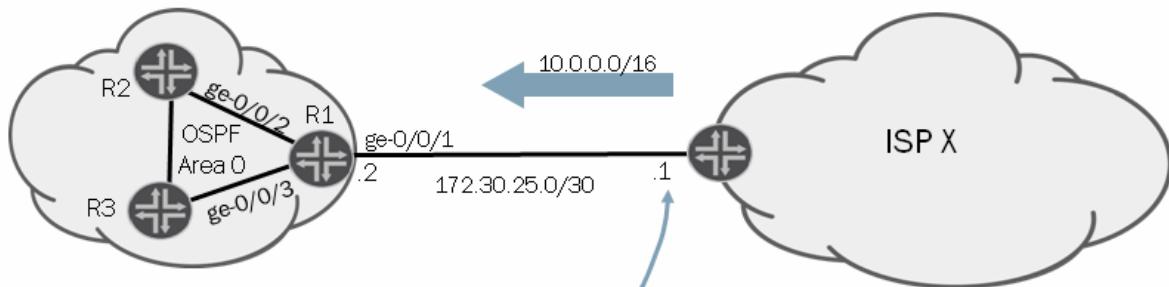


The graphic provides the objective and sample topology used in this case study.

Case Study: Creating the Generated Route

```
[edit routing-options]
user@R1# show
generate {
    defaults {
        preference 130;
    }
    route 0.0.0.0/0;
}
```

Parameters defined under **defaults** section are applied to generated routes that do not include explicit definitions



Note: The default next hop for generated routes is the next hop associated with the primary contributing route. In our example, this default next hop will be 172.30.25.1.

The graphic provides a sample generated route configuration and some relevant notes.

Case Study: Defining the Required Policy

```
[edit policy-options]
user@R1# show policy-statement match-contributing-prefix
term match-bgp-prefix {
    from {
        protocol bgp;
        route-filter 10.0.0.0/16 exact;
    }
    then accept;
}
term else-reject {
    then reject;
}
```

Policy matches 10.0.0.0/16 prefix exactly and rejects all other potential contributors

```
[edit policy-options]
user@R1# show policy-statement export-default
term match-default {
    from {
        protocol aggregate;
        route-filter 0.0.0.0/0 exact;
    }
    then accept;
}
```

Policy matches generated route defined on the previous slide

The protocol aggregate match condition is used for aggregate and generated routes

The graphic provides two sample routing policies. The first routing policy, named *match-contributing-prefix*, matches the desired contributing prefix (10.0.0.0/16) exactly and rejects all other potential contributing routes. In this example, all prefixes with a forwarding next hop that is not the local device are contributing routes because we are using a default generated route (0.0.0.0/0).

The second routing policy, named *export-default*, matches the generated route defined in the previous graphic. Note the use of the *protocol aggregate* match condition, which is used for both aggregate and generated routes.

Case Study: Applying the Required Policy

```
[edit routing-options]
user@R1# show
generate {
    defaults {
        preference 130;
    }
    route 0.0.0.0/0 policy match-contributing-prefix;
}
```

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

The *export-default* policy exports the default generate route only when the required condition is met.

This inset shows the application of the policies defined on the previous page. We applied the *match-contributing-prefix* policy directly to the 0.0.0.0/0 generated route, and we applied the *export-default* policy as an export policy under the [edit protocols ospf] hierarchy level.

Case Study—Monitoring the Results: Part 1

```
user@R1> show route 0/0 exact detail

inet.0: 14 destinations, 14 routes (14 active, 0 holddown, 0 hidden)
0.0.0.0/0 (1 entry, 1 announced)
  *Aggregate Preference: 130
    Next hop type: Router, Next hop index: 546
    Next-hop reference count: 4
    Next hop: 172.30.25.1 via ge-0/0/1.100, selected
    State: <Active Int Ext>
    Local AS: 65400
    Age: 1:03:46
    Task: Aggregate
    Announcement bits (2): 0-KRT 2-OSPF
    AS path: I
          Flags: Generate Depth: 0      Active
    Contributing Routes (1):
      10.0.0.0/16 proto BGP
```

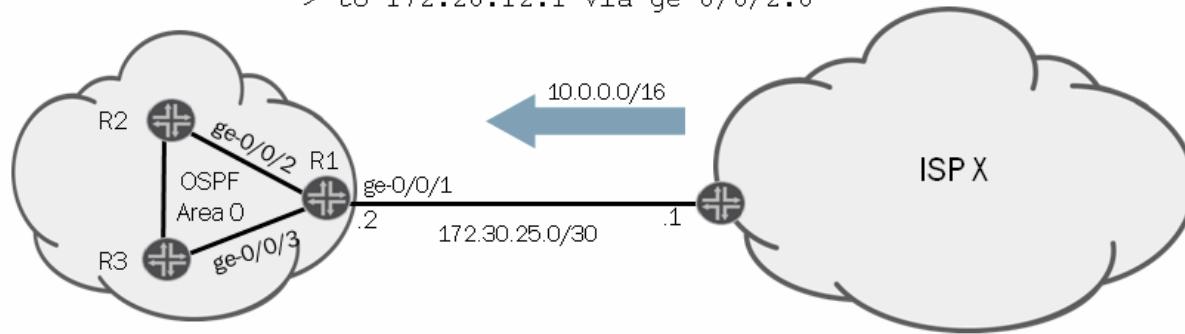
The inset shows one verification step performed on the R1 device. Here we see the output from the **show route 0/0 exact detail** command, which shows the generated route along with its related details, including the 10.0.0.0/16 contributing route.

Case Study—Monitoring the Results: Part 2

```
user@R2> show route 0/0 exact

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

0.0.0.0/0          *[OSPF/150] 00:15:30, metric 0, tag 0
                  > to 172.20.12.1 via ge-0/0/2.0
```



The graphic shows a secondary verification step performed on the R2 device. Here we see that R2 installed an external OSPF route for the 0.0.0.0/0 prefix. Based on this output, we can conclude that the export policy applied at the [export protocols ospf] hierarchy level is functioning correctly.

Overview: Martian Addresses

Martian addresses are host or network addresses for which all routing information is ignored. These addresses are never installed in the routing table and are typically sent by improperly configured systems on the network and have destination addresses that are obviously invalid.

In IPv4, the following are the default Martian addresses:

- 0.0.0.0/8
- 127.0.0.0/8
- 128.0.0.0/16
- 191.255.0.0/16
- 192.0.0.0/24
- 223.255.255.0/24
- 240.0.0.0/4

In IPv6, the loopback address, the reserved and unassigned prefixes from RFC 2373, and the link-local unicast prefix are the default Martian addresses. We focus on IPv4 in this section.

Adding Additional Prefixes

```
[edit routing-options]
user@R1# show martians
23.0.0.0/8 orlonger;
31.0.0.0/8 orlonger;
36.0.0.0/8 orlonger;
```

Specify prefix and match-type

Within the [edit routing-options martians] configuration hierarchy, you can specify additional prefixes. In addition to the prefix and bit mask, you must also include a match type, which is a policy control that determines which exact routes are covered by the configuration statement. The six default match types are the following:

- exact;
- longer;
- orlonger;
- prefix-length-range;
- through; and
- upto.

Identifying Martian Addresses

```
user@R1> show route martians

inet.0:
...
23.0.0.0/8 orlonger -- disallowed
31.0.0.0/8 orlonger -- disallowed
36.0.0.0/8 orlonger -- disallowed
```

Prefixes are added to list

Default list omitted

To see the current list of Martian addresses for all routing tables, use the **show route martians** command. You can filter the generated output to display the list for a specific table by adding the **table** option and specifying the desired table name as shown in the following sample output:

```
user@R1> show route martians table inet.0

inet.0:
    0.0.0.0/0 exact -- allowed
    0.0.0.0/8 orlonger -- disallowed
    127.0.0.0/8 orlonger -- disallowed
    128.0.0.0/16 orlonger -- disallowed
    191.255.0.0/16 orlonger -- disallowed
```

```
192.0.0.0/24 orlonger -- disallowed
223.255.255.0/24 orlonger -- disallowed
240.0.0.0/4 orlonger -- disallowed
23.0.0.0/8 orlonger -- disallowed
31.0.0.0/8 orlonger -- disallowed
36.0.0.0/8 orlonger -- disallowed
```

You use the **allow** command to remove IP address blocks from the Martian address list as shown in the following example:

```
[edit routing-options]
user@R1# set martians 240/4 orlonger allow

[edit routing-options]
user@R1# commit and-quit
commit complete
Exiting configuration mode

user@R1> show route martians table inet.0

inet.0:
    0.0.0.0/0 exact -- allowed
    0.0.0.0/8 orlonger -- disallowed
    127.0.0.0/8 orlonger -- disallowed
    128.0.0.0/16 orlonger -- disallowed
    191.255.0.0/16 orlonger -- disallowed
    192.0.0.0/24 orlonger -- disallowed
    223.255.255.0/24 orlonger -- disallowed
    240.0.0.0/4 orlonger -- allowed
    23.0.0.0/8 orlonger -- disallowed
    31.0.0.0/8 orlonger -- disallowed
    36.0.0.0/8 orlonger -- disallowed
```

A Review of Routing Instances

Device Running the Junos OS

Routing Instance (master)	Routing Instance (cust-A)	Routing Instance (cust-B)
inet.0	cust-A.inet.0	cust-B.inet.0
inet6.0	cust-A.inet6.0	cust-B.inet6.0
ge-0/0/0.0	ge-0/0/3.0	ge-1/0/0.0
ge-0/0/1.0	ge-0/0/4.0	ge-1/0/1.0
lo0.0	lo0.1	lo0.2
Default Route	Default Route	Default Route
OSPF	OSPF	OSPF

The Junos OS logically groups routing tables, interfaces, and routing protocol parameters to form unique routing instances. The software 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

Instance	Type	
master	Primary RIB forwarding	Active/holddown/hidden
inet.0 inet6.0		3/0/1 4/0/0

Routing instance name

Participating route tables; the presence of inet6.0 table indicates IPv6 is in use

The Junos OS creates a default unicast routing instance named 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@R1> show route instance
Instance          Type
Primary RIB
__juniper_private1__ forwarding      Active/holddown/hidden
    __juniper_private1__.inet.0        2/0/2
    __juniper_private1__.inet6.0       1/0/0

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

__master.anon__     forwarding

master             forwarding      7/0/0
    inet.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.

Some typical uses of user-defined routing instances include filter-based forwarding (FBF), which is sometimes referred to as policy-based routing, 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

```
[edit routing-instances new-instance]
user@R1# show
instance-type virtual-router;
interface ge-0/0/0.0;
interface ge-0/0/1.0;
interface lo0.1;
routing-options {
    static {
        route 0.0.0.0/0 next-hop 172.26.25.1;
    }
}
protocols {
    ospf {
        area 0.0.0.0 {
            interface ge-0/0/0.0;
            interface ge-0/0/1.0;
            interface lo0.1;
        }
    }
}
```

Routing instance name is user-defined

Routing instance type

Define interfaces under the [edit interfaces] hierarchy and reference them under their respective routing instance

The inset illustrates a basic routing instance configuration example.

Working with Routing Instances

```
user@R1> show interfaces terse routing-instance new-instance
Interface          Admin Link Proto      Local                  Remote
ge-0/0/0.0          up     up   inet      172.26.25.5/24
ge-0/0/1.0          up     up   inet      172.25.182.5/24
lo0.1              up     up   inet      192.168.100.52      --> 0/0

user@R1> ping 172.26.25.1 rapid count 25 routing-instance new-instance
PING 172.26.25.1 (172.26.25.1): 56 data bytes
!!!!!!!
--- 172.26.25.1 ping statistics ---
25 packets transmitted, 25 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.014/1.875/2.073/0.285 ms

user@R1> show route table new-instance.inet.0
new-instance.inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 02:06:18
                  > to 172.26.25.1 via ge-0/0/0.0
```

Software automatically creates IP unicast table

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.

You can filter many of the common outputs generated through the command-line interface (CLI) **show** commands by referencing the name of a given routing instance. The first example in the inset 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 next example in the inset shows this option in action with the **ping** utility.

To view a routing table associated with a specific routing instance, you simply use the **show route table table-name** CLI command, as shown on the last example in the inset. To view all routing instances on your Junos device, you can issue the **show route instance** command.

Sharing Routes Between Routing Tables

The Junos OS provides a simple method for placing routing information in multiple tables simultaneously. This method uses a routing information base (RIB) group. Once defined within the [edit routing-options] portion of the configuration hierarchy, you can apply a RIB group in other sections of the configuration.



Some deployment scenarios require networking devices, such as a router, to know multiple routing topologies. Example scenarios include VPNs, multicast networks, and FBF. In a VPN network, multiple customer network topologies use a common infrastructure. These separate topologies require the networking devices to maintain multiple different routing tables. In a multicast environment, you might want unicast and multicast traffic to use different network links. These separate forwarding topologies require separate routing tables. Finally, the Junos OS feature of FBF allows for the forwarding of packets based on criteria other than the destination IP address within the packet. The forwarding address might be different than the current active route in the routing table. Again, a separate routing table is necessary.

Other options for sharing routes between routing instances exist. You can use the **instance-import**, **instance-export** and **auto-export** configuration options to share routes between multiple routing instances and eliminate the configuration of separate routing table groups for each instance. Usage guidelines and coverage of these configuration options are outside the scope of this guide.

Defining a RIB Group

```

[edit routing-options]
user@R1# show
rib-groups {
    <rib-group-name> {
        export-rib <routing-table-name>;
        import-rib <routing-table-names>;
        import-policy <policy-name>;
    }
}
  
```

The export-rib indicates table from where routes should be retrieved

The import-rib indicates tables in which routes should be placed

The import-policy is used to control which routes are installed in the routing table group

The Junos OS uses RIB groups to place route information into multiple tables on a Junos OS-based device. You assign the name of the RIB group within the [edit routing-options] configuration hierarchy. Within the rib-group configuration, two main configuration options exist. The import-rib statement can list multiple routing tables, and informs the software where to place incoming route information. The export-rib statement can list only a single routing table, and informs the software where to extract route information. The import-rib statement must list the primary routing table first within the configuration. The primary routing table is considered the routing table where the routing information would ordinarily be placed without the presence of a RIB group. Because you can list only a single routing table within the export-rib statement, and that single routing table must be the primary RIB, the export-rib statement is frequently omitted from the configuration.

RIB Group Application

```

[edit protocols ospf]
user@R1# set rib-group ?
Possible completions:
<rib-group>      Routing table group for importing OSPF routes
  
```

Once you create a RIB group, you can apply it to other sections of the configuration. Potential applications of a RIB group include interface routes, static routes, OSPF, IS-IS, RIP, BGP, Protocol Independent Multicast (PIM), and Multicast Source Discovery Protocol (MSDP).

A sample configuration that shares OSPF routes between the `inet.0` and `test.inet.0` routing tables is shown in the following sample output:

```
[edit routing-options]
user@R1# show
rib-groups {
    test {
        import-rib [ inet.0 test.inet.0 ];
    }
}
```

```
[edit protocols ospf]
user@R1# show
rib-group test;
area 0.0.0.0 {
    interface ge-0/0/1.0;
    interface lo0.0;
}
```

We can verify that the OSPF routes were shared (from `inet.0` to `test.inet.0`) using the `show route table` commands as shown in the following sample output:

```
user@R1> show route table inet.0 protocol ospf

inet.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

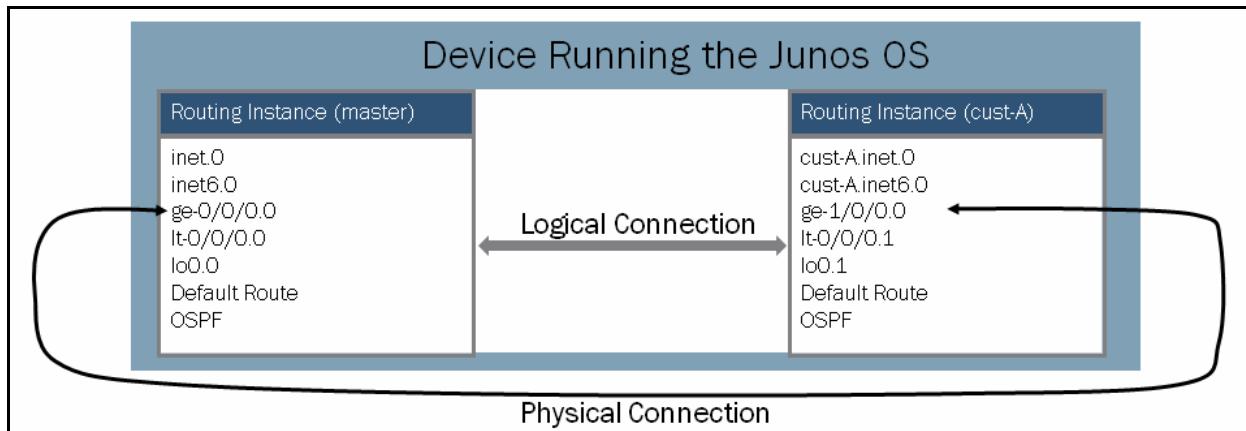
172.20.101.0/24      *[OSPF/150] 00:00:30, metric 0, tag 0
                      > to 172.20.77.2 via ge-0/0/1.0
172.20.201.0/24      *[OSPF/150] 00:00:30, metric 0, tag 0
                      > to 172.20.77.2 via ge-0/0/1.0
192.168.2.1/32       *[OSPF/10] 00:00:30, metric 1
                      > to 172.20.77.2 via ge-0/0/1.0
224.0.0.5/32          *[OSPF/10] 2w1d 02:37:55, metric 1
                      MultiRecv
```

```
user@R1> show route table test.inet.0 protocol ospf
```

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

172.20.101.0/24      *[OSPF/150] 00:00:27, metric 0, tag 0
                      > to 172.20.77.2 via ge-0/0/1.0
172.20.201.0/24      *[OSPF/150] 00:00:27, metric 0, tag 0
                      > to 172.20.77.2 via ge-0/0/1.0
192.168.2.1/32       *[OSPF/10] 00:00:27, metric 1
                      > to 172.20.77.2 via ge-0/0/1.0
224.0.0.5/32          *[OSPF/10] 00:00:27, metric 1
                      MultiRecv
```

Routing Between Instances



In addition to sharing routes between instances, you can also form logical or physical connections between instances on the same Junos device and route between the connected instances.

To connect two routing instances with a logical connection, you configure a logical tunnel interface for each instance. Then you configure a peer relationship between the logical tunnel interfaces, thus creating a point-to-point connection. To configure a point-to-point connection between two routing instances, you configure the logical tunnel interface using the **lt-fpc/pic/port** format. A sample logical tunnel interface configuration with two units—one for each instance—follows:

```
[edit interfaces lt-0/0/0]
user@R1# show
unit 0 {
    encapsulation ethernet;
    peer-unit 1;
    family inet {
    }
}
unit 1 {
    encapsulation ethernet;
    peer-unit 0;
    family inet;
}
```

Logical tunnels are not supported on all Junos devices. On some Junos devices that support logical tunnel interfaces, you must install the appropriate services PIC or services module. Refer to the platform specific documentation for your product for support details.

When configuring logical tunnel interfaces, note the following:

- You can configure each logical tunnel interface with one of the following encapsulation types: Ethernet, Ethernet circuit cross-connect (CCC), Ethernet VPLS, Frame Relay, Frame Relay CCC, VLAN, VLAN CCC, or VLAN VPLS.
- You can configure the IP, IPv6, International Organization for Standardization (ISO), or MPLS protocol family.
- The peering logical interfaces must belong to the same logical tunnel interface derived from the Tunnel Services PIC or Adaptive Services Module.
- You can configure only one peer unit for each logical interface. For example, unit 0 cannot peer with both unit 1 and unit 2.
- To enable the logical tunnel interface, you must configure at least one physical interface statement.

In addition to logical tunnel interfaces, you can also use physical interfaces to connect and route between routing instances. This implementation method typically requires two physical ports—one for each instance. You define the physical interfaces as you normally would and simply associate each interface with its respective routing instance under the [edit routing-instance instance-name] hierarchy.

Review Questions

1. Which configuration option allows unique preference values for static routes to the same destination?
2. What is the default next hop for aggregate and generated routes?
3. What is the purpose of the Martian address list?
4. List some common uses of routing instances.

Answers

1.

You can use the **qualified-next-hop** option to assign a unique route preference.

2.

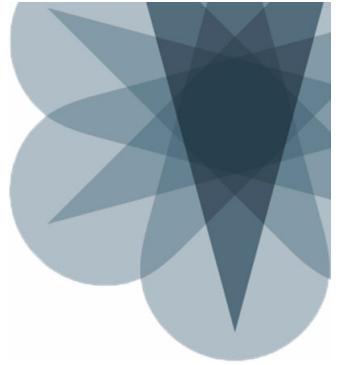
Aggregate routes have a default next hop of `reject`, whereas generated routes use the forwarding next hop of the primary contributing route.

3.

The Martian address list identifies prefixes that are nonroutable.

4.

Some common uses of routing instances include FBF, VPN services, and system virtualization.



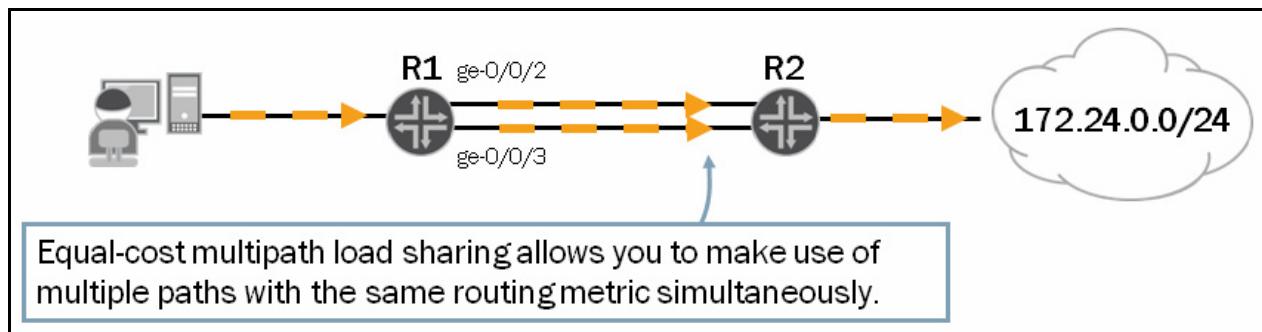
JNCIS-SP Study Guide—Part 1

Chapter 2: Load Balancing and Filter-Based Forwarding

This Chapter Discusses:

- Load-balancing concepts and operations;
- Implementation and monitoring of Layer 3 load balancing;
- Benefits of filter-based forwarding (FBF);
- Configuration and monitoring of FBF; and
- Multitopology routing.

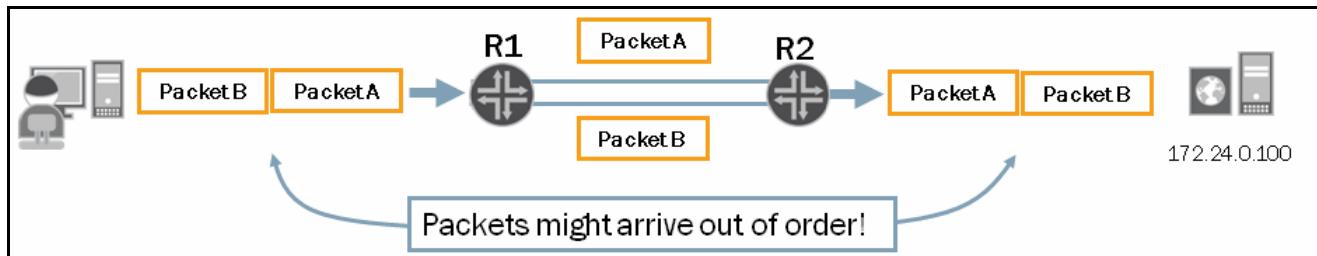
Equal-Cost Multipath Load Sharing



Equal-cost multipath load sharing (or load balancing) enables the ability to distribute traffic destined to the same destination prefix over paths with equal costs. Using equal-cost paths for traffic destined to the same destination prefix allows you to make use of redundant paths simultaneously rather than one of multiple paths.

Load-balancing methods include per-packet and per-flow. We cover these methods in detail on subsequent pages.

Per-Packet Load Balancing

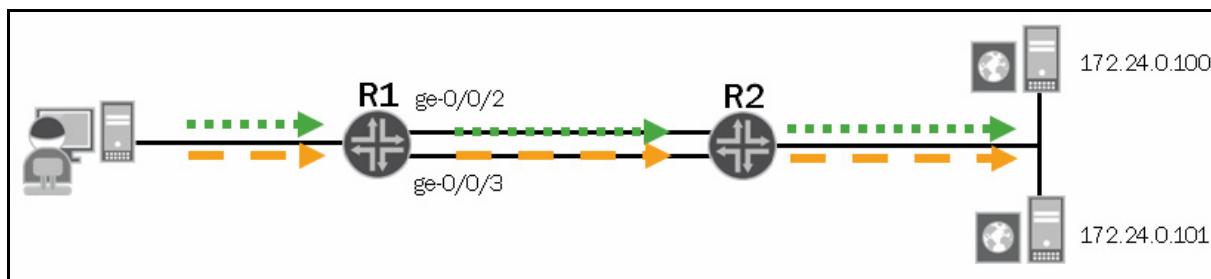


When a router performs per-packet load balancing, it forwards packets, in a round-robin fashion, out the egress interfaces, which connect to the equal-cost paths that lead to the destination. This load-balancing method appears to have some key benefits, such as equal distribution over the equal-cost paths; however, per-packet load balancing can actually decrease network performance.

In the example, R1 receives two packets (Packet A and Packet B). R1 sends Packet A out one interface and Packet B out the other interface. R2 receives both packets and forwards them on to the destination. There is no guarantee that R2 will receive and process Packet A before Packet B. Assuming that both packets belong to a common session (or flow), this out-of-sequence processing might require the destination device to reorder the packets or the source device to retransmit the packets, thus causing a reduction in performance.

Note that modern Junos devices do not implement per-packet load balancing. We highlight the current load-balancing implementation used in modern Junos devices in greater detail on a subsequent page.

Per-Flow Load Balancing

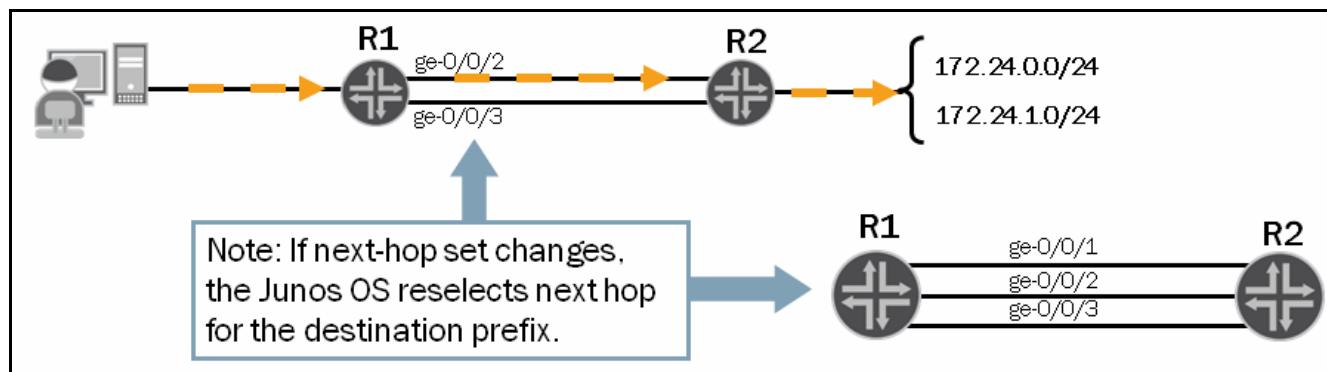


Unlike per-packet load balancing, per-flow load balancing maintains individual traffic flows between end stations, which results in a number of benefits. One benefit is that the packets generally arrive at the destination in the order they were sent. Thus, the end station does not have to reorder the packets, and the applications experience less delay. In addition, because similar user traffic uses the same path through the network, network-wide policies such as quality of service (QoS) are easier to implement. All protocols—not just TCP—can benefit from the establishment of flows through a network.

By default, the Junos operating system considers traffic that enters the same ingress interface and has the same source and destination addresses and the same protocol a single flow. You can change the elements of a packet that constitute a flow to include Layer 3 and Layer 4 port data. We cover the required configuration for this option on a subsequent page.

All traffic belonging to the same flow is forwarded by the system out the same egress interface. The example illustrates two distinct traffic flows that R1 forwards out ge-0/0/2 and ge-0/0/3, which are the egress interfaces associated with the equal-cost paths that lead to the destination network.

Default Behavior for the Junos OS



The default IGP load-balancing behavior for the Junos OS is to choose one of the available equal-cost paths over which traffic for the received destination prefixes will be sent. This process comes down to an issue of control. You know exactly where traffic is flowing on your network at any point in time. So when multiple, equal-cost paths exist in the routing table, the Junos OS selects one of the next hops from the available next-hop set and installs that next hop into the forwarding table.

If the information in the next-hop set changes, the Junos OS repeats the selection process. The next-hop set could change because of the addition or subtraction of a next hop. It could change because of the actual next-hop value changing for any of the possible paths.

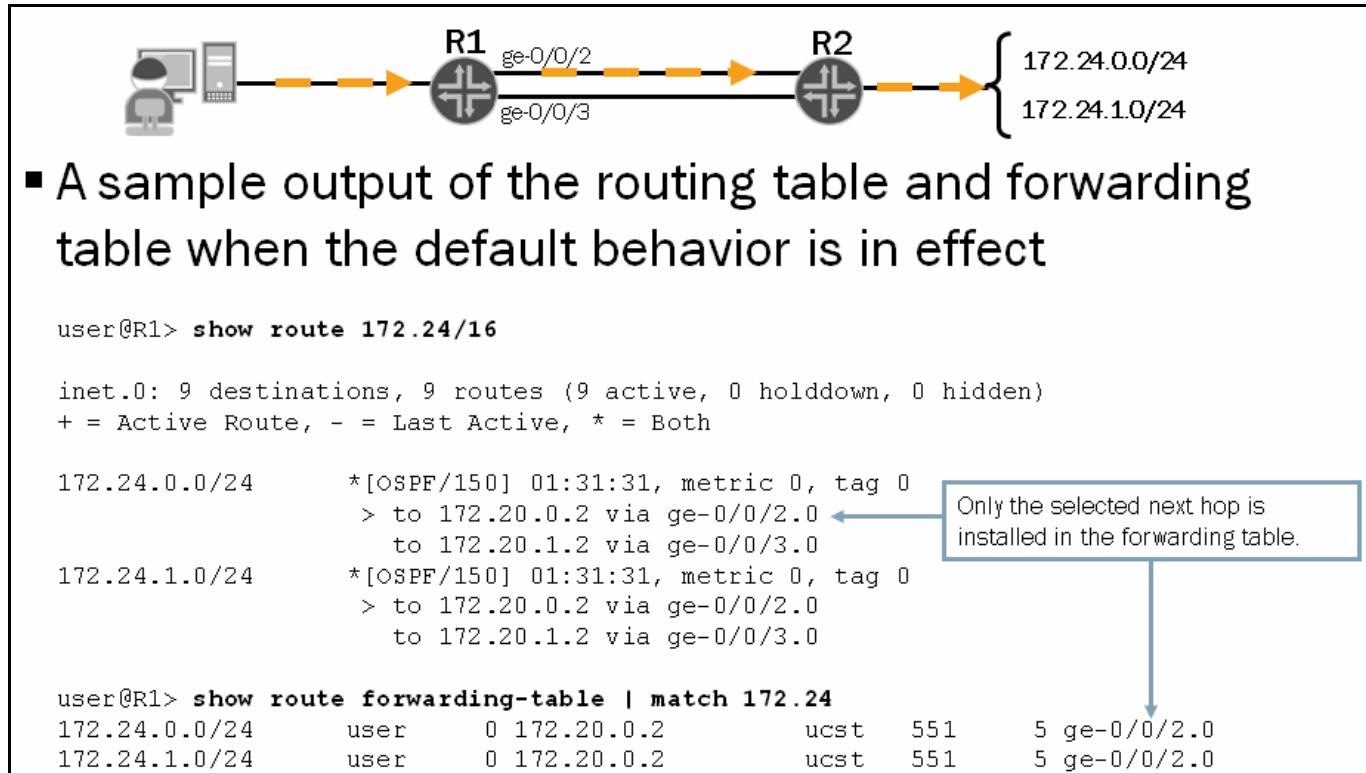
The default BGP load balancing is slightly different from the load balancing described previously, which applies to IGPs. The default BGP load balancing is known as per-prefix load balancing. Per-prefix load balancing occurs when routes are received from an internal BGP peer and some of those routes have identical BGP next-hop attributes (an IP address). When the router performs the lookup to find the BGP next-hop attribute value, the router might find multiple, equal-cost paths to that IP address.

In this case, the total number of received BGP routes that fits this criterion are spread across the available network paths. However, each individual BGP route still uses only one path through the network.

For example, suppose that IBGP advertises 100 routes from R2 to R1. R2 performs a next-hop self, so R1 now has multiple, equal-cost paths to R2's loopback address. In this case, the 100 routes will be randomly distributed across the equal-cost paths to R2. Note that a perfect distribution of the routes is not guaranteed because of the nature of the hashing algorithm used to select a next hop for a given prefix. In operation, the actual load-balancing distribution approximates an ideal distribution when a large number of prefixes is being factored.

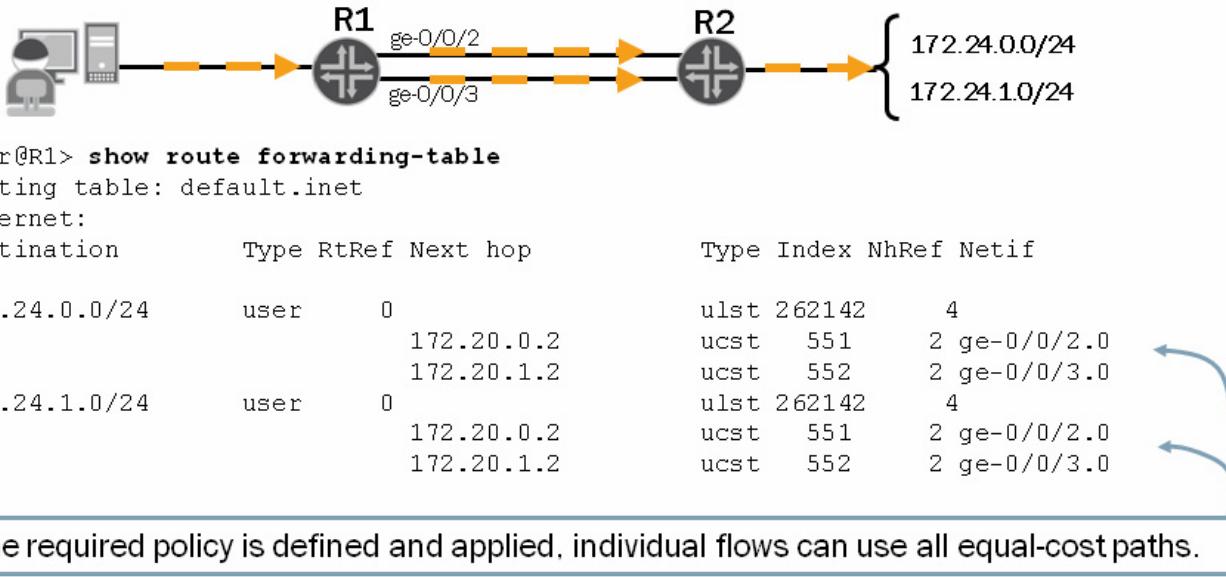
The Junos OS gives you policy controls to alter these default load-balancing behaviors. We cover the policy controls in the next section and BGP attributes in a subsequent chapter.

Viewing the Tables



The inset highlights the routing table and forwarding table when the default behavior is in effect. Notice that only the selected next hop is installed in the forwarding table for the referenced destination prefixes. On a subsequent page, we show a sample capture once the default load-balancing behavior has been altered to contrast with this output.

Changing the Default Behavior



You can alter the default load-balancing behavior using routing policy. Once a load-balancing policy is properly defined and applied, you see all equal-cost paths installed in the forwarding table as shown in the graphic. You can contrast this sample output with the output shown in the previous section where a load-balancing policy is not in place. We examine load-balancing configuration examples on subsequent pages in this section.

Configuration Examples

```

[edit policy-options]
user@R1# show
policy-statement load-balance-all {
    then {
        load-balance per-packet;
    }
}

[edit policy-options]
user@R1# show
policy-statement load-balance-some {
    from {
        route-filter 172.24.0.0/24 exact;
        route-filter 172.24.1.0/24 exact;
    }
    then {
        load-balance per-packet;
    }
}

```

All routes match a load-balancing policy without a `from` statement.

Only the specified routes match a load-balancing policy with a `from` statement.

Note: The `per-packet` policy option does not necessarily mean the device will perform per-packet load balancing. See next slide for details.

To load-balance traffic for all routes in the routing table, create a policy that matches all routes or select routes similar to the examples shown in the inset. The top example matches all destination prefixes because no `from` statement is configured. The second example matches only the listed destination prefixes using the `from` statement.

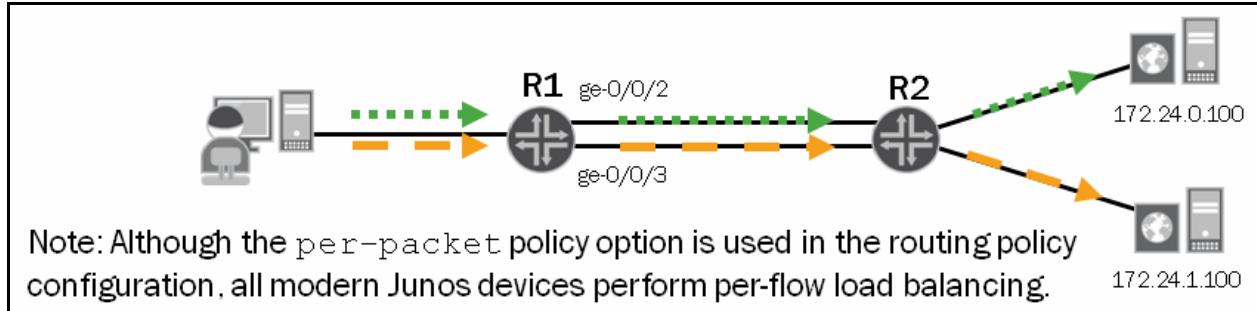
Because we are working with routing policy, we can use any of the policy match criteria to decide which routes are load-balanced and which routes are not load-balanced.

The action for both policies is `load-balance per-packet`. This action does not necessarily mean the software will perform a per-packet load-balancing operation. The actual load-balancing operation is platform dependent and is discussed in greater detail on the next page.

To perform load balancing, you must apply this policy as an export policy under the [edit routing-options] configuration hierarchy to the forwarding table. Once committed, this configuration starts to load-balance traffic for all routes having multiple, equal-cost paths through the network. We show a policy application example on a subsequent page.

Keep in mind that a load-balancing policy affects only the local router's forwarding behavior. To have all routers in the network load-balance traffic, you must create and apply a similar policy on each router.

Behavior Is Platform Dependent



The load-balancing policy created in the Junos OS always contains the same configuration syntax of `load-balance per-packet`. However, the actual load-balancing behavior is platform dependent.

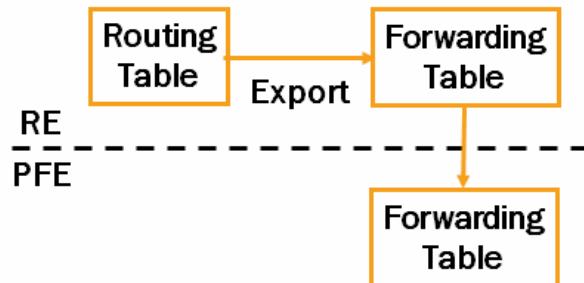
For platforms with the Internet Processor II functionality, packets are forwarded based on traffic flows. Depending on the actual platform and hardware, you can install up to 64 equal-cost paths in to the forwarding table. Check your product-specific documentation for support details. With the per-flow load-balancing method, all packets headed for a destination route that contain the same flow characteristics are all forwarded out the same outbound interface. This process avoids issues of packet arrival sequence and the need to resequence packets, which can slow TCP operation considerably.

For older routers still using the Internet Processor I ASIC, the balancing behavior is truly per-packet. The traffic is spread across the active equal-cost paths randomly, with the forwarding table performing the load distribution. Each packet that matches a destination route is forwarded across a different outbound interface in a round-robin fashion. The Internet Processor I ASIC can support up to eight different equal-cost paths per destination route.

Applying a Load-Balancing Policy

```
[edit routing-options]
user@R1# show
forwarding-table {
    export <policy-name>;
}
```

Active routes with equal cost paths are exported from the routing table to the forwarding table.



Once the load-balancing policy is defined, you must apply the policy as an export policy under the [edit routing-options] configuration hierarchy to the forwarding table. Once committed, this configuration starts to load-balance traffic for the routes referenced within the routing policy that have multiple, equal-cost paths through the network.

Note that defining and applying a load-balancing policy affects only the forwarding table. The routing table remains as it was before you defined and applied the load-balancing policy.

Determining Traffic Flows

By default, the Junos OS considers traffic that enters the same ingress interface and has the same source and destination addresses and the same protocol a single flow.

You can change the elements of a packet that constitute a flow to also include Layer 4 port data. Once the software determines the criteria for a given flow, the system forwards all traffic that matches the flow out the same egress interface.

Incoming Interface Index	Destination Address
Source Address	Protocol

Including Port Data when Determining Traffic Flows

```
[edit forwarding-options]
user@R1# show
hash-key {
    family inet {
        Layer-3;
        Layer-4;
    }
}
```

The Junos OS will now include Layer 4 port data when determining traffic flows.

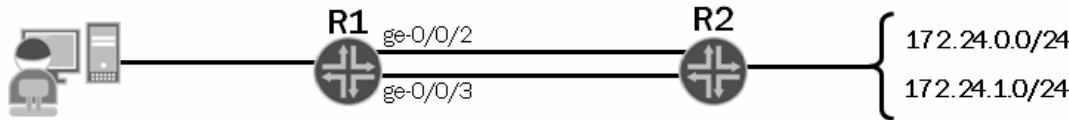
If you want the flow calculation to include IP Layer 4 source and destination port information along with the default Layer 3 parameters, you must configure the `hash-key` accordingly under the `[edit forwarding-options]` section. An example of the required configuration is shown in the inset. Note that you must include both the Layer 3 and Layer 4 options and not just the Layer 4 option. If you omit the Layer 3 option, the Junos OS will not use all of the listed Layer 3 and Layer 4 parameters when determining the flow calculation.

You can also optimize MPLS and VPLS flows by modifying the `hash-key` options under the `[edit forwarding-options]` section using the appropriate `set family mpls` and `set family multiservice` statements respectively. Detailed coverage of these options are outside the scope of this study guide.

By default, IP version 6 (IPv6) packets are automatically load-balanced based on Layer 3 and Layer 4 information and traffic class.

Case Study: Topology and Objectives

- Ensure that both equal-cost next hops are installed in R1's forwarding table for the listed prefixes
- Ensure that R1 includes Layer 3 and Layer 4 port information when determining traffic flows



The graphic displays the topology and objectives for our case study.

Case Study: Configuring Load Balancing

```
[edit policy-options]
user@R1# show
policy-statement load-balance {
    from {
        route-filter 172.24.0.0/24 exact;
        route-filter 172.24.1.0/24 exact;
    }
    then {
        load-balance per-packet;
    }
}

[edit routing-options]
user@R1# show
forwarding-table {
    export load-balance;
}

[edit forwarding-options]
user@R1# show
hash-key {
    family inet {
        layer-3;
        layer-4;
    }
}
```

Definition of load-balancing policy

Application of load-balancing policy

Inclusion of Layer 3 and Layer 4 port data when determining traffic flows

The inset displays the required configuration to accomplish the stated objectives on the previous page. Remember that a commit is required once the highlighted configuration tasks are complete.

Case Study: Monitoring Load Balancing

```
user@R1> show route forwarding-table
Routing table: default.inet
Internet:
Destination          Type RtRef Next hop
...
172.24.0.0/24        user     0
                      172.20.0.2
                      172.20.1.2
172.24.1.0/24        user     0
                      172.20.0.2
                      172.20.1.2
...
...
```

ulst indicates a list of unicast next hops. A packet sent to this next hop goes to any next hop in the list.

Type	Index	NhRef	Netif
ulst	262142	3	
ucst	551	4	ge-0/0/2.0
ucst	552	2	ge-0/0/3.0
ulst	262142	3	
ucst	551	4	ge-0/0/2.0
ucst	552	2	ge-0/0/3.0

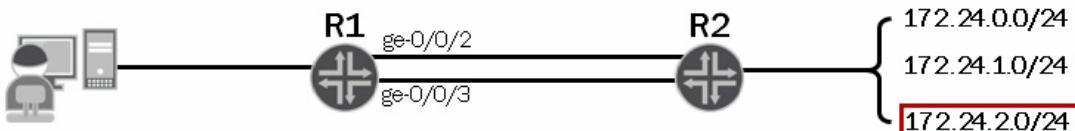


The graphic shows the output for the **show route forwarding-table** command, which is the primary verification step to ensure that the load-balancing policy took effect. In the sample output, we see that both forwarding next hops were installed in the forwarding table for both destination prefixes.

Test Your Knowledge

- R2 advertises the 172.24.2.0/24 prefix to R1 through the ge-0/0/2 and ge-0/0/3 interfaces

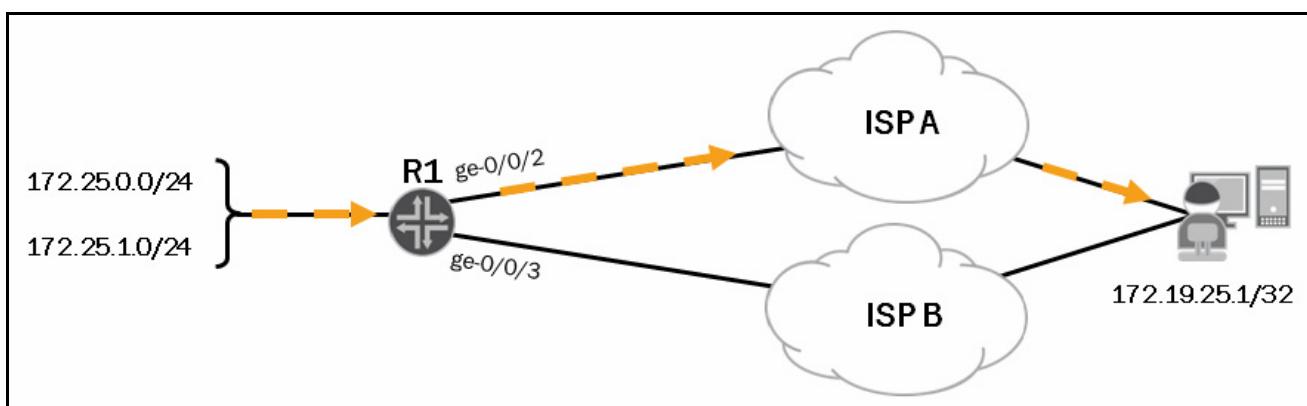
- What load-balancing method will R1 use when forwarding traffic to the newly learned destination prefix? (Assume the recently defined configuration is still in place.)



Assuming we still have the previously defined configuration in place, R1 will select one of the two equal-cost paths. Remember that our load-balancing policy matched only on the 172.24.0.0/24 and 172.24.1.0/24 prefixes.

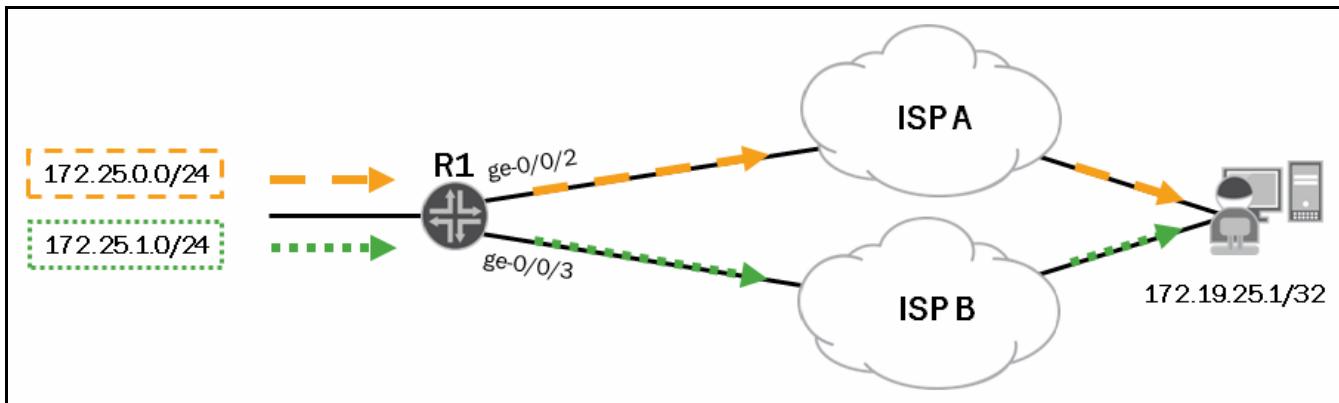
```
[edit policy-options]
user@R1# show
policy-statement load-balance {
    from {
        route-filter 172.24.0.0/24 exact;
        route-filter 172.24.1.0/24 exact;
    }
    then {
        load-balance per-packet;
    }
}
```

If you want to match on all destination prefixes with equal-cost paths, you must remove the `from` statement from the load-balancing policy.

Traditional Routing

In a typical routing scenario, a router looks only at the incoming packet's destination address when determining the forwarding next hop. Although this approach works, it does not provide much flexibility. We discuss filter-based forwarding on the next page, which provides you with flexibility for determining distinct forwarding paths for traffic based on a wide variety of options.

Filter-Based Forwarding



Filter-based forwarding allows a router to make forwarding decisions based on criteria in addition to the destination prefix. A match filter is created and applied to the interface on which traffic is received. The Junos OS uses the filter to classify packets and determine their forwarding path within the router.

Filter-based forwarding is supported for Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6).

As shown in the graphic, you can use filter-based forwarding for service-provider selection when you have Internet connectivity provided by different ISPs. We illustrate the configuration and verification steps for filter-based forwarding in the next section.

Configuring Filter-Based Forwarding

1. Creating a match filter	Defined under [edit firewall] hierarchy. Matches traffic with respective routing instance.
2. Creating routing instances	Defined under [edit routing-instances] hierarchy. Creates unique routing tables to forward traffic toward destination along associated path.
3. Creating a RIB group	Defined under [edit routing-options] hierarchy. Shares interface routes between instances so directly connected next hops can be resolved.

The table illustrates the three primary configuration tasks when implementing filter-based forwarding. The subsequent pages provide configuration examples for these stated configuration tasks.

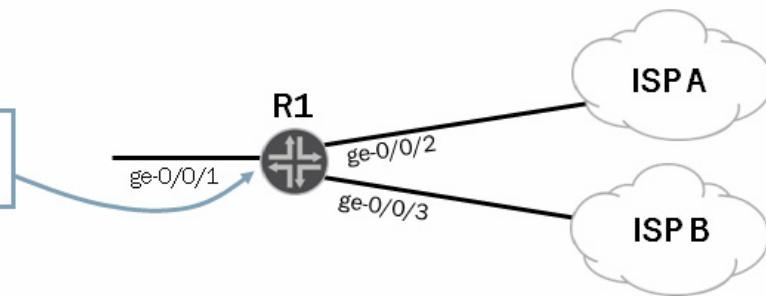
If you are planning to implement filter-based forwarding, you should know that source-class usage filter matching and unicast reverse-path forwarding checks are not supported on interfaces configured for filter-based forwarding. We do not cover source-class usage or unicast reverse-path forwarding (RPF) in this study guide. For more details on these topics, refer to <http://www.juniper.net>.

Step 1: Create and Apply a Match Filter

```
[edit]
user@R1# show firewall
family inet {
    filter filter-name {
        term term-name {
            from {
                match-conditions
            }
            then {
                routing-instance instance-name
            }
        }
    }
}
```

The defined filter is applied as an input filter to ingress interface.

The filter matches on desired criteria with an action of `routing-instance instance-name`.



The inset shows a sample match filter configuration. Match filters are standard firewall filters used to classify traffic based on a desired match criteria, such as source IP address. You use an action of `routing-instance` along with the name of the desired routing instance to which you want the traffic forwarded. We define the routing instance in the next step, and we provide a sample configuration on the next page.

Although not explicitly shown in the inset, you must apply the match filter as an input filter on the ingress interface where the traffic will be received. We provide a full configuration example, which includes the definition and application of a match filter, in the case study later in this section.

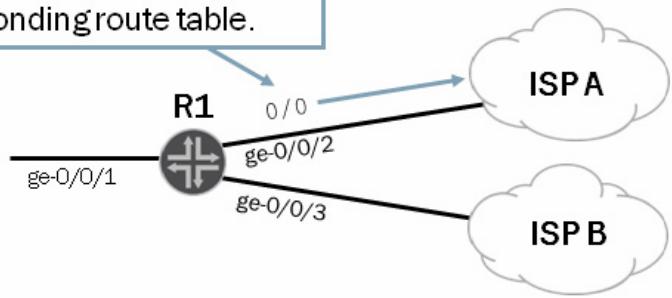
Remember that the default action for traffic not specifically permitted through a firewall filter is discard. Because of this default action, you should ensure that all traffic is accounted for in the defined firewall filter. We highlight this point on a subsequent page.

Step 2: Create the Routing Instances

```
[edit]
user@R1# show routing-instances
instance-name {
    instance-type forwarding;
    routing-options {
        static {
            route 0.0.0.0/0 next-hop ip-address;
        }
    }
}
```

The instance type used for filter-based forwarding is always **forwarding**.

The instance includes routing information (typically a default static route) that is stored in the corresponding route table.



Once a packet meets the criteria defined within the match filter, that packet is evaluated against the routing table associated with the routing instance specified in the match filter's action statement. The software then forwards the packet to the next hop that corresponds to the destination address entry in the routing table associated with the routing instance. You can include a default static route, as shown in the example, or you can include less specific static routes. These static routes typically use a forwarding next hop IP address, but can also direct traffic to a different routing instance using the next-table option, as shown in the following capture:

```
routing-instances {
    instance-name {
        instance-type forwarding;
        routing-options {
            static {
                route 0.0.0.0/0 next-table inet.0;
            }
        }
    }
}
```

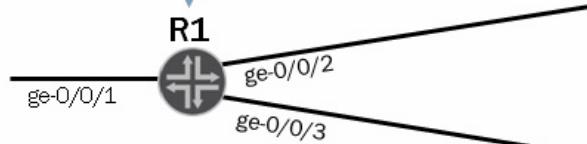
As illustrated in the inset, you create routing instances that specify the routing tables and the destination to which a packet is forwarded at the [edit routing-instances] hierarchy level. The instance type used for filter-based forwarding is **forwarding**. The number of routing instances you define depends on the number of next-hop forwarding paths you will use.

Step 3: Create a RIB Group

```
[edit]
user@R1# show routing-options
interface-routes {
    rib-group inet group-name;
}
rib-groups {
    group-name {
        import-rib [ inet.0 instance-name.inet.0 ];
    }
}
```

You must reference the corresponding table for each of the participating instances.

The sample configuration shares interface routes between referenced instances. This requirement can resolve directly connected next hops.

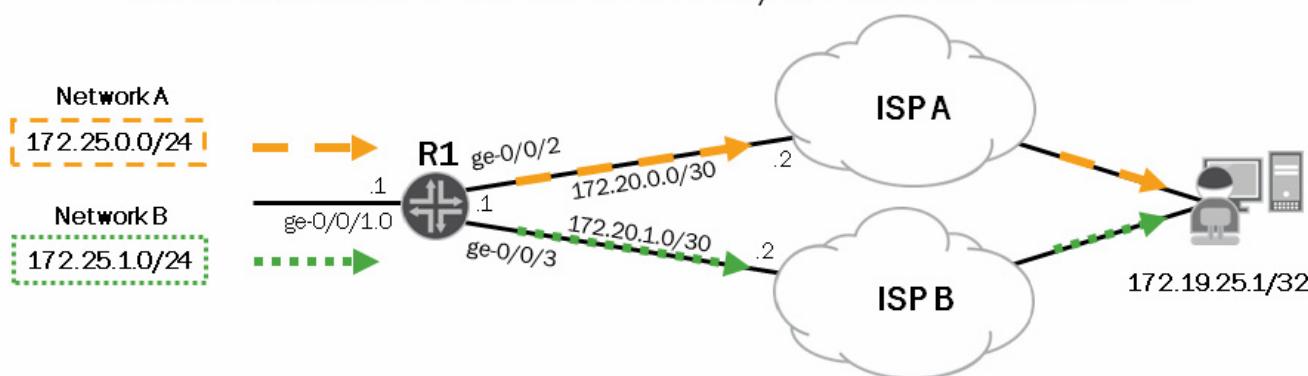


In this step, you create a routing table (or RIB) group that adds interface routes to the forwarding routing instances used in FBF, as well as to the default routing instance `inet.0`. This part of the configuration resolves the routes installed in the routing instances to directly connected next hops on that interface. You create routing table groups at the `[edit routing-options]` hierarchy level.

The configuration example in the inset installs interface routes into the default routing instance `inet.0`, as well as the `instance-name.inet.0` routing tables. You must specify `inet.0` as one of the routing instances that the interface routes are imported into. If `inet.0` is not specified, interface routes are not imported into the default routing instance.

Case Study: Topology and Objective

- Use FBF to balance Internet-bound traffic based on source address on the R1 device
 - Traffic sourced from 172.25.0.0/24 should use ISP A
 - Traffic sourced from 172.25.1.0/24 should use ISP B



The graphic provides us with the topology and objective for the case study.

Case Study: Configure the Match Filter

```
[edit firewall]
user@R1# show family inet filter my-match-filter
term match-subnet-A {
    from {
        source-address {
            172.25.0.0/24;
        }
    }
    then {
        routing-instance ISP-A; ## 'ISP-A' is not defined
    }
}
term match-subnet-B {
    from {
        source-address {
            172.25.1.0/24;
        }
    }
    then {
        routing-instance ISP-B; ## 'ISP-B' is not defined
    }
}
```

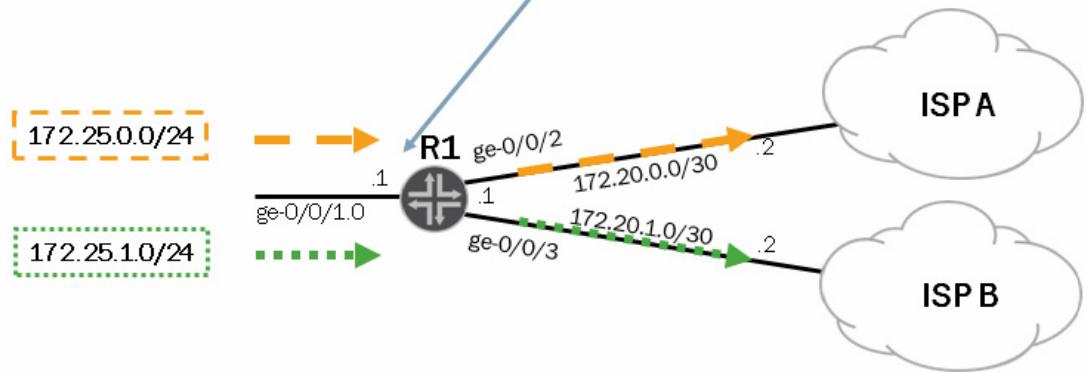
Routing instances will be defined in a subsequent step.

The inset displays the sample match filter configuration.

Case Study: Apply the Match Filter

```
[edit interfaces ge-0/0/1]
user@R1# show
unit 0 {
    family inet {
        filter {
            input my-match-filter;
        }
        address 172.25.0.1/24;
        address 172.25.1.1/24;
    }
}
```

The match filter is applied as an input filter on the ge-0/0/1.0 interface.



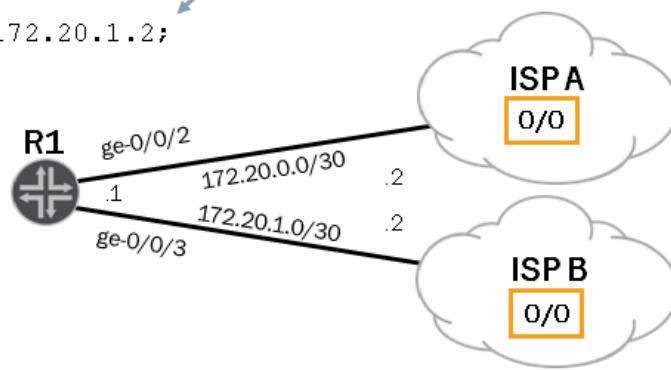
The graphic shows the application of the match filter defined on the previous page.

Case Study: Configure the Routing Instances

```
[edit routing-instances]
user@R1# show
ISP-A {
    instance-type forwarding; ← Instances use the forwarding instance type.
    routing-options {
        static {
            route 0.0.0.0/0 next-hop 172.20.0.2;
        }
    }
}
ISP-B {
    instance-type forwarding;
    routing-options {
        static {
            route 0.0.0.0/0 next-hop 172.20.1.2;
        }
    }
}
```

Instances use the forwarding instance type.

Instances include required routing information.

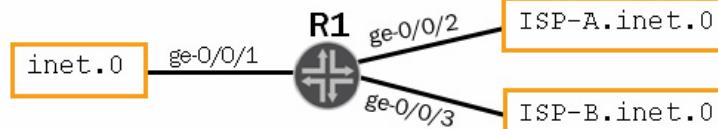


The graphic shows the sample configuration of the routing instances that will be used to establish distinct forwarding paths within R1.

Case Study: Configure the RIB Group

```
[edit routing-options]
user@R1# show
interface-routes {
    rib-group inet my-rib-group;
}
rib-groups {
    my-rib-group {
        import-rib [ inet.0 ISP-A.inet.0 ISP-B.inet.0 ];
    }
}
```

Reference the corresponding table for each of the participating instances to share interface routes.



The graphic shows a sample RIB group, which is used to share interface routes. You must share interface routes for next-hop resolution when forwarding traffic between routing instances.

Case Study: Verify the Results—Part 1

```
user@R1> show route table ISP-A.inet.0

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

0.0.0.0/0      *[Static/5] 00:13:49
                > to 172.20.0.2 via ge-0/0/2.0
172.20.0.0/30 *#[Direct/0] 00:13:49
                > via ge-0/0/2.0
172.20.0.1/32 *#[Local/0] 00:00:15
                Local via ge-0/0/2.0
172.20.1.0/30 *#[Direct/0] 00:13:49
                > via ge-0/0/3.0
172.20.1.1/32 *#[Local/0] 00:00:17
                Local via ge-0/0/3.0
172.25.0.0/24  *#[Direct/0] 00:13:49
                > via ge-0/0/1.0
172.25.0.1/32 *#[Local/0] 00:00:17
                Local via ge-0/0/1.0
172.25.1.0/24  *#[Direct/0] 00:13:49
                > via ge-0/0/1.0
172.25.1.1/32 *#[Local/0] 00:00:17
                Local via ge-0/0/1.0
```

Note: All interface routes exist in the route table associated with the ISP-A routing instance. The ISP-B.inet.0 table has the same routing entries.

Once the configuration shown in the previous section is committed, you can verify the entries in the relevant routing tables using the **show route table table-name** command. The inset shows the contents of the ISP-A.inet.0 routing table, which

includes all interface routes for `inet.0`, `ISP-A.inet.0`, and `ISP-B.inet.0`, along with the default static route defined for the routing instance.

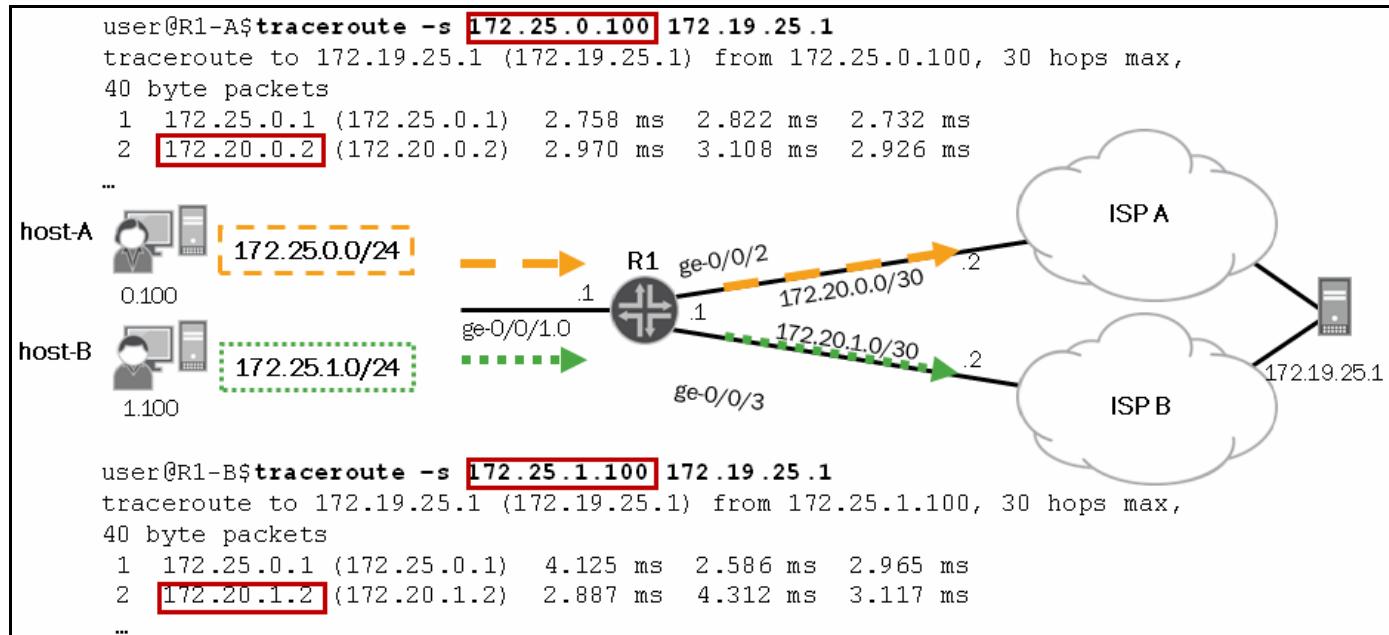
The contents of the `ISP-B.inet.0` routing table are identical and are shown in the following output:

```
user@R1> show route table ISP-B.inet.0

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

0.0.0.0/0          *[Static/5] 03:32:41
                  > to 172.20.1.2 via ge-0/0/3.0
172.20.0.0/30     *[Direct/0] 03:32:41
                  > via ge-0/0/2.0
172.20.0.1/32     *[Local/0] 03:19:07
                  Local via ge-0/0/2.0
172.20.1.0/30     *[Direct/0] 03:32:41
                  > via ge-0/0/3.0
172.20.1.1/32     *[Local/0] 03:19:09
                  Local via ge-0/0/3.0
172.25.0.0/24     *[Direct/0] 03:32:41
                  > via ge-0/0/1.0
172.25.0.1/32     *[Local/0] 03:19:09
                  Local via ge-0/0/1.0
172.25.1.0/24     *[Direct/0] 03:32:41
                  > via ge-0/0/1.0
172.25.1.1/32     *[Local/0] 03:19:09
                  Local via ge-0/0/1.0
```

Case Study: Verify the Results—Part 2

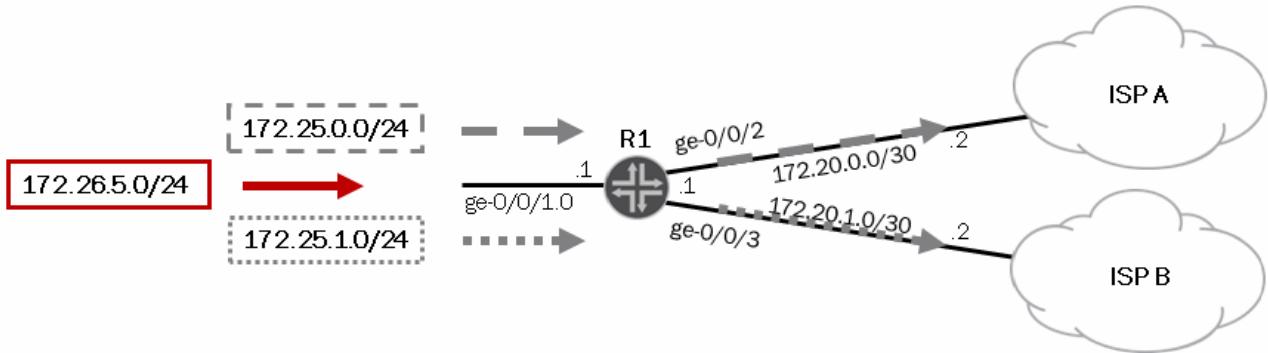


As indicated in the graphic, you can also perform a traceroute operation from a host on the subnets for which FBF is being performed to verify that the traffic takes the correct forwarding path.

As stated earlier, our objective was to ensure that traffic sourced from the `172.25.0.0/24` subnet used ISP A and traffic sourced from the `172.25.1.0/24` subnet used ISP B. The captures shown in the graphic, which display the traceroute results from hosts on the two previously mentioned subnets, indicate that our FBF implementation works as designed.

Test Your Knowledge

- Using the previous example and configuration, what would R1 do with traffic sourced from 172.26.5.0/24?



R1 will drop all traffic entering the ge-0/0/1.0 interface not sourced from the 172.25.0.0/24 and 172.25.1.0/24 subnets. You can include an additional term to accept all other traffic.

```
term else-accept {
    then {
        accept;
    }
}
```

This graphic is designed to make you think through the affects of the configuration implemented in the preceding case study.

Because the configuration used in the preceding case study only accepts traffic sourced from the 172.25.0.0/24 and 172.25.1.0/24 subnets, you must either modify the existing terms or include a third term to accept traffic with a different source address. The following example illustrates the use of a third term named `else-accept` which permits all other traffic:

```
[edit firewall]
user@R1# show family inet filter my-match-filter
term match-subnet-A {
    from {
        source-address {
            172.25.0.0/24;
        }
    }
    then {
        routing-instance ISP-A;
    }
}
term match-subnet-B {
    from {
        source-address {
            172.25.1.0/24;
        }
    }
    then {
        routing-instance ISP-B;
    }
}
term else-accept {
    then {
```

```

    accept;
}
}

```

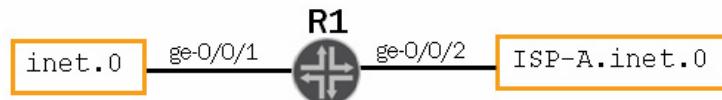
The `else-accept` term accepts all traffic that does not match the preceding terms. The system forwards the traffic that matches the `else-accept` term out the appropriate egress interface using the normal route lookup process based on the destination prefix. We recommend you use a similar approach when implementing filter-based forwarding so traffic is not dropped unintentionally.

Using the `instance-import` Option

In earlier releases of the Junos OS, interinstance route sharing often required configuration of routing table groups by means of the `rib-group` statement. Although these configurations performed well, the routing table group technique has several limitations:

- Lack of intuitiveness—A routing table group is an unfamiliar configuration construct for many users.
- Complex configuration requirements—Routing table groups specify a primary import routing table that must match the routing table of the VRF instance on which they are applied. Thus, a different routing table group is defined for each of the instances that participate in interinstance route export.
- Redundancy—The information imported and exported by the routing table groups is already present in the router or can be deduced from most configurations (for example, overlapping VPNs).
- Per-protocol configuration—Routing table groups must be applied to every protocol containing routes designated for export.

The need exists to share routes between the `inet.0` and `ISP-A.inet.0` tables. Normally, `rib-groups` are used for this purpose.



Configuring the `instance-import` Option

```
user@R1> show route table ISP-A.inet.0
```

```
user@R1>
```

No routes currently exist in the `ISP-A.inet.0` table before committing the following configuration.

```
[edit]
user@R1# show policy-options
policy-statement ISP-A-import {
    from instance master;
    then accept;
}
```

Could be any routing-instance name. In this case, the `master` keyword is used to refer to the master routing instance to import routes from `inet.0` into `ISP-A.inet.0`.

```
[edit]
user@R1# show routing-instances
ISP-A {
    instance-type forwarding;
    routing-options {
        static {
            route 0.0.0.0/0 next-hop 172.20.0.2;
        }
        instance-import ISP-A-import;
    }
}
```

Call the `ISP-A-import` policy created above.

In Junos Release 5.4 and later, you can use the `instance-import` policy keywords to perform nonforwarding, interinstance route sharing. The keywords are assigned at the `[edit routing-instances instance-name routing-options]` hierarchy level.

In the example, no routes currently exist in the ISP-A.inet.0 route table. After committing the configuration shown in the inset, we verify on the next page that the routes from the inet.0 table have been imported into the ISP-A.inet.0 table.

Verify the Results

```
user@R1> show route table ISP-A.inet.0

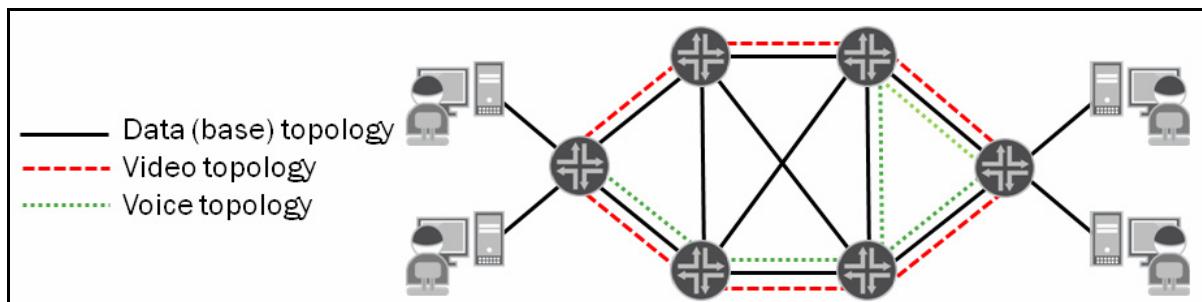
ISP-A.inet.0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 00:13:49
                  > to 172.20.0.2 via ge-0/0/2.0
172.20.0.0/30     *[Direct/0] 00:13:49
                  > via ge-0/0/2.0
172.20.0.1/32     *[Local/0] 00:00:15
                  Local via ge-0/0/2.0
172.20.1.0/30     *[Direct/0] 00:13:49
                  > via ge-0/0/3.0
172.20.1.1/32     *[Local/0] 00:00:17
                  Local via ge-0/0/3.0
172.25.0.0/24     *[Direct/0] 00:13:49
                  > via ge-0/0/1.0
172.25.0.1/32     *[Local/0] 00:00:17
                  Local via ge-0/0/1.0
172.25.1.0/24     *[Direct/0] 00:13:49
                  > via ge-0/0/1.0
172.25.1.1/32     *[Local/0] 00:00:17
                  Local via ge-0/0/1.0
```

All inet.0 routes have been imported into the ISP-A.inet.0 table.

Using the same example as in the previous FBF case study, you can see that the results are the same. All inet.0 routes have been imported.

Multitopology Routing Basics



Although detailed coverage of multitopology routing is outside the scope of this study guide, we will go over some basics. Multitopology routing enables you to configure class-based forwarding for different types of traffic, such as voice, video, and data. Each type of traffic is defined by a topology that is used to create a new routing table for that topology. Multitopology routing provides the ability to generate forwarding tables based on the resolved entries in the routing tables for the custom topologies you create. In this way, packets of different classes can be routed independently from one another.

Configuring Topologies

```
[edit]
user@R1# show routing-options
topologies {
    family inet {
        topology my-video-topology
    }
}
```

Use family `inet6` to specify IPv6 traffic.

Simple string value. Typically, you would use a name that describes the type of traffic used within the topology.

For multitopology routing to run on the router, you must configure one or more topologies. For each topology, you specify a string value, such as video, that defines the type of traffic, as well as an interface family, such as IPv4 or IPv6. Each topology that you configure creates a new routing table and populates it with direct routes from the topology.

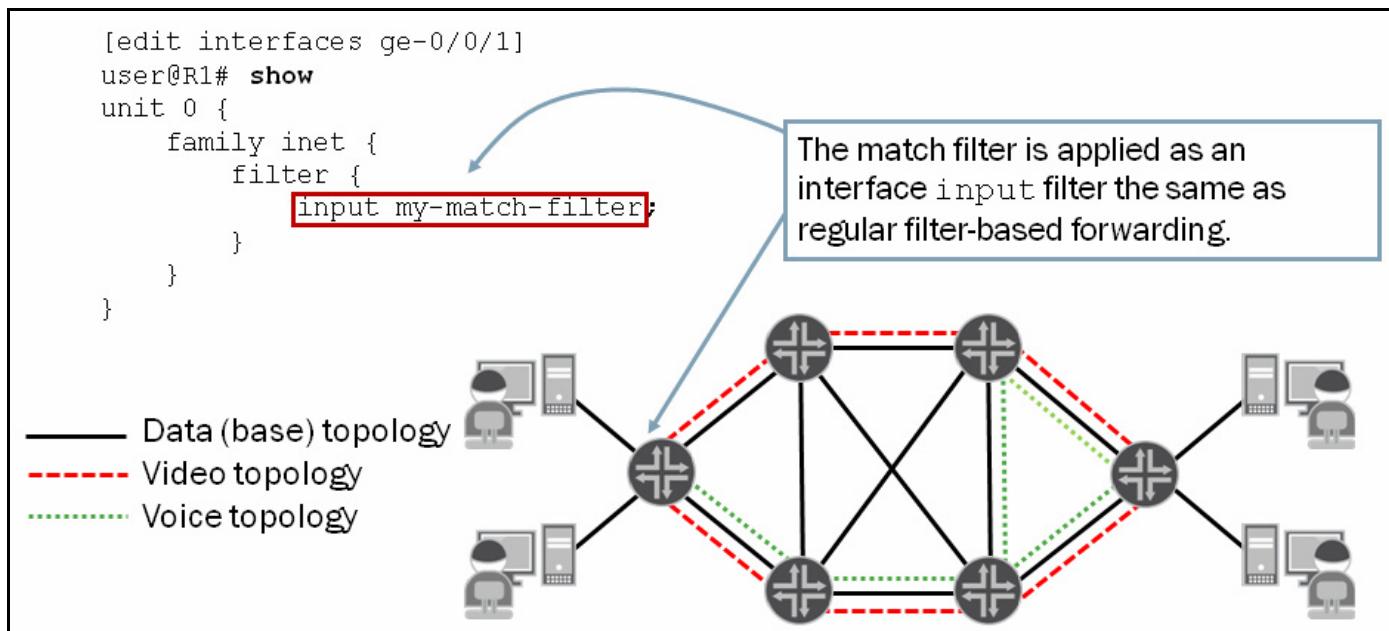
Configuring Filter-Based Forwarding for Multitopology Routing

```
[edit]
user@R1# show firewall
family inet {
    filter my-match-filter {
        term term-name {
            from {
                forwarding-class expedited-forwarding
            }
            then {
                topology my-video-topology
            }
        }
    }
}
```

Can be any of the four main classes:
assured-forwarding, best-effort, expedited-forwarding,
or network-control

Normally, each routing instance supports only one default topology to which all forwarding classes are forwarded. With Multitopology routing, you can configure a firewall filter on the ingress interface to map a specific forwarding class, such as expedited forwarding, with a specific topology. The traffic that matches the specified forwarding class is then added to the routing table for that topology.

Case Study: Apply the Match Filter



The graphic shows the application of the match filter defined on the previous page.

Review Questions

1. Describe the default load-balancing behavior when equal-cost paths for a destination exist.
2. Describe the operation of the load-balance per-packet policy action.
3. Which instance type does FBF use?
4. What is the purpose of the RIB group when configuring FBF?

Answers

1.

For IGP, the Junos OS chooses one of the available equal-cost paths for destination prefixes. For BGP, the Junos OS alternates its selection among the equal-cost paths (known as per-prefix load balancing).

2.

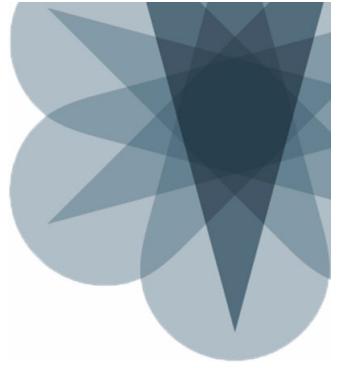
The actual behavior associated with the per-packet action depends on the platform. Most modern Junos OS-based devices perform per-flow load balancing, whereas platforms that use the Internet Processor ASIC (no longer sold) perform a per-packet load-balancing operation.

3.

FBF uses the **forwarding** instance type.

4.

The RIB group is used to share interface routes, which is required for next-hop resolution when FBF is used.



JNCIS-SP Study Guide—Part 1

Chapter 3: Open Shortest Path First

This Chapter Discusses:

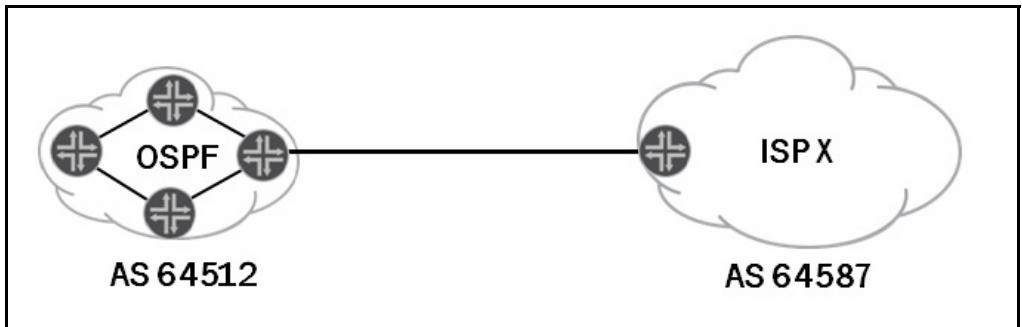
- OSPF protocol operations;
- Purpose of the designated router (DR);
- OSPF area types; and
- Configuration, monitoring, and troubleshooting of OSPF.

OSPF Overview

OSPF is a link-state routing protocol designed for use within an autonomous system (AS). As a link-state interior gateway protocol (IGP), OSPF allows for faster reconvergence, supports larger internetworks, and is less susceptible to bad routing information than distance-vector protocols.

Link State Advertisements

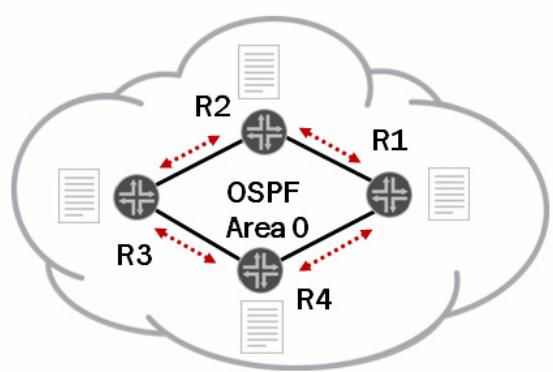
Once an OSPF router becomes neighbors with other OSPF routers, it can begin to share information about its attached networks. OSPF routers use link-state advertisements (LSAs) to reliably flood information about their network links and the state of those links to their neighboring OSPF routers.



As link-state information is shared between OSPF routers, each OSPF router creates and maintains a link-state (or topological) database. The OSPF routers use the information provided within the LSAs as input for the shortest-path first (SPF) algorithm to calculate the best path for each destination prefix.

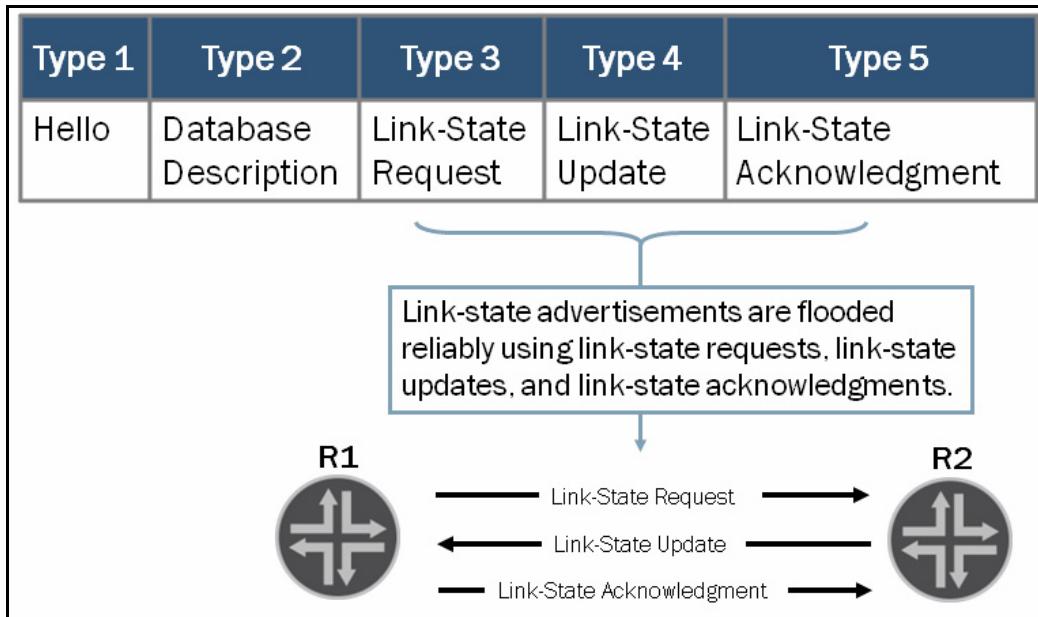
The Link State Database

In addition to discovering neighbors and flooding LSAs, a third major task of the OSPF protocol is to create and maintain the link-state database (LSDB). The link-state, or topological, database stores the LSAs as a series of records. The contents stored within the LSDB include details such as the advertising router's ID, its attached networks and neighboring routers, and the cost associated with those networks or neighbors. According to the OSPF RFC, each router in an OSPF area must have an identical LSDB to ensure accurate routing knowledge. We discuss OSPF areas later in this chapter.



The information recorded in the database is used as input data to calculate the shortest paths (that is, least-cost paths) for all destination prefixes within the network. OSPF uses the SPF algorithm or Dijkstra algorithm to calculate, all at once, the shortest paths to all destinations. It performs this calculation by creating a tree of shortest paths incrementally and picking the best candidate from that tree. The results of this calculation are then handed to the router's routing table for the actual forwarding of data packets.

OSPF Packet Types



The graphic lists the OSPF packet types. We discuss each packet type in more detail on subsequent pages in this section.

Hello Packet

Type 1	Type 2	Type 3	Type 4	Type 5
Hello	Database Description	Link-State Request	Link-State Update	Link-State Acknowledgment

- Multicast hello packets are used to establish and maintain OSPF neighbor relationships
 - Sent to 224.0.0.5—all OSPF routers address
 - Consist of the OSPF header plus the following fields:

Network mask*	Hello interval*	Dead interval*	Options*
Router priority	Designated router	Backup designated router	Neighbor

* Fields that must match to form an adjacency over a broadcast medium: a matching network mask is not required for point-to-point links

All OSPF routers send hello packets on all their links on a regular cycle, which is 10 seconds by default. The hello packet is multicast to the *all OSPF routers* multicast address of 224.0.0.5. In addition to the OSPF header, OSPF includes additional information specific to that link.

Some of the link-specific information must match between neighbors before they can form an adjacency. In the fields table, an asterisk (*) marks the fields that must match to form an adjacency. The following list contains the details of these fields:

- *Network mask*: This field is evaluated only on broadcast media links (that is, Ethernet). All routers on the segment must agree on the subnet mask of the link.
- *Hello interval*: The two routers must agree on how often to send hello packets, which this field determines.
- *Dead interval*: Also referred to as the keepalive, this timer states how long to wait before removing the adjacency for a neighbor. The two routers must agree on this timer.
- *Options*: This 8-bit field represents such things as the ability to be a stub area. These options are critical for correct OSPF operation, so they, too, must match between neighbors.

Database Description Packet

Type 1	Type 2	Type 3	Type 4	Type 5
Hello	Database Description	Link-State Request	Link-State Update	Link-State Acknowledgment

OSPF uses database description (DD) packets only during the adjacency formation process between two OSPF routers. The DD packets serve two main purposes: determining who is in charge of the database synchronization, and actually transferring the LSA headers between the two systems.

For database synchronization, the two OSPF routers must decide who is in charge. The router with the highest router ID (RID) is selected to be in charge of the database synchronization process. The router in charge of the database synchronization process is known as the *master*; the other router is the *slave*. Simply put, it is the job of the master to set and maintain the sequence numbers used during the database transfer. After the transfer is complete, the knowledge of the master/slave relationship is forgotten.

OSPF also uses DD packets to transmit LSA headers between the systems. This transmission serves as the minimum synchronization between the routers.

The details of the DD packet fields are as follows:

- *OSPF header*: This header is 24 bytes.
- *Sequence number*: This field ensures that the full sequence of DD packets is received in the database synchronization process. The sequence number is set by the master to some unique value in the first DD packet, and the sequence is incremented in subsequent packets.
- *LSA header*: This header lists some or all of the headers of the LSAs in the originator's link-state database. The header contains enough information to uniquely identify the LSA and the particular instance of the LSA.

Link-State Request Packet

Type 1	Type 2	Type 3	Type 4	Type 5
Hello	Database Description	Link-State Request	Link-State Update	Link-State Acknowledgment

- Link-state request packets are sent by an OSPF router when that router detects its database is stale
 - Used to request precise version of database and consist of the OSPF header, link-state type, link-state ID, and advertising router



After receiving a number of DD packets, a router might find that its neighbor is advertising an LSA header that is not currently in its own database. In that situation, the receiving router sends out a link-state request packet to the sending router. In simplest terms, the link-state request packet is the request for information; it contains the LSA header for the missing link.

The following list contains the details of the link-state request packet fields:

- *OSPF header*: This header is 24 bytes.
- *Link-state type*: This field contains the LSA type number, which identifies the type of LSA (for example, a router LSA or network LSA).
- *Link-state ID*: This field is type-dependent on the LSA header.
- *Advertising router*: This field contains the router ID of the router that originated the LSA.

Link-State Update Packet

Type 1	Type 2	Type 3	Type 4	Type 5
Hello	Database Description	Link-State Request	Link-State Update	Link-State Acknowledgment

- Link-state update packets are the basic information block in OSPF and can carry multiple LSAs

- Transmitted using multicast to either the *all OSPF routers* address (224.0.0.5) or the *all DRs* address (224.0.0.6), and consist of the OSPF header, number of advertisements, and LSAs



A link-state update packet is the basic information block in OSPF. It can contain multiple LSAs. Link-state update packets are reliably transmitted using multicast to either the *all OSPF routers* address (224.0.0.5) or the *all DRs* address (224.0.0.6), depending on the link type.

OSPF sends link-state update packets in two different ways: in response to a link-state request packet during the adjacency database synchronization, and after an adjacency is formed, if information about that link changes.

The following list provides the details of the link-state update packet fields:

- *OSPF header*: This header is 24 bytes.
- *Number of advertisements*: This field specifies the number of LSAs included in this packet.
- *Link-state advertisements*: This field contains the full LSAs as described in OSPF LSA formats. Each update can carry multiple LSAs, up to the maximum packet size allowed on the link.

Link-State Acknowledgment Packet

Type 1	Type 2	Type 3	Type 4	Type 5
Hello	Database Description	Link-State Request	Link-State Update	Link-State Acknowledgment

- Link-state acknowledgment packets are received in response to link-state update packets

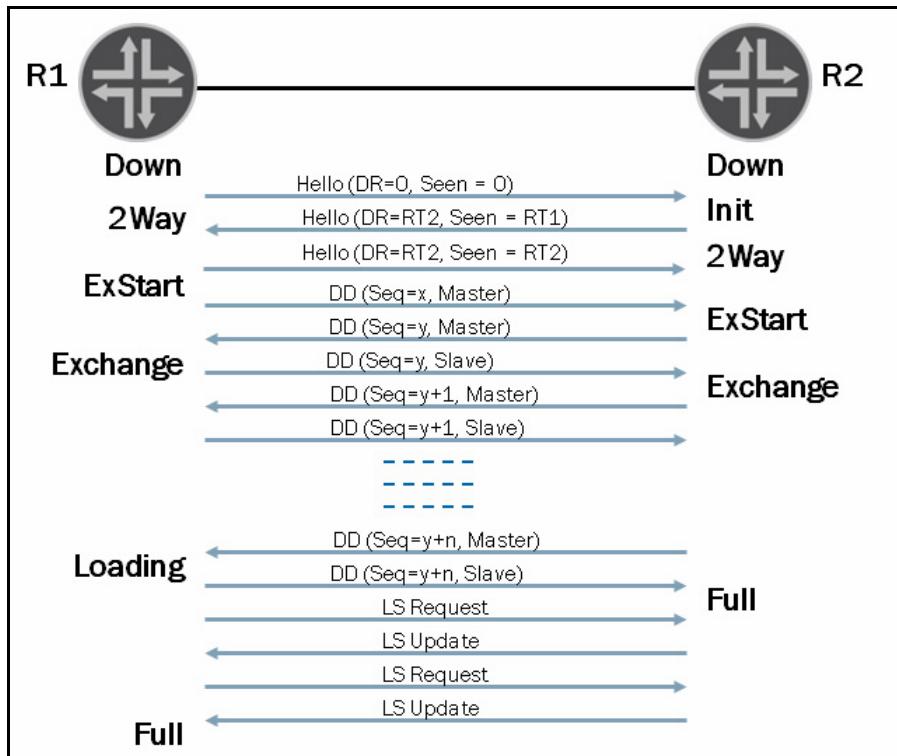
- A single acknowledgment packet can include responses to multiple update packets and consist of the OSPF header and the LSA header



OSPF sends link-state acknowledgment packets after the receipt of a link-state update packet. OSPF routers that receive link-state update packets send link-state acknowledgment packets in unicast fashion back to the originating system to acknowledge that the link-state update packet was received. This acknowledgment is the basis for the reliable flooding in OSPF. An individual link-state acknowledgment packet can contain an acknowledgment for a single link-state update packet or for multiple link-state update packets.

The link-state acknowledgment packet consists of nothing more than an OSPF packet header and a list of LSA headers.

Forming an Adjacency

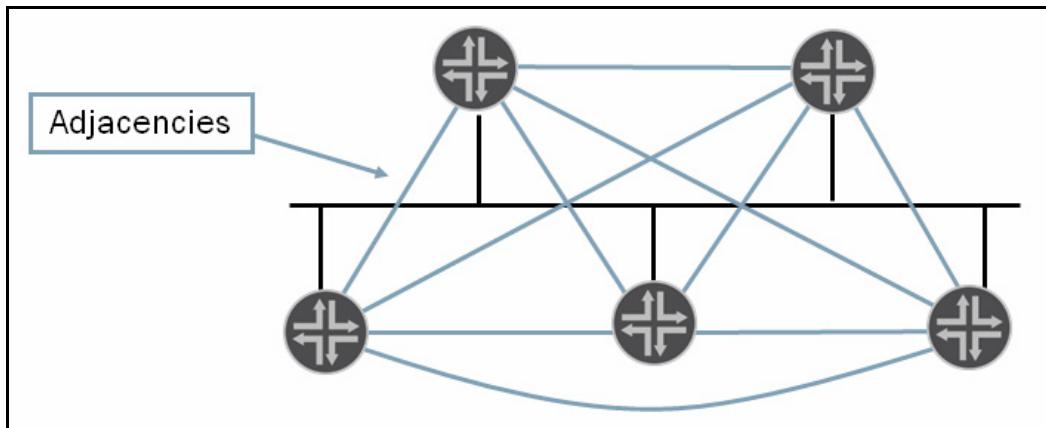


Before any link-state information can be flooded by an OSPF router, that network link must be eligible for flooding. The key to eligibility for OSPF is the adjacency, which is the relationship formed between two OSPF-speaking routers. This adjacency ensures that both routers know about each other and can agree on certain parameters about the link. This agreement assures that data traffic can be transmitted reliably across that link.

Seven possible adjacency states in OSPF exist. These states describe the process of an OSPF adjacency formation:

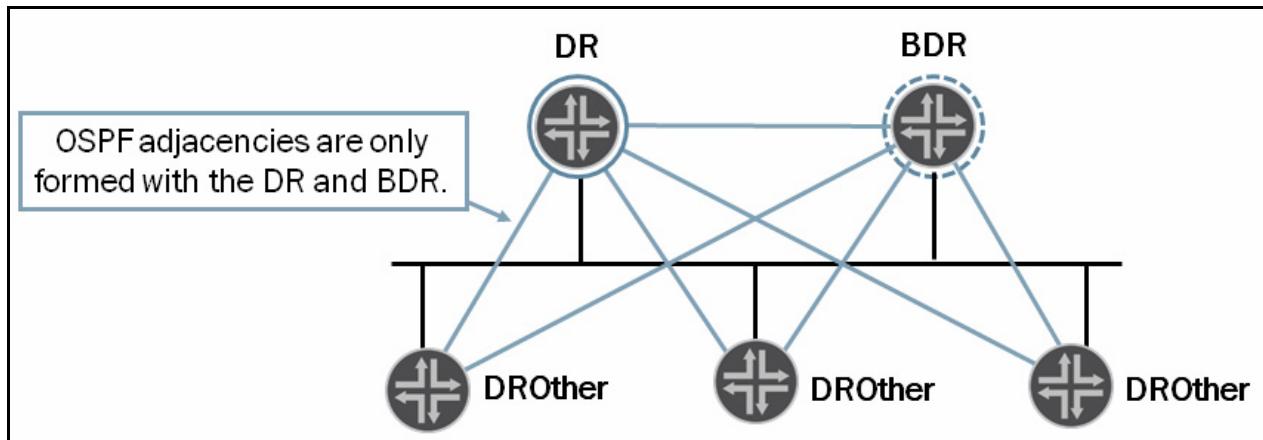
1. **Down:** This initial state indicates that OSPF is waiting for a start event.
2. **Init:** This state indicates that a hello packet has been sent, but bidirectional communication has not been established.
3. **2Way:** This state indicates that a hello packet has been received with the router's RID listed in the neighbor section. Bidirectional communication has been achieved.
4. **ExStart:** This state indicates that the routers negotiate between themselves to determine which router is in charge of the database synchronization process.
5. **Exchange:** This state indicates that the routers exchange LSA headers describing their own databases. If a router does not know about a received LSA header, it can transmit a link-state request for the complete information.
6. **Loading:** This state indicates a router has finished transmitting its database to its peer but is still receiving database information.
7. **Full:** This state indicates that the databases are now fully synchronized. The network link now can be advertised to the OSPF network.

Adjacency Optimization: Part 1



OSPF routers want to form an adjacency with all other routers with which they exchange hello packets. On a broadcast medium such as Ethernet, this desire can pose quite a problem. As more routers are added to the link, more adjacencies must be formed. This full-mesh requirement places extra load on the routers with little extra benefit because they all are advertising the same link information.

Adjacency Optimization: Part 2



To avoid the problem described on the previous page, OSPF has a single router represent the broadcast link to the rest of the network. This router is named the designated router. It is the designated router's job to form an adjacency with all other OSPF routers on the link and to advertise the link-state information to the AS.

Using a designated router to represent the broadcast link can significantly reduce traffic on the segment. To avoid issues during a failure, a backup designated router (BDR) is also elected. The BDR also forms an adjacency with all other OSPF routers on the segment. The BDR does not, however, advertise the learned link-state information to the AS but rather is ready to assume the role of the designated router if a failure should occur.

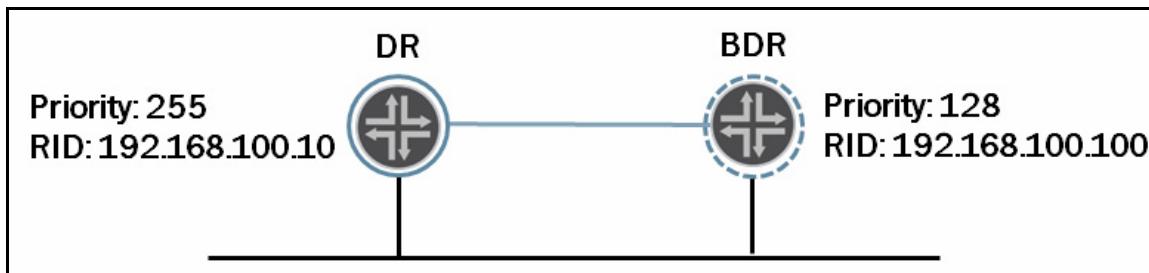
Adjacency Optimization: Part 3

```
[edit protocols ospf]
user@R1# show
area 0.0.0.0 {
    interface ge-1/0/0.0 {
        interface-type p2p;
    }
}
```

user@R1> show ospf interface					
Interface	State	Area	DR ID	BDR ID	Nbrs
ge-1/0/0.0	PtToPt	0.0.0.0	0.0.0.0	0.0.0.0	1

Adding interface-type p2p to a link tells OSPF not to perform a DR/BDR election on that link. This step can save up to forty seconds of wait time for the adjacency to form. Another benefit of the interface-type p2p option is that no Type 2 LSA is generated describing the multi-access segment, helping to reduce the size of the OSPF LSDB.

Electing the Designated Router



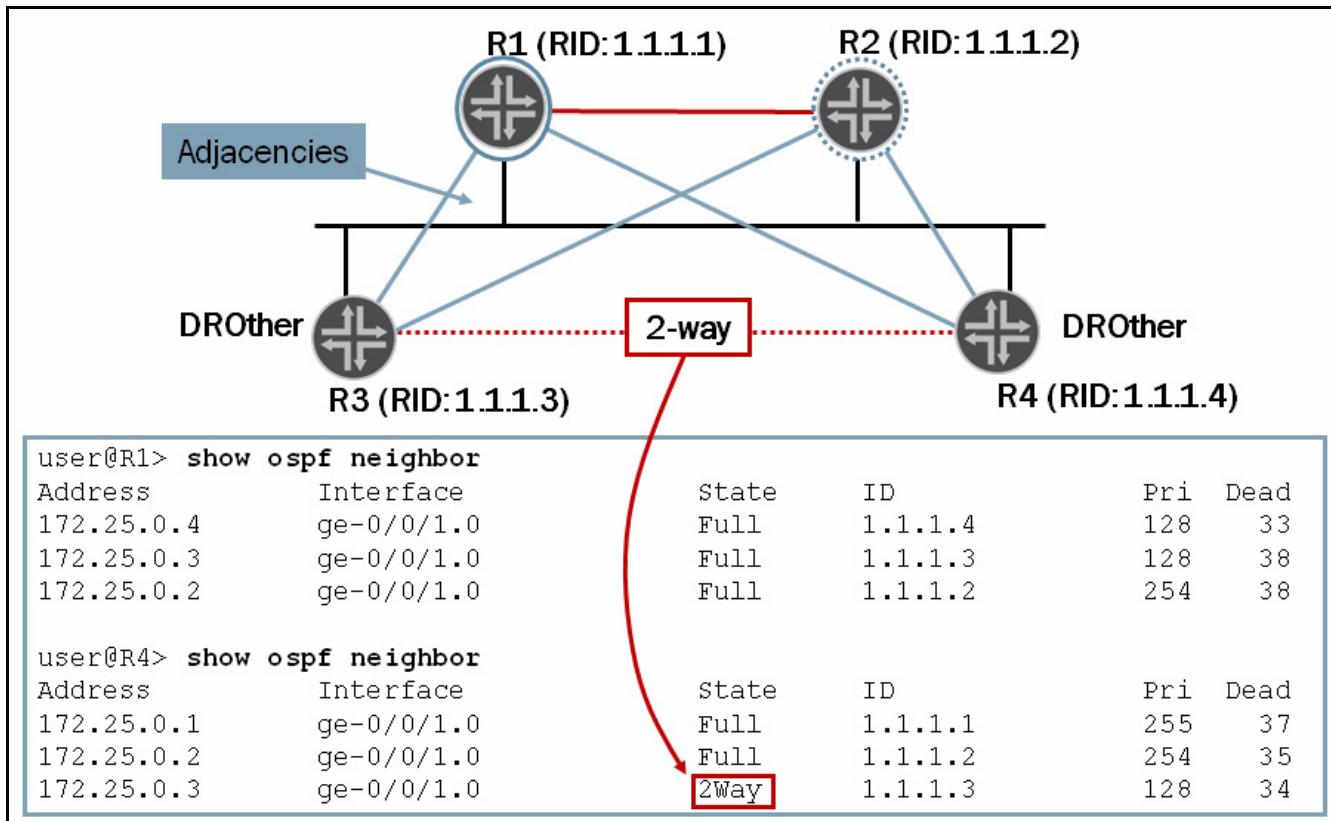
OSPF bases the election of the designated router on two election criteria: priority and RID. OSPF DR priorities can range from 0 through 255, with the Junos operating system default being 128. A router with a higher priority has a better chance of becoming the designated router because priority is the first tiebreaker in a DR election. A router with a DR priority of 0 is not eligible for election and never becomes the designated router. In the event of a priority tie, the second tiebreaker for DR elections is the RID of the routers—the higher the value of the RID, the better the chance of becoming the designated router for the segment.

The election of the designated router on an OSPF link is a nondeterministic event. To avoid network instabilities, the current designated router always operates until it leaves the network. Thus, if a router with a DR priority of 250 comes online and sees that there is already a designated router present on the segment, it does not assume the designated router role.

The election of the very first designated router on a segment occurs within 40 seconds of the first router transmitting a hello packet. This wait time is honored every time an election is held.

Once the designated router is elected, a BDR is elected using the same election rules. It is the BDR's responsibility to monitor the designated router and ensure that it is working properly. If the BDR notices that the designated router has left the network, it automatically assumes the role of the designated router. A new BDR is then elected on the segment. Again, the BDR election is nondeterministic.

OSPF Neighbor Relationship

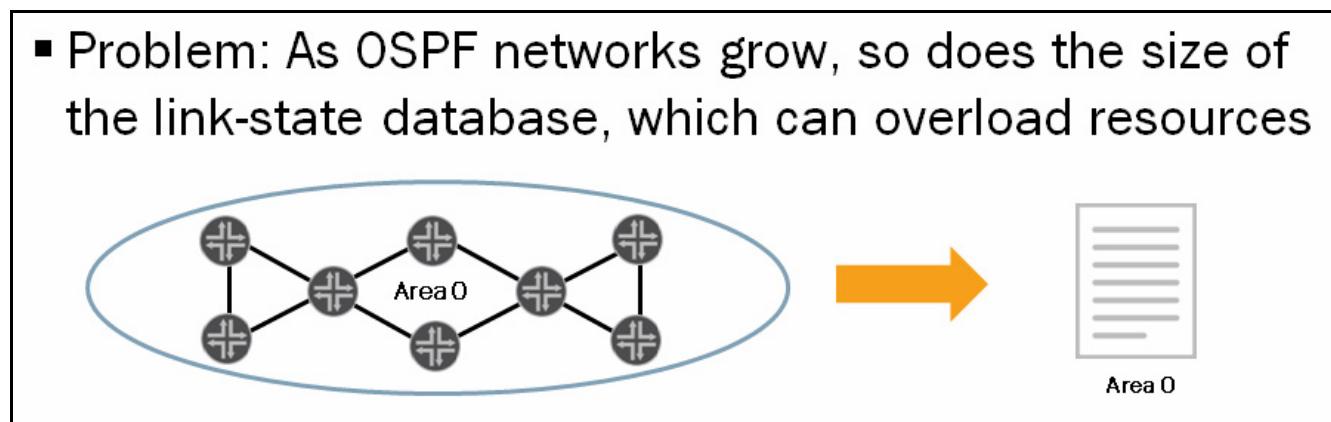


As soon as an OSPF router sees a hello packet on an interface, it starts to retain knowledge of that neighbor. You can display this information with the operational CLI command **show ospf neighbor**. The graphic displays an example of this command. The address column is the interface IP address of the neighboring router. The ID column is the router ID (RID) of the neighboring router.

In the graphic, we see four OSPF routers connected to the same network segment. R1 and R2 are elected as the designated router and the BDR respectively. Because both R3 and R4 are DROthers (that is, neither of them are elected as the DR or BDR), they do not form a full adjacency with each other. DROther devices form the two-way neighbor state with all other DROther devices. We illustrate this point in the graphic.

OSPF Scalability

- Problem: As OSPF networks grow, so does the size of the link-state database, which can overload resources



With a link-state protocol, flooding link-state update packets and running the SPF algorithm consume router resources. As the size of the network grows and more routers are added to the AS, this use of resources becomes a bigger and bigger issue. This issue stems from the RFC requirement that all OSPF routers share an identical LSDB. Eventually, the routers spend so much time flooding and running the SPF algorithm that they cannot route data packets. Clearly, this scenario is not optimal.

Shrinking the Link-State Database

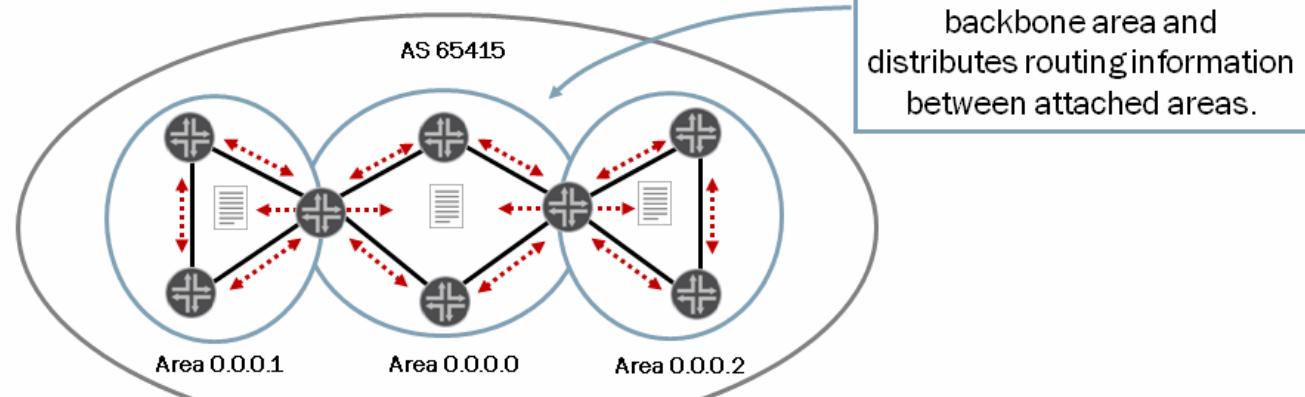
- Solution: Implement OSPF areas to shrink the size of the link-state database



The solution to this problem (and link-state protocol scalability in general) is to reduce the size of the LSDB. You can reduce size of the LSDB using multiple OSPF areas rather than a single area that encompasses the entire AS. Note that the type of OSPF areas used is important when attempting to shrink the size of the LSDB. We discuss the various OSPF area types on a subsequent page.

In addition to adding new OSPF areas that restrict LSA flooding, you can also perform route summarization on the borders between OSPF areas. Route summarization has two key benefits: 1) it reduces the size of the LSDB; and 2) it can hide some instabilities in one OSPF area from all other OSPF areas. For route summarization to be effective, you must carefully plan the addressing within your OSPF network so that subnets can be more easily summarized. Complete coverage of route summarization is beyond the scope of this study guide.

OSPF Areas



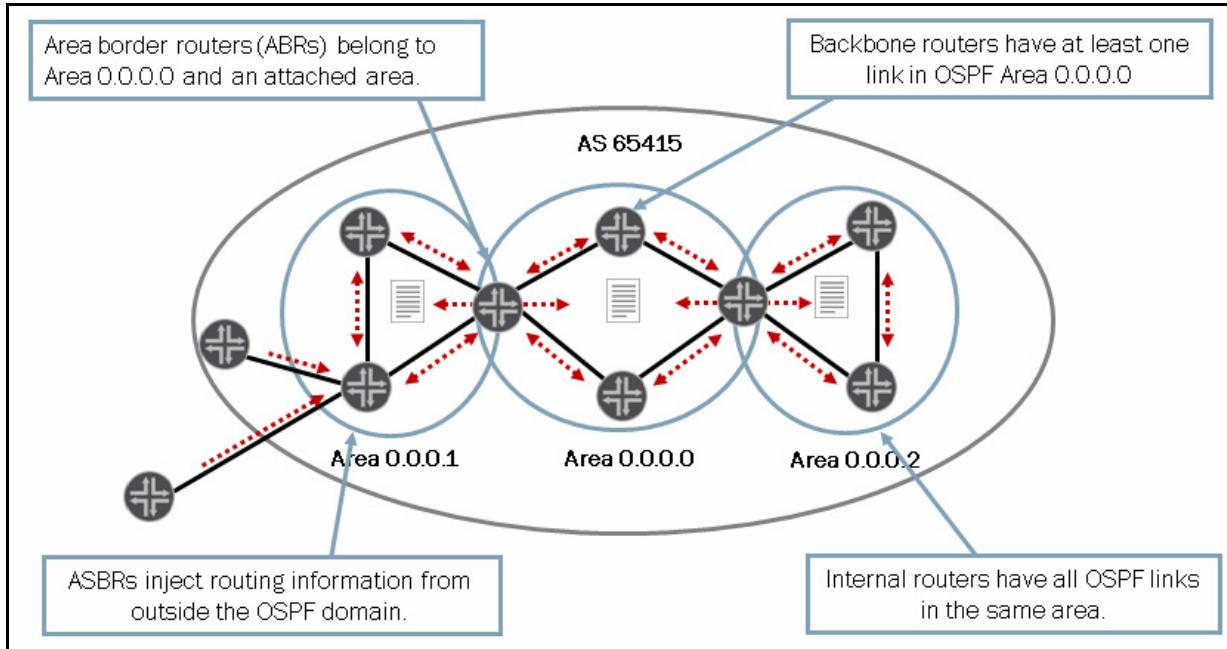
- Areas

- An AS can be divided into smaller groups called areas
- LSA flooding can be constrained to an area, which effectively reduces the size of the link-state database
- All routers maintain an identical copy of the link-state database on a per-area basis

Using OSPF, you can segment an AS into smaller groups referred to as areas. Using areas achieves the OSPF hierarchy that can facilitate growth and scalability. You can constrain LSA flooding by using multiple areas, which can effectively reduce the size of the LSDB on an individual OSPF router. Each OSPF router maintains a separate and identical LSDB for each area to which it is connected.

To ensure correct routing knowledge and connectivity, OSPF maintains a special area referred to as the backbone area. The backbone area is designated as Area 0.0.0.0 (or simply Area 0). All other OSPF areas must connect themselves to the backbone for connectivity. By default, all data traffic between OSPF areas transits the backbone. You can alter this default behavior and eliminate the requirement of all interarea traffic transiting Area 0.0.0.0 by configuring a multi-area adjacency on the same logical interface. The multi-area adjacency feature is documented in RFC 5185 and is beyond the scope of this study guide.

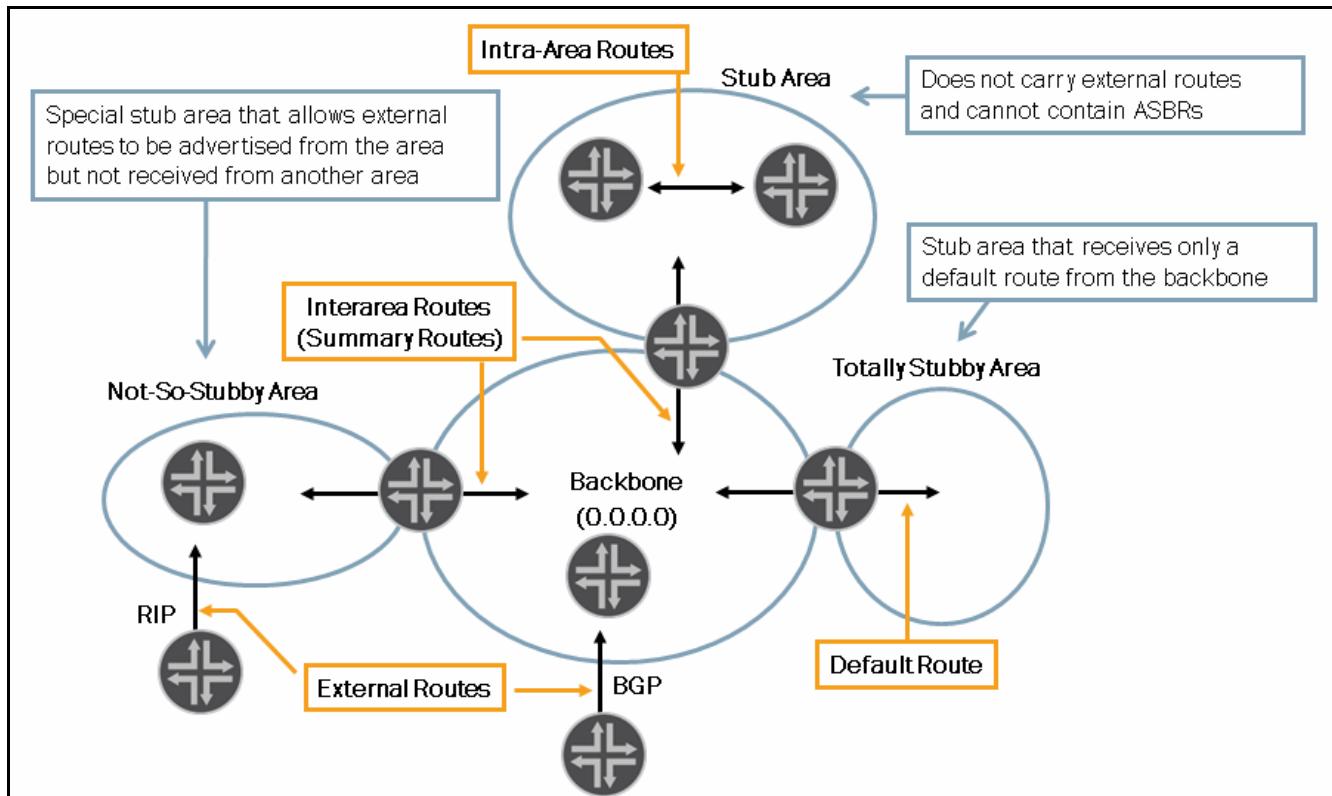
OSPF Routers



OSPF routers can take on a number of different roles within an OSPF domain. The following list shows the common types of OSPF routers along with a brief description:

- **Area border router (ABR):** An OSPF router with links in two areas, the ABR is responsible for connecting OSPF areas to the backbone. It transmits network information between the backbone and other areas.
- **Autonomous system boundary router (ASBR):** An OSPF router that injects routing information from outside the OSPF AS, an ASBR is typically located in the backbone. However, the OSPF specification allows an ASBR to be in other areas as well.
- **Backbone router:** Defined as any OSPF router with a link to Area 0 (the backbone), this router can also be an internal or area border router depending on whether it has links to other, nonbackbone areas.
- **Internal router:** An internal router is an OSPF router with all its links within an area. If that router is located within the backbone area (Area 0.0.0.0), it is also known as a backbone router.

OSPF Area Types



This graphic introduces some OSPF area types and illustrates the relationship between these areas, including the types of routes exchanged between them. In the graphic, all areas are connected directly to the backbone. In the rare situations where a new area is introduced that cannot have direct physical access to the backbone, you can configure a virtual link. The coverage of virtual links is beyond the scope of this study guide.

OSPF classifies different types of routing information as follows:

- Routes that are generated from within an area, where the destination belongs to the area, are referred to as *intra-area*, or *internal*, routes.
- Routes that originate from other areas are referred to as *interarea* or *summary* routes.
- Routes that originate from other routing protocols, or different OSPF processes, and that are injected into OSPF through redistribution, are referred to as *external* routes.

Stub areas are areas through which, or into which, AS external advertisements are not flooded (LSA types 4 and 5). You might want to create stub areas when much of the topological database consists of AS external advertisements. Doing so reduces the size of the topological databases and, therefore, the amount of memory required on the internal routers in the stub area.

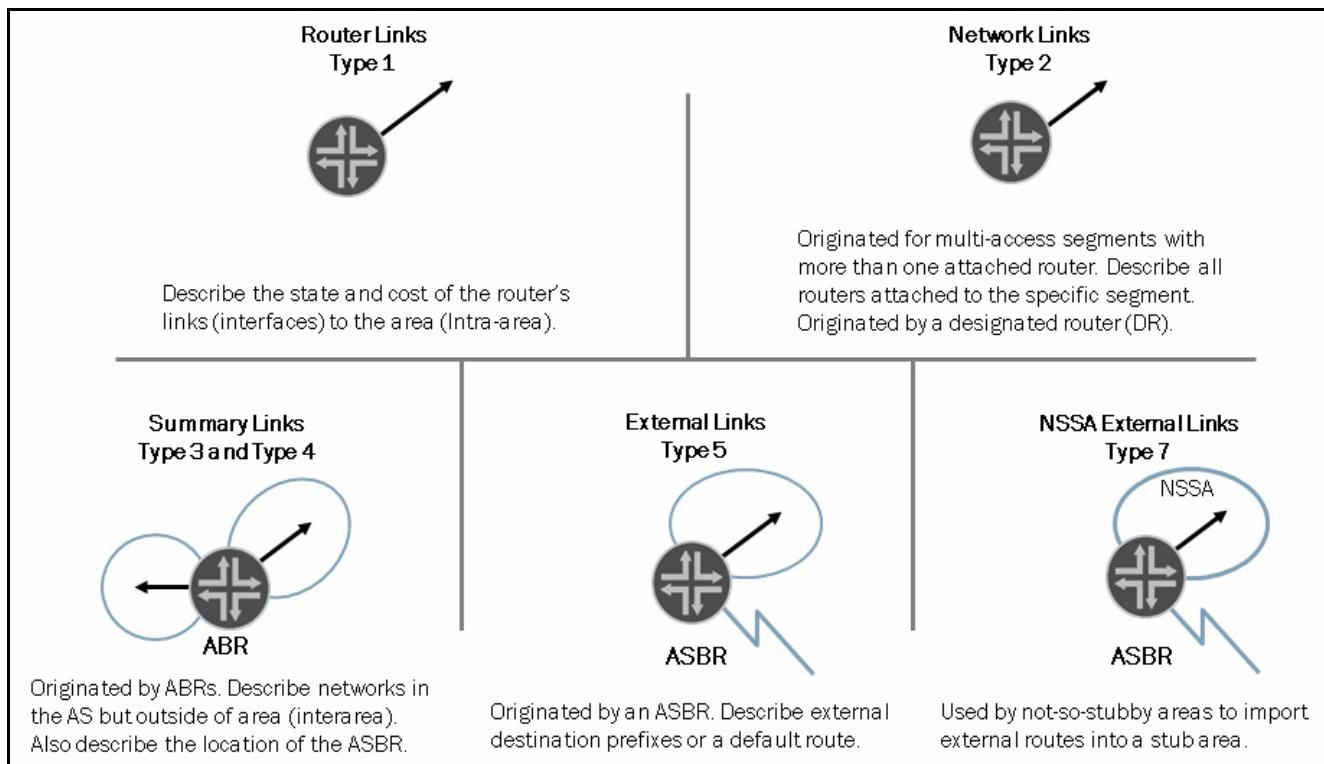
When you configure an ABR for stub area operation, a default route is normally advertised in the place of the external routes that are blocked from stub areas. The default route provides routers in the stub area with reachability to external destinations. In the Junos OS, ABRs require explicit configuration for default route generation.

Note that you cannot create a virtual link through a stub area, and a stub area cannot contain an AS boundary router.

A totally stubby area is a stub area that receives only the default route from the backbone. ABRs do not flood LSA types 3, 4, or 5 into totally stubby areas.

An OSPF stub area has no external routes in it, so you cannot redistribute routes from another protocol into a stub area. A not-so-stubby-area (NSSA) allows external routes to be flooded within the area. These routes are then leaked into other areas. However, external routes from other areas still do not enter the NSSA. (ABR does not flood LSA types 4 and 5 into an attached NSSA.)

An Overview of the LSA Packet Types)



The graphic highlights the LSA types used in modern OSPF networks. These LSAs each represent a portion of the OSPF network and are flooded into the AS based on the function of the router.

We highlight some key details for these LSA packet types:

- **Router (type 1):** Router LSAs describe the interfaces and neighbors of each OSPF router to all other OSPF routers within the same area (intra-area).
- **Network (type 2):** Network LSAs describe an Ethernet segment. These LSAs are sent by the designated router to other OSPF routers within the same area (intra-area).
- **Summary (type 3):** Summary LSAs describe IP prefixes learned from Router and Network LSAs. These LSAs are sent by the ABR attached to the area from where the prefix information was learned and sent to other OSPF areas (interarea). Note that as summary LSAs are re-injected into different areas, the LSA type never changes, but the cost and advertising router details do change.
- **ASBR Summary (type 4):** ASBR Summary LSAs describe the router-id of ASBR routers located in remote areas. These LSAs are sent by the ABR attached to the area in which the ASBR is located to other OSPF areas (interarea). Note that as ASBR summary LSAs are re-injected into different areas, the LSA type never changes, but the cost and advertising router details do change.
- **External (type 5):** External LSAs describe IP prefixes redistributed from other routing protocols, such as RIP, BGP, or even static routes. These LSAs are sent by ASBRs injecting the external routes into OSPF. By default, the Junos OS marks these LSAs with the type 2 designation, which means the cost of the associated OSPF route is not added. You can alter this default behavior and mark these external prefixes with the type 1 designation, which means the cost to the ASBR will be included. External LSAs are flooded to all OSPF areas except areas defined as stub areas.
- **NSSA External (type 7):** NSSA External LSAs are similar to External LSAs (type 5) in that they describe IP prefixes redistributed from other routing protocols, such as RIP, BGP, or even static routes. These LSAs are sent by ASBRs in NSSA areas. These LSAs are translated to type 5 LSAs by the ABR attached to NSSA area in which the type 7 LSAs originate.

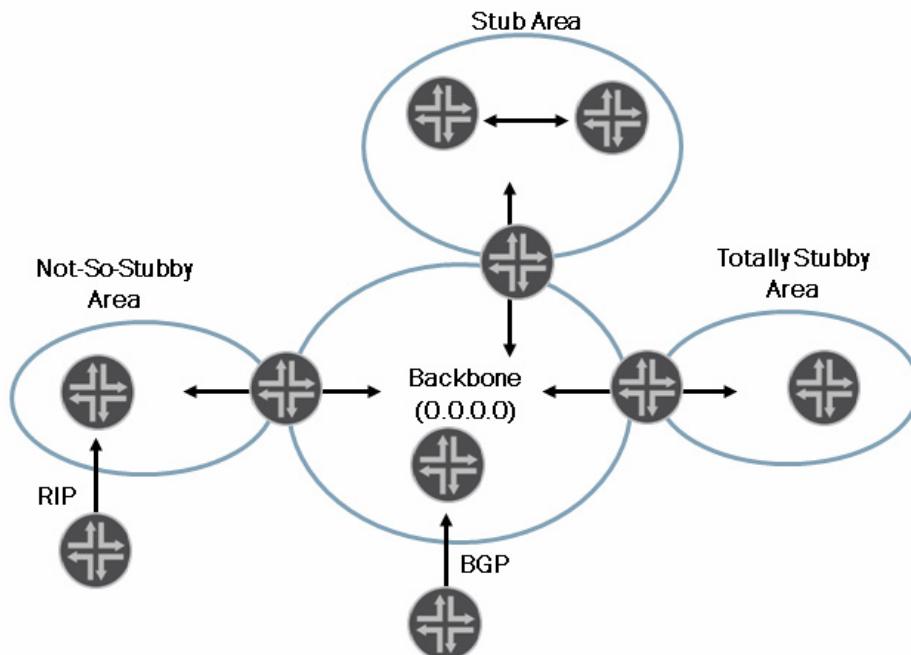
In addition to the LSA types already discussed, the OSPF specification also includes the following LSA types:

- **Type 6:** Multicast OSPF LSA;
- **Type 8:** External attributes LSA;

- Type 9: Opaque LSA (link scope);
- Type 10: Opaque LSA (area scope—used for traffic engineering); and
- Type 11: Opaque LSA (AS scope).

Test Your Knowledge

- Which of the recently discussed LSAs would you expect to find in each of the listed areas?



Use this graphic to validate your understanding of what types of LSAs can be found in the various OSPF area types.

Remember that stub areas are areas through which, or into which, AS external advertisements are not flooded (LSA type 4 and type 5). Stub areas do, however, include LSA type 1, type 2, and type 3. One of the type 3 LSAs is typically used to represent a default route used for external reachability.

A totally stubby area is a stub area that receives only the default route from the backbone. ABRs do not flood LSA type 3 (other than the type 3 LSA representing the default route), type 4, or type 5 into totally stubby areas.

An NSSA area allows type 7 LSAs, which represent external routes flooded within the area, along with type 1, type 2, and type 3 LSAs. NSSA areas do not allow LSA type 4 and type 5. Note that the Junos OS allows you to also restrict type 3 LSAs in NSSA areas using the **no-summaries** configuration option.

Junos OS OSPF Support

The Junos OS supports OSPFv2 and OSPFv3. OSPFv2 was designed for IPv4, while OSPFv3 works with both IPv4 and IPv6. IPv6 support can vary between software versions and Junos OS-based devices. Refer to the appropriate technical publications for your specific product and software version for details.

The inset lists some of the key features supported by the Junos OS. We described the various OSPF area types on a previous page in the last section. The following are short descriptions of the other features:

- **Authentication:** All OSPFv2 protocol exchanges can be authenticated to guarantee that only trusted routers participate in the AS's routing. By default, OSPFv2 authentication is disabled. The Junos OS supports simple authentication, Message Digest 5 (MD5) authentication, and IP Security (IPsec) authentication.
- **Summarization:** ABRs send summary link advertisements to describe the routes to other areas. To minimize the number of these advertisements that are flooded, you can configure the router to coalesce, or summarize, a range of IP addresses and send reachability information about these addresses in a single LSA.

- **External prefix limits:** By default, there is no limit to the number of prefixes that can be exported into OSPF. You can use the **prefix-export-limit** statement to limit the number of external prefixes allowed in the OSPF network.
- **Graceful restart (GR):** With graceful restart enabled, a restarting router informs the neighbors before restarting. The neighbors act as if the router is still within the network topology, and continue forwarding traffic to the restarting router. A grace period is set to specify the time period for which the neighbors should consider the restarting router as part of the topology. Graceful restart is not enabled by default.
- **Bidirectional Forwarding Detection (BFD):** This protocol is a simple hello mechanism that detects failures in a network. BFD works with a wide variety of network environments and topologies. Hello packets are sent at a specified, regular interval. A neighbor failure is detected when the router stops receiving a reply after a specified interval. The BFD failure-detection timers have shorter time limits than the OSPF failure-detection mechanisms, providing faster detection. These timers are also adaptive. For example, the timer can adapt to a higher value if an adjacency fails, or a neighbor can negotiate a higher value than the one configured.

Basic Configuration Example

```
[edit protocols]
user@R1# show
ospf {
    area <area-id> {
        <area options>;
        interface <interface-name> {
            <interface options>;
        }
    }
}
ospf3 {
    area <area-id> {
        <area options>;
        interface <interface-name> {
            <interface options>;
        }
    }
}
```

Used for IPv4 routing environments

Used for IPv4 or IPv6 routing environments

The inset illustrates the basic hierarchy and syntax used to configure OSPFv2 and OSPFv3.

Determining the Router ID

```
[edit routing-options]
user@R1# show
router-id 192.168.100.1;
```

The RID is a 32-bit number in dotted quad notation.

Every OSPF router has a single RID; it is a 32-bit number in dotted quad notation. OSPF uses this RID for several purposes, including designated router election and LSA originator identification.

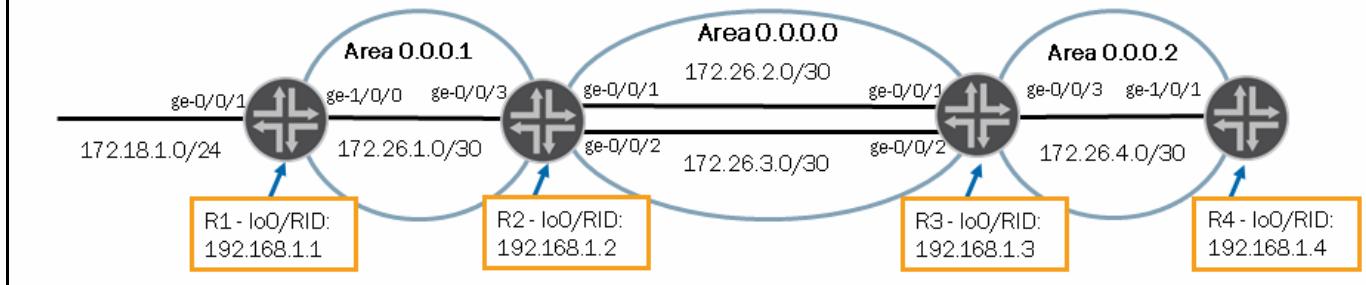
You can explicitly configure the RID under the [edit routing-options] hierarchy. However, if you do not manually define a RID, the Junos OS uses one of the IP addresses configured on the router at the time the OSPF process starts. If the loopback interface is configured with a non-127/8 IP address, the Junos OS uses that address first. If a loopback interface is not configured, the Junos OS uses the next suitable address, typically the dedicated management interface. Remember, the dedicated management interface does not support transit traffic, so sourcing the RID from that interface can result in the inability to ping the RID. For this reason and also because IP address definitions can change, we highly recommend that you configure a RID.

If you desire reachability through OSPF for your network to a defined loopback IP address, you must include the lo0 interface in OSPF. Note that if the IP address assigned to the lo0 interface uses a subnet mask less specific than /32, then two prefixes will be advertised into OSPF; one prefix for the defined subnet and one for the associated /32 host prefix. As an example, if lo0.0 were assigned the IP address 172.27.1.1/24 and were included in OSPF then the 172.27.1.1/24 and 172.27.1.1/32 prefixes would be advertised into OSPF.

Case Study: Topology and Objectives

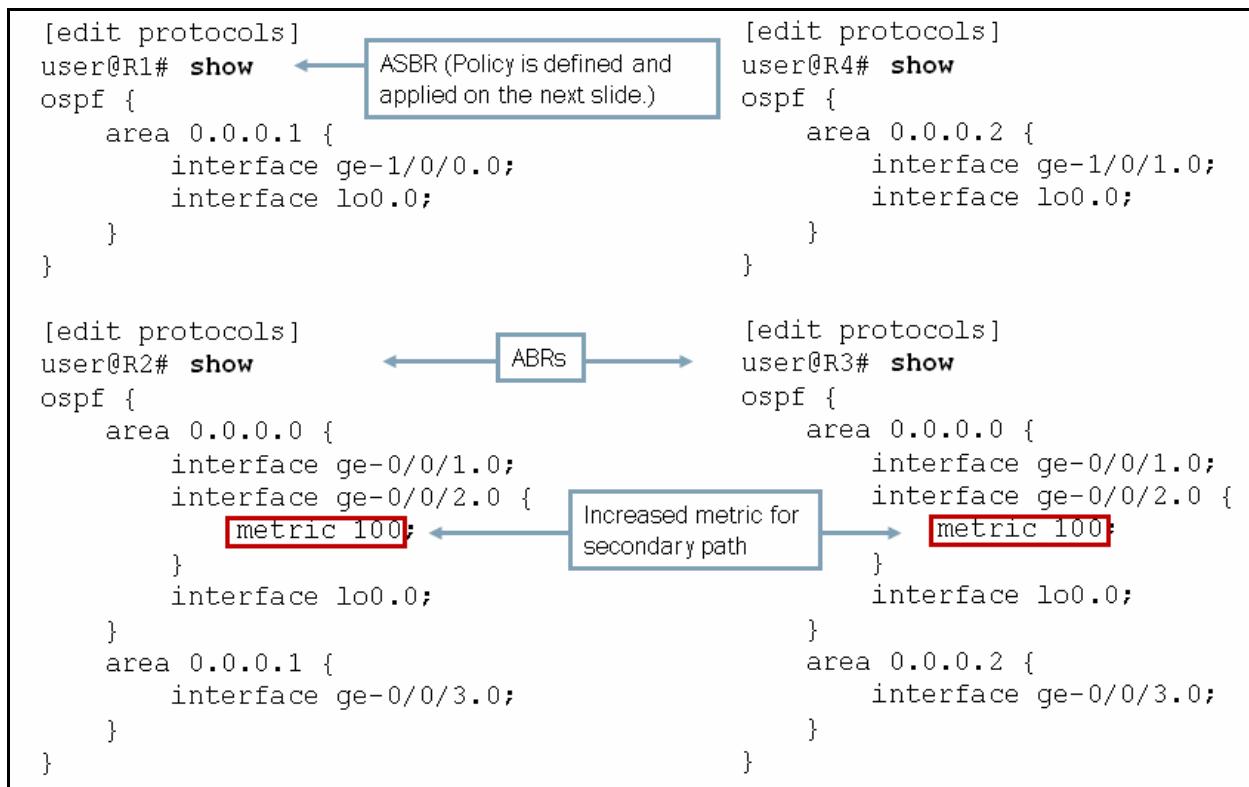
- Use the following topology as a guide to implement OSPF

- On R1, redistribute the 172.18.1.0/24 network and ensure that it is installed as an *external* OSPF route
- Use metrics to ensure that the path using the ge-0/0/1 interfaces within the backbone area is preferred



The graphic displays the topology and objectives for our case study.

Case Study: Configuring OSPF



The inset displays the required configuration to accomplish some of the stated objectives on the previous page. Here we can see the OSPF-related configuration for all four routers. We know that R1 will become an ASBR because of the stated objective that it should redistribute its connected route into OSPF as an external OSPF route. We will define and apply the redistribution policy on the next page. R2 and R3 are designated as ABRs because they both connect to the backbone area (Area 0.0.0.0) and a nonbackbone area (Area 0.0.0.1 and Area 0.0.0.2 respectively).

The Junos OS does not require you to include the entire dotted quad notation when defining an OSPF area. In other words, if you issue the command **set protocols ospf area 0 interface ge-0/0/0.0**, the software interprets area 0 as area 0.0.0.0 and generates the following configuration stanza:

```
[edit]
user@R1# show protocols
ospf {
    area 0.0.0.0 {
        interface ge-0/0/0.0;
    }
}
```

The inset also displays an increased metric on both R2 and R3 for the ge-0/0/2 interfaces to make the path using the ge-0/0/1 interfaces more preferred. All OSPF interfaces have a cost, which is a routing metric that is used in the link-state calculation. Routes with lower total path metrics are preferred over those with higher path metrics. The cost of a route is described by a single dimensionless metric that is determined using the following formula:

$$\text{cost} = \text{reference-bandwidth} / \text{bandwidth}$$

You can modify the reference-bandwidth value. The bandwidth value refers to the actual bandwidth of the physical interface. You can override the default behavior of using the reference bandwidth to calculate the metric cost of a route by configuring a specific metric value for any OSPF interface. To modify the reference bandwidth, include the **reference-bandwidth** statement as shown below:

```
[edit protocols ospf]
user@R1# set reference-bandwidth ?
Possible completions:
<reference-bandwidth> Bandwidth for calculating metric defaults
```

The default value of the reference bandwidth is 100 Mbps (which you specify as 100,000,000), which gives a metric of 1 for any interface with a physical bandwidth that is 100 Mbps or greater. For reference-bandwidth, you can configure a value from 9600 through 1,000,000,000,000 bits. For example, if you set the reference bandwidth to 1 Gbps (that is, reference-bandwidth is set to 1,000,000,000), a 100-Mbps interface has a default metric of 10. By default, the loopback interface (lo0) metric is 0. No bandwidth is associated with the loopback interface. You can specify a metric value from 1 through 65,535.

Case Study: Defining and Applying the Redistribution Policy

```
[edit policy-options]
user@R1# show
policy-statement 2ospf {
    term match-direct-route {
        from {
            protocol direct;
            route-filter 172.18.1.0/24 exact;
        }
        then accept;
    }
}

[edit protocols]
user@R1# show
ospf {
    export 2ospf; ← Redistribution policy is applied under [edit protocols ospf] hierarchy.
    area 0.0.0.1 {
        interface ge-1/0/0.0;
        interface lo0.0;
    }
}
```

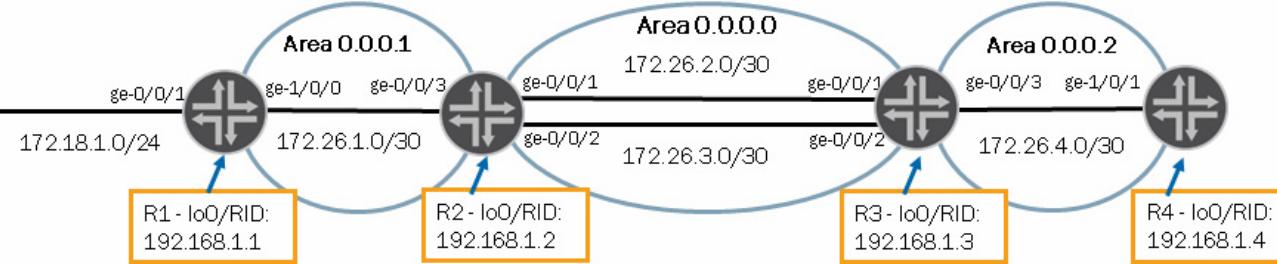
The inset displays the remainder of the required configuration used to accomplish the stated objectives for this case study. Remember that a **commit** is required to activate all configuration changes on all participating devices.

Test Your Knowledge

- What configuration option allows R1 to inject the 172.18.1.0/24 prefix into OSPF as an internal OSPF route while prohibiting adjacency formation?

```
[edit protocols]
user@R1# set ospf area 1 interface ge-0/0/1.0 passive
```

Include the passive option for the interface

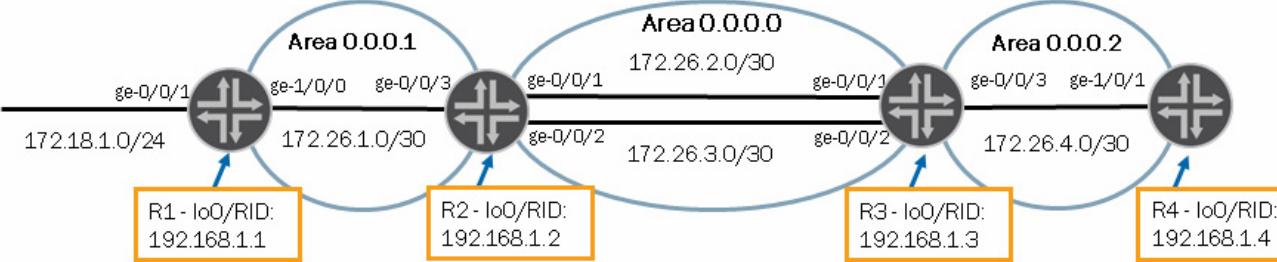


If you want to inject an interface route into OSPF as an internal OSPF route while prohibiting adjacency formation for that interface, you should include the interface under the `[edit protocols ospf area area-number]` hierarchy level along with the **passive** configuration option. The **passive** configuration option is covered in the *Junos Routing Essentials* (JRE) course.

Case Study: Displaying OSPF Adjacency Information

- Use the **show ospf neighbor** command to display OSPF adjacency information

```
user@R2> show ospf neighbor
Address           Interface          State      ID          Pri  Dead
172.26.2.2       ge-0/0/1.0        Full      192.168.1.3   128  39
172.26.3.2       ge-0/0/2.0        Full      192.168.1.3   128  36
172.26.1.1       ge-0/0/3.0        Full      192.168.1.1   128  34
```



Once the required configuration is in place on the participating OSPF devices, you should confirm the state of the OSPF adjacencies using the **show ospf neighbor** command, as shown in the graphic. You can also include the **detail** or

extensive options with the **show ospf neighbor** command to obtain additional information, as shown in the following output:

```
user@R2> show ospf neighbor extensive
Address           Interface          State      ID          Pri  Dead
172.26.1.1       ge-0/0/3.0        Full       192.168.1.1   128   33
  Area 0.0.0.1, opt 0x42, DR 172.26.1.2, BDR 172.26.1.1
  Up 22:01:45, adjacent 22:01:37
  Topology default (ID 0) -> Bidirectional
172.26.2.2       ge-0/0/1.0        Full       192.168.1.3   128   32
  Area 0.0.0.0, opt 0x42, DR 172.26.2.2, BDR 172.26.2.1
  Up 1d 03:41:28, adjacent 1d 03:41:28
  Topology default (ID 0) -> Bidirectional
172.26.3.2       ge-0/0/2.0        Full       192.168.1.3   128   34
  Area 0.0.0.0, opt 0x42, DR 172.26.3.2, BDR 172.26.3.1
  Up 1d 03:43:14, adjacent 1d 03:43:14
  Topology default (ID 0) -> Bidirectional
```

The output fields associated with these commands include the following:

- **Address:** Displays the address of the neighbor.
- **Intf:** Displays the interface through which the neighbor is reachable.
- **State:** Displays the state of the neighbor, which can be **Attempt**, **Down**, **Exchange**, **ExStart**, **Full**, **Init**, **Loading**, or **2Way**.
- **ID:** Displays the RID of the neighbor.
- **Pri:** Displays the priority of the neighbor to become the designated router.
- **Dead:** Displays the number of seconds until the neighbor becomes unreachable.
- **area** (detail and extensive output only): Displays the area in which the neighbor is located.
- **opt** (detail and extensive output only): Displays the option bits from the neighbor.
- **DR** (detail and extensive output only): Displays the address of the designated router.
- **BDR** (detail and extensive output only): Displays the address of the BDR.
- **Up** (detail and extensive output only): Displays the length of time since the neighbor came up.
- **adjacent** (detail and extensive output only): Displays the length of time since the adjacency with the neighbor was established.

In some instances, you might need to reform OSPF adjacencies. You can use the **clear ospf neighbor** command to clear existing neighbor adjacencies. Once an adjacency is cleared, it should be reformed immediately. Use this command as needed, but remember that it can cause a momentary disruption.

Case Study: Verifying the Route Entries and Selected Paths

- Use **show route** commands to verify route entries and their selected paths

```
user@R2> show route 172.18.1.0/24
```

External prefix injected by R1

```
inet.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
172.18.1.0/24      *[OSPF/150] 02:37:46, metric 0, tag 0
                   > to 172.26.1.1 via ge-0/0/3.0
```

```
user@R2> show route 172.26.4.0/30
```

Remote subnet connecting R3 and R4 is reachable through desired path.

```
inet.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
172.26.4.0/30      *[OSPF/10] 02:24:29, metric 2
                   > to 172.26.2.2 via [ge-0/0/1.0]
```

The inset illustrates the **show route** commands used to verify that the required route entries are installed and that the desired paths are selected for those routes.

Other Key Monitoring Commands

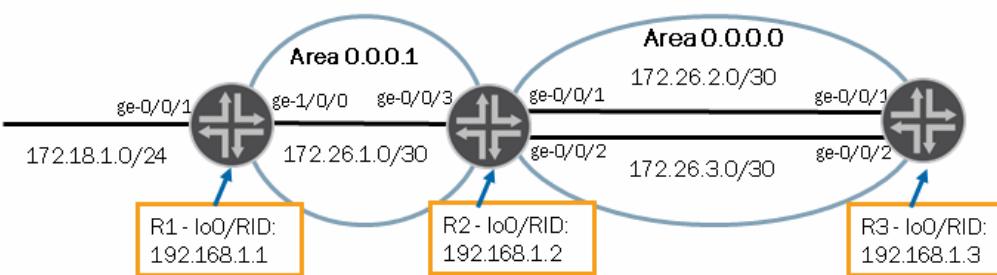
As illustrated in the inset, additional **show** commands exist that can provide detailed information for the OSPF operation on your device. We cover these highlighted commands and some sample outputs on subsequent pages.

- **show ospf route**
- **show ospf database**
- **show ospf statistics**
- **show ospf log**

Displaying OSPF Interface Parameters

```
user@R2> show ospf interface
```

Interface	State	Area	DR ID	BDR ID	Nbrs
ge-0/0/1.0	BDR	0.0.0.0	192.168.1.3	192.168.1.2	1
ge-0/0/2.0	DR	0.0.0.0	192.168.1.2	192.168.1.3	1
lo0.0	DR	0.0.0.0	192.168.1.2	0.0.0.0	0
ge-0/0/3.0	DR	0.0.0.1	192.168.1.2	192.168.1.1	1



You can add the **detail** or **extensive** options to the **show ospf interface** command to obtain additional information. A sample capture is shown here:

```

user@R2> show ospf interface extensive
Interface          State   Area        DR ID      BDR ID      Nbrs
ge-0/0/3.0         DR      0.0.0.1    192.168.1.2  192.168.1.1  1
  Type: LAN, Address: 172.26.1.2, Mask: 255.255.255.252, MTU: 1500, Cost: 1
  DR addr: 172.26.1.2, BDR addr: 172.26.1.1, Priority: 128, Adj count: 1
  Hello: 10, Dead: 40, ReXmit: 5, Not Stub
  Auth type: None
  Topology default (ID 0) -> Cost: 0
ge-0/0/1.0         BDR     0.0.0.0    192.168.1.3  192.168.1.2  1
  Type: LAN, Address: 172.26.2.1, Mask: 255.255.255.252, MTU: 1500, Cost: 1
  DR addr: 172.26.2.2, BDR addr: 172.26.2.1, Priority: 128, Adj count: 1
  Hello: 10, Dead: 40, ReXmit: 5, Not Stub
  Auth type: None
  Topology default (ID 0) -> Cost: 0
...

```

The output fields of the **show ospf interface** command are the following:

- **Intf:** Displays the name of the interface running OSPF.
- **State:** Displays the state of the interface. It can be BDR, Down, DR, DROther, Loop, PtToPt, or Waiting.
- **Area:** Displays the number of the area in which the interface is located.
- **DR ID:** Displays the address of the area's designated router.
- **BDR ID:** Displays the BDR for a particular subnet.
- **Nbrs:** Displays the number of neighbors on this interface.
- **Type (detail and extensive output only):** Displays the type of interface. It can be LAN, NBMA, P2MP, P2P, or Virtual.
- **address (detail and extensive output only):** Displays the IP address of the neighbor.
- **mask (detail and extensive output only):** Displays the mask of the interface.
- **MTU (detail and extensive output only):** Displays the interface's maximum transmission unit (MTU).
- **cost (detail and extensive output only):** Displays the interface's cost (metric).
- **DR addr (detail and extensive output only):** Displays the address of the designated router.
- **BDR addr:** Displays the address of the BDR.
- **adj count (detail and extensive output only):** Displays the number of adjacent neighbors.
- **Flood list (extensive output only):** Displays the list of LSAs pending flood on this interface.
- **Ack list (extensive output only):** Displays the list of pending acknowledgments on this interface.
- **Descriptor list (extensive output only):** Displays the list of packet descriptors.
- **Dead (detail and extensive output only):** Displays the configured value for the dead timer.
- **Hello (detail and extensive output only):** Displays the configured value for the hello timer.
- **ReXmit (detail and extensive output only):** Displays the configured value for the retransmit timer.
- **OSPF area type (detail and extensive output only):** Displays the type of OSPF area, which can be Stub, Not Stub, or NSSA.

Displaying OSPF Route Information

Topology default Route Table:						
Prefix	Path	Route	NH	Metric	NextHop	Nexthop
	Type	Type	Type		Interface	addr/label
192.168.1.1	Intra	AS BR	IP	1	ge-0/0/3.0	172.26.1.1
192.168.1.3	Intra	Area BR	IP	1	ge-0/0/1.0	172.26.2.2
172.18.1.0/24	Ext2	Network	IP	0	ge-0/0/3.0	172.26.1.1
172.26.1.0/30	Intra	Network	IP	1	ge-0/0/3.0	
172.26.2.0/30	Intra	Network	IP	1	ge-0/0/1.0	
172.26.3.0/30	Intra	Network	IP	100	ge-0/0/2.0	
172.26.4.0/30	Inter	Network	IP	2	ge-0/0/1.0	172.26.2.2
192.168.1.1/32	Intra	Network	IP	1	ge-0/0/3.0	172.26.1.1
192.168.1.2/32	Intra	Network	IP	0	lo0.0	
192.168.1.3/32	Intra	Network	IP	1	ge-0/0/1.0	172.26.2.2
192.168.1.4/32	Inter	Network	IP	2	ge-0/0/1.0	172.26.2.2

External prefix injected by R1

Metric for ge-0/0/2.0 interface was modified in earlier configuration example.

The **show ospf route** command displays routes in the unicast routing table, `inet.0`, installed by OSPF. You can use additional keywords to display only OSPF routes learned by specific LSA types. The following sample capture shows the available keywords for this command:

```
user@R2> show ospf route ?
Possible completions:
<[Enter]>          Execute this command
abr                  Display OSPF routes to area border routers
asbr                 Display OSPF routes to AS border routers
detail               Display detailed output
extern               Display external OSPF routes
instance             Name of OSPF instance
inter                Display interarea OSPF routes
intra                Display intraarea OSPF routes
|                   Pipe through a command
```

The output fields of the **show ospf route** command are the following:

- **Prefix:** Displays the destination of the route.
- **Route/Path Type:** Displays how the route was learned:
 - ABR: Route to area border router;
 - ASBR: Route to AS border router;
 - Ext: External router;
 - Inter: Interarea route;
 - Intra: Intra-area route; or
 - Network: Network router.
- **Metric:** Displays the route's metric value.
- **Next hop i/f:** Displays the interface through which the route's next hop is reachable.
- **Next hop addr:** Displays the address of the next hop.
- **area (detail output only):** Displays the area ID of the route.

- options (detail output only): Displays the option bits from the LSA.
- origin (detail output only): Displays the router from which the route was learned.

Displaying Entries in the OSPF LSDB

OSPF database, Area 0.0.0.0								
Type	ID	Adv	Rtr	Seq	Age	Opt	Cksum	Len
Router	*192.168.1.2	192.168.1.2		0x8000000c	1387	0x22	0x84ae	60
Router	192.168.1.3	192.168.1.3		0x80000023	1249	0x22	0x545e	60
Network	172.26.2.2	192.168.1.3		0x80000005	2049	0x22	0x43e3	32
Network	172.26.3.2	192.168.1.3		0x80000005	2449	0x22	0x38ed	32
Summary	*172.26.1.0	192.168.1.2		0x80000007	2541	0x22	0x4db7	28
Summary	172.26.4.0	192.168.1.3		0x80000025	2249	0x22	0xe9f8	28
Summary	*192.168.1.1	192.168.1.2		0x80000006	1618	0x22	0xa3bb	28
Summary	192.168.1.4	192.168.1.3		0x8000001a	1649	0x22	0x57ef	28
ASBRSum	*192.168.1.1	192.168.1.2		0x80000007	2310	0x22	0x93c9	28
OSPF database, Area 0.0.0.1								
Type	ID	Adv	Rtr	Seq	Age	Opt	Cksum	Len
Router	192.168.1.1	192.168.1.1		0x80000007	56	0x22	0x82c3	48
...								
OSPF AS SCOPE link state database								
Type	ID	Adv	Rtr	Seq	Age	Opt	Cksum	Len
Extern	172.18.1.0	192.168.1.1		0x80000005	96	0x22	0x374c	36

ABRs maintain a separate database for each OSPF area to which they are attached.

The **show ospf database** command displays the OSPF database. The display is organized by OSPF area and the LSA types within each area. As noted in the inset, ABRs have a separate database for each area in which they participate. The options for the **show ospf database** command are the following:

- brief (optional): Displays a brief listing of all entries in the OSPF link-state database; this setting is the default.
- detail (optional): Displays detailed information about the entries in the OSPF link-state database.
- extensive (optional): Displays extremely detailed information about the entries in the OSPF link-state database.

The asterisk (*) indicates that this router originated this entry. The sequence numbers are used to determine if an LSA is new; they start with a value of 0x80000001. According to the OSPF specification, an LSA's originator should refresh all valid LSAs every 30 minutes, or 1800 seconds, to prevent LSAs from aging out. The Junos OS implementation increases the refresh rate to every 50 minutes to help minimize the consumption of network bandwidth and CPU cycles. Stale LSAs are LSAs with an age greater than the implementation's refresh rate. Any LSA older than 3600 seconds are purged automatically. The age column displays how old the current LSA is and counts from 1–3600, so the remaining lifetime of an LSA can be calculated by subtracting the value in the age column from 3600.

In addition to the **brief**, **detail**, and **extensive** options, the **show ospf database** command also supports various LSA filter options that allow you to filter the output. The following capture shows the various LSA filter options you can use:

```
user@R2> show ospf database ?
Possible completions:
  <[Enter]>          Execute this command
  advertising-router   Router ID of advertising router
  area                 OSPF area ID
  asbrsummary          Summary AS boundary router link-state advertisements
  brief                Display brief output (default)
  detail               Display detailed output
  extensive            Display extensive output
  external              External link-state advertisements
  instance             Name of OSPF instance
  link-local            Link local link-state advertisements
  lsa-id               Link-state advertisement ID
```

netsummary	Summary network link-state advertisements
network	Network link-state advertisements
nssa	Not-so-stubby area link-state advertisements
opaque-area	Opaque area-scope link-state advertisements
router	Router link-state advertisements
summary	Display summary output
	Pipe through a command

As mentioned previously, the **show ospf database extensive** command displays a much more detailed view of the OSPF link-state database. The following list shows the additional output fields when the **extensive** option is used:

- **bits**: Displays the flags describing the router that generated the LSP.
- **link count**: Displays the number of links in the advertisement.
- Each link contains the following output fields:
 - **id**: Displays the ID of a router or subnet on the link.
 - **data**: For stub networks, displays the subnet mask; otherwise, it displays the IP address of the router that generated the LSP.
 - **type**: Displays the type of link; it can be PointToPoint, Transit, Stub, or Virtual.
 - **TOS count**: Displays the number of type-of-service (ToS) entries in the advertisement.
 - **TOS 0 metric**: Displays the metric for ToS 0.
- Each ToS entry contains the following output fields:
 - **TOS**: Displays the ToS value.
 - **metric**: Displays the metric for the ToS.
 - **Aging timer** (extensive output only): Displays how long until the LSA expires (displayed as hrs:min:sec).
 - **Installed** (extensive output only): Displays how long ago the route was installed.
 - **expires** (extensive output only): Displays how long until the route expires (displayed in hrs:min:sec).
 - **Ours** (extensive output only): Indicates that this advertisement is local.

Type Content in LSDB

Type	Link ID	Link Data
1. Point-to-point	Neighbor's RID	Originating router's interface IP address
2. Link-to-transit network	DR's Interface IP address	Originating router's interface IP address
3. Link-to-stub network	IP network number	Network's subnet mask or host address (/32)
4. Virtual link	Neighbor's RID	MIB-II index of originating router's interface

If needed, you can use the **clear ospf database** command to clear the OSPF database. You can also include the **purge** option to force the local router to set all LSAs in its database to max-age. These LSAs are then re-flooded according to the OSPF specification, which states that a router must flood any LSA that it has aged to max-age, regardless of whether that LSA was generated by the local router. All routers receive the newly flooded LSAs, which are now set to max-age, because of the reliable flooding mechanisms used by the OSPF protocol; the router that originated a given LSA is compelled to refresh that LSA when it receives an updated copy that indicates the LSA has reached max-age.

Albeit somewhat disruptive, this procedure tends to eliminate stale or bogus database entries without having to wait for the normal aging out process, which can take as long as 3600 seconds (one hour).

Displaying OSPF SPF-Related Information

The **show ospf log** command displays the entries in the OSPF log for SPF calculations. Note that the SPF algorithm performs multiple calculations on different portions of the link-state database. Use this command to verify that you have a stable OSPF routing domain. Multiple recalculations in a short period of time indicate potential instability. The output fields of this command are the following:

- When: Displays the time when the SPF calculation was made.
- Type: Displays the type of calculation, which can be Cleanup, External, Interarea, NSSA, Redist, SPF, Stub, Total, or Virtuallink.
- Elapsed: Displays, in seconds, the how much time has passed since the calculation was made.

```
user@R2> show ospf log
Last instance of each event type
When          Type          Elapsed
04:28:24      SPF           0.0000074
04:28:24      Stub          0.0000030
04:28:24      Interarea    0.0000042
04:28:24      External      0.0000016
04:28:24      NSSA          0.0000003
04:28:24      Cleanup        0.0000049

Maximum length of each event type
When          Type          Elapsed
20:09:11      SPF           0.000110
...
Last 100 events
When          Type          Elapsed
16:38:21      NSSA          0.0000003
...
```

Displaying OSPF Statistics

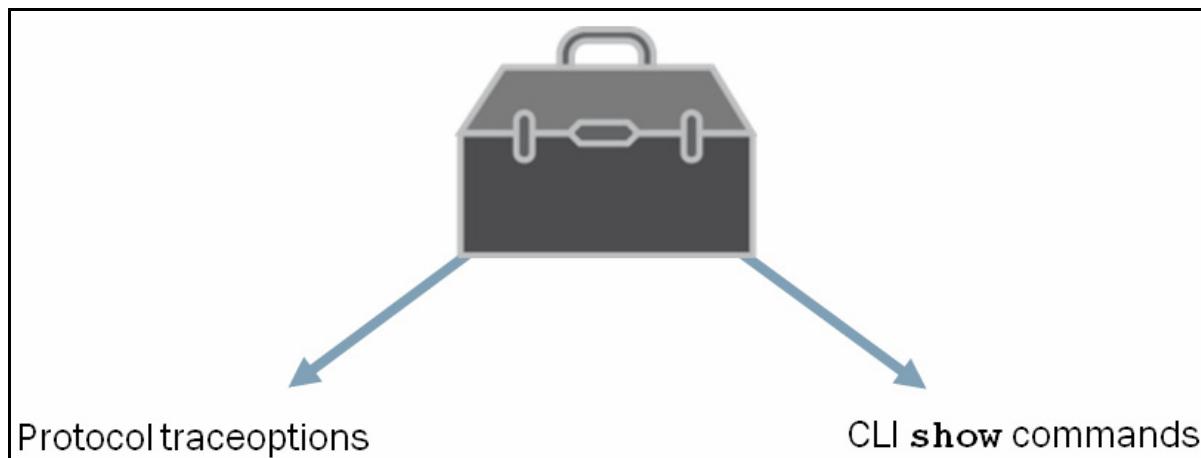
```
user@R2> show ospf statistics
```

Packet type	Total		Last 5 seconds	
	Sent	Received	Sent	Received
Hello	52	17	0	0
DbD	9	7	0	0
LSReq	2	2	0	0
LSUpdate	46	45	0	0
LSAck	37	33	0	0
DBDs retransmitted	:		0, last 5 seconds :	0
LSAs flooded	:		40, last 5 seconds :	0
LSAs flooded high-prio	:		10, last 5 seconds :	0
LSAs retransmitted	:		0, last 5 seconds :	0
LSAs transmitted to nbr:			8, last 5 seconds :	0
LSAs requested	:		2, last 5 seconds :	0
LSAs acknowledged	:		39, last 5 seconds :	0
...				

The **show ospf statistics** command displays details of the number and type of OSPF packets sent and received by the router. The output fields of this command are the following:

- Packet type: Displays the type of OSPF packet.
- Total Sent/Received: Displays the total number of packets sent and received.
- Last 5 seconds Sent/Received: Displays the total number of packets sent and received in the last 5 seconds.
- LSAs retransmitted: Displays the total number of link-state advertisements transmitted and number retransmitted in the last 5 seconds.
- Receive errors: Displays the number and type of receive errors.

OSPF Troubleshooting Tools



In the last section, we highlighted a number of helpful CLI **show** commands that you can use to monitor and troubleshoot OSPF. On subsequent pages, we cover traceoptions and some additional CLI **show** commands that can help with troubleshooting efforts.

Troubleshooting Common Adjacency Problems

Problem	Checklist
No neighbor detected	<ul style="list-style-type: none"> ▪ Check physical and data link layer connectivity ▪ Check for mismatched IP subnet/mask, area number, area type, authentication, hello/dead interval, or network type
Stuck in ExStart state	<ul style="list-style-type: none"> ▪ Check MTU settings to ensure that they match
Stuck in 2-way state	<ul style="list-style-type: none"> ▪ Normal for DR-Other neighbor

Use the details outlined in the table when troubleshooting OSPF adjacencies.

OSPF Tracing

```
[edit protocols]
user@R1# show
ospf {
    traceoptions {
        file trace-ospf;
        flag error detail;
        flag event detail;
    }
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface lo0.0;
    }
}
```

User-defined file-name and flag options. Include the **detail** option to generate additional details in the associated log file.

To perform debugging functions on the OSPF routing process, use the Junos OS **traceoptions** function. The trace output (debug information) is directed to the named log file, which is stored in the `/var/log` directory. You can view the log file using the **show log** or **monitor start** operational mode commands. In addition to specifying the trace file, you also must tell the router what information you want to trace. You can accomplish this goal by specifying one or more **flag** keywords.

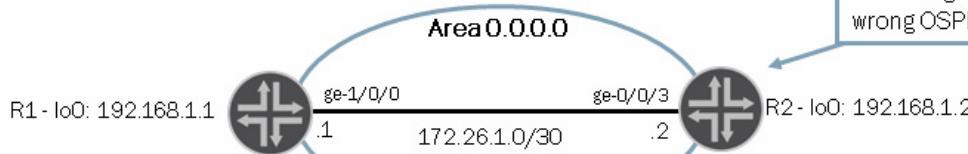
Although you can direct tracing only to a single file, you can trace many options by using the **flag** keyword multiple times. In addition, you can add granularity by using the **detail**, **receive**, and **send** flag modifiers.

You can include the following tracing flags for OSPF:

```
[edit protocols ospf]
user@R1# set traceoptions flag ?
Possible completions:
  all                      Trace everything
  database-description      Trace database description packets
  error                     Trace errored packets
  event                     Trace OSPF state machine events
  flooding                  Trace LSA flooding
  general                   Trace general events
  graceful-restart          Trace graceful restart
  hello                     Trace hello packets
  ldp-synchronization       Trace synchronization between OSPF and LDP
  lsa-ack                   Trace LSA acknowledgment packets
  lsa-analysis              Trace LSA analysis
  lsa-request                Trace LSA request packets
  lsa-update                 Trace LSA update packets
  normal                    Trace normal events
  nsr-synchronization        Trace NSR synchronization events
  on-demand                  Trace demand circuit extensions
  packet-dump                Dump the contents of selected packet types
  packets                   Trace all OSPF packets
  policy                     Trace policy processing
  route                     Trace routing information
  spf                       Trace SPF calculations
  state                     Trace state transitions
  task                      Trace routing protocol task processing
  timer                     Trace routing protocol timer processing
```

Displaying the Log File Contents: Part 1

```
user@R1> show log trace-ospf
Oct 13 09:05:51.748087 OSPF [redacted] packet ignored: area mismatch (0.0.0.1) from
172.26.1.2 on intf ge-1/0/0.0 area 0.0.0.0
Oct 13 09:05:51.748208 OSPF rcvd Hello 172.26.1.2 -> 224.0.0.5 (ge-1/0/0.0
IFL 73 area 0.0.0.0)
Oct 13 09:05:51.748237 Version 2, length 44, ID 192.168.1.1, area 0.0.0.1
Oct 13 09:05:51.748250 checksum 0x8c5c, authtype 0
Oct 13 09:05:51.748264 mask 255.255.255.252, hello_ivl 10, opts 0x2, prio
128
Oct 13 09:05:51.748281 dead_ivl 40, DR 172.26.1.2, BDR 0.0.0.0
```



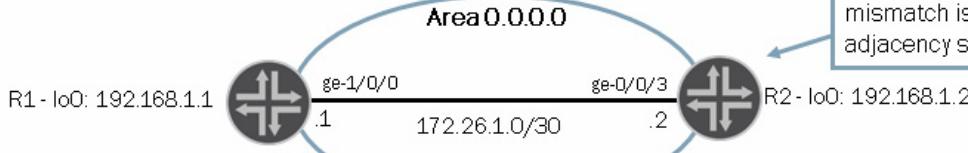
According to the log file, R2 has the wrong OSPF area configured.

You can use the **show log *file-name*** command to display the contents of the traceoptions log file. Alternatively, you can use the **monitor start** command to view the same information. In many cases, such as the one illustrated in the graphic, OSPF adjacency issues become quite obvious when you examine the logs.

Displaying the Log File Contents: Part 2

Address	Interface	State	ID	Pri	Dead
172.26.1.1	ge-0/0/3.0	ExStart	192.168.1.1	128	31

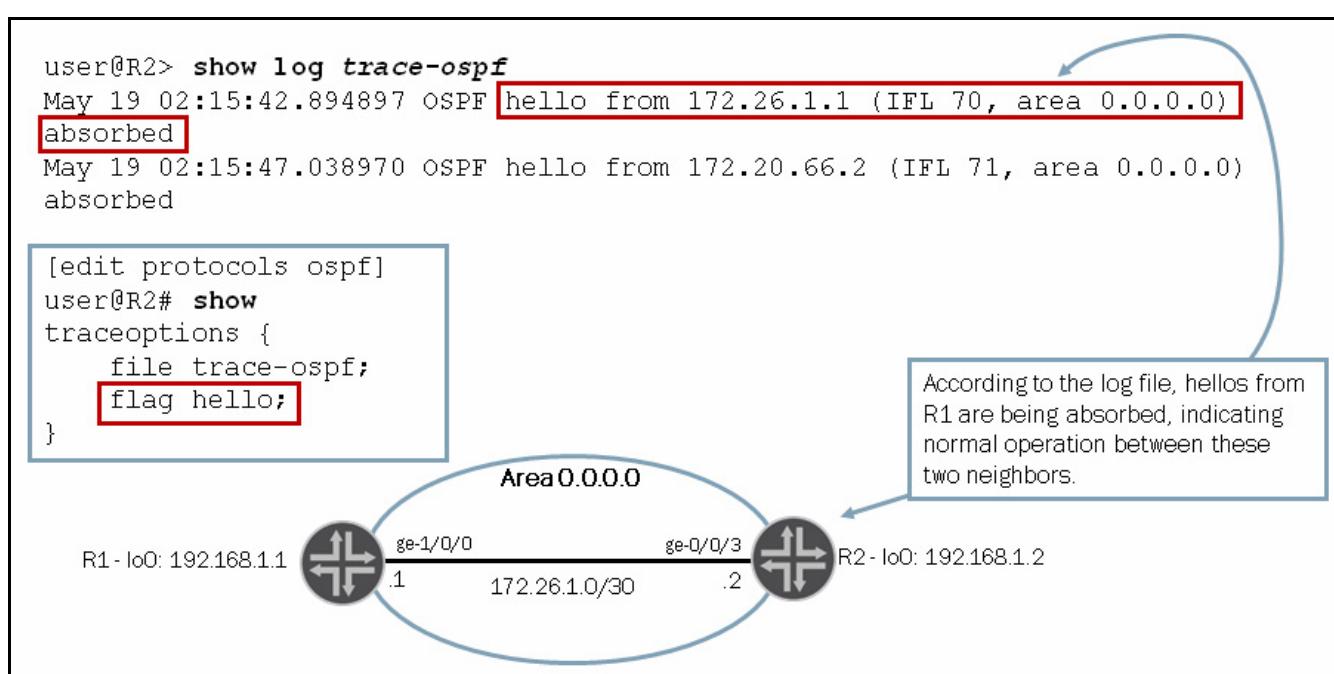
```
user@R2> show log trace-ospf
May 19 02:17:48.956726 OSPF [redacted] packet ignored: MTU mismatch from 172.26.1.1 on
intf ge-0/0/3.0 area 0.0.0.0
May 19 02:17:53.763661 OSPF packet ignored: MTU mismatch from 172.26.1.1 on
intf ge-0/0/3.0 area 0.0.0.0
```



According to the log file, the MTU values on the Gigabit Ethernet interfaces do not match. This mismatch is a common cause of an adjacency stuck in the ExStart state.

At times, an OSPF adjacency will not transition to the Full state and becomes stuck in the ExStart or ExChange state. The most common reason for this issue is a mismatch in the MTU settings and is easily discernible from a traceoptions log file.

Displaying the Log File Contents: Part 3



OSPF hellos will appear as “absorbed” in the log file when everything is operating normally within an OSPF adjacency.

Viewing OSPF Error Counters

- Use the **show ospf statistics** command to view OSPF errors

```

user@R1> show ospf statistics

...
Receive errors:
 410 area mismatches
 17 mtu mismatches
 81 Hellos received with our router ID

```

- Use **clear ospf statistics** to refresh counters

```
user@R1> clear ospf statistics
```

Although we highlighted the **show ospf statistics** command earlier, we did not illustrate the lower portion of that command’s output. The output in the inset shows the Receive errors counter with some error types and their respective counters. You can refresh the counters using the **clear ospf statistics** command.

Review Questions

1. What is the purpose of OSPF LSAs?
2. What benefits can exist by segmenting a large single-area OSPF environment into multiple areas?
3. What is the difference between an ABR and an ASBR?
4. List some common OSPF area types and their functional considerations.

Answers

1.

OSPF routers use LSAs to share information about their attached networks with other neighboring OSPF routers.

2.

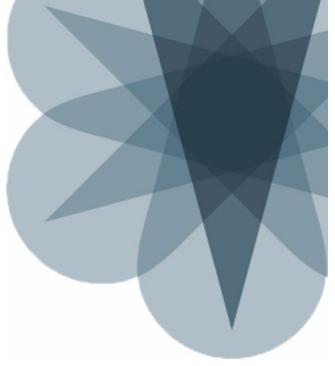
You can restrict the size of an OSPF router's LSDB by incorporating multiple areas. In some types of OSPF areas, such as stub, totally stubby, and NSSA areas, only certain types of LSAs are flooded.

3.

ABRs connect nonbackbone areas to the backbone area and ASBRs inject external prefixes into OSPF using redistribution policy. Note that an OSPF router can be an ABR and an ASBR simultaneously.

4.

There are several types of OSPF areas such as stub, totally stubby, and NSSA areas. Stub areas do not carry external routes and cannot contain ASBRs. Totally stubby areas are stub areas that do not receive summary LSAs from the backbone but rather receive a single LSA that represents a default route. NSSA areas allow external routes to be advertised from the area but not received from another area. It is also worth mentioning the backbone area (Area 0.0.0.0) to which all other areas connect.



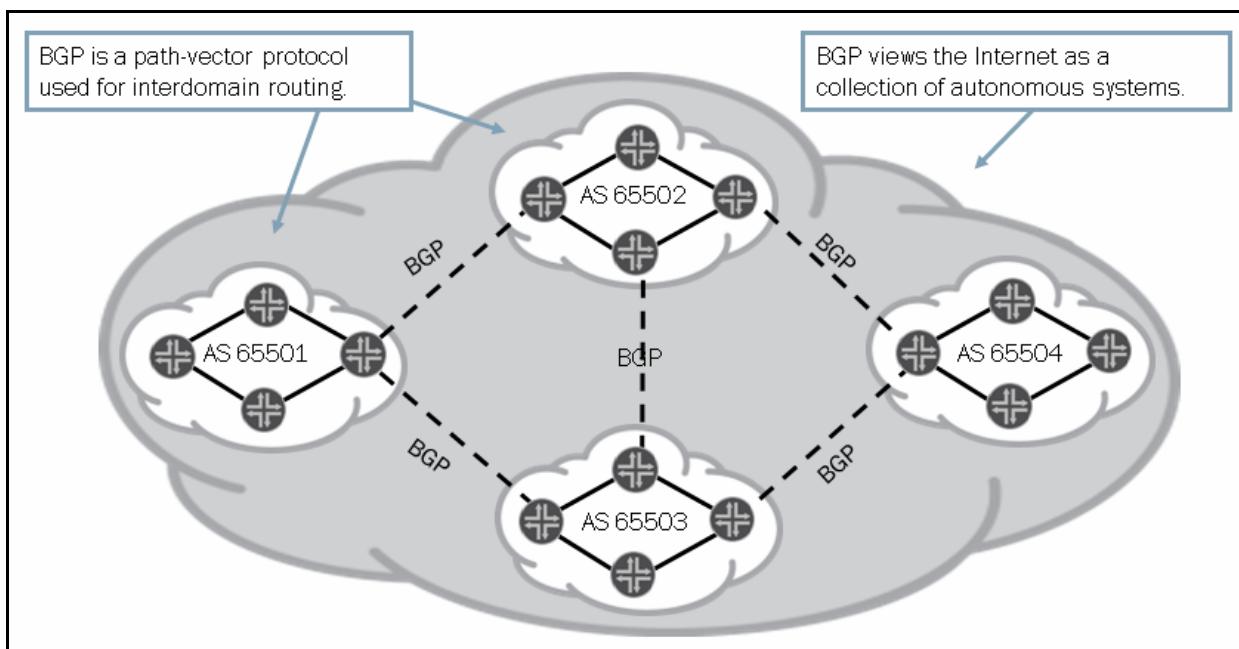
JNCIS-SP Study Guide—Part 1

Chapter 4: Border Gateway Protocol

This Chapter Discusses:

- Border Gateway Protocol (BGP) operations;
- Common BGP attributes;
- The BGP route selection algorithm;
- BGP peering options and the default route advertisement rules; and
- Configuration and monitoring of BGP.

What Is BGP?



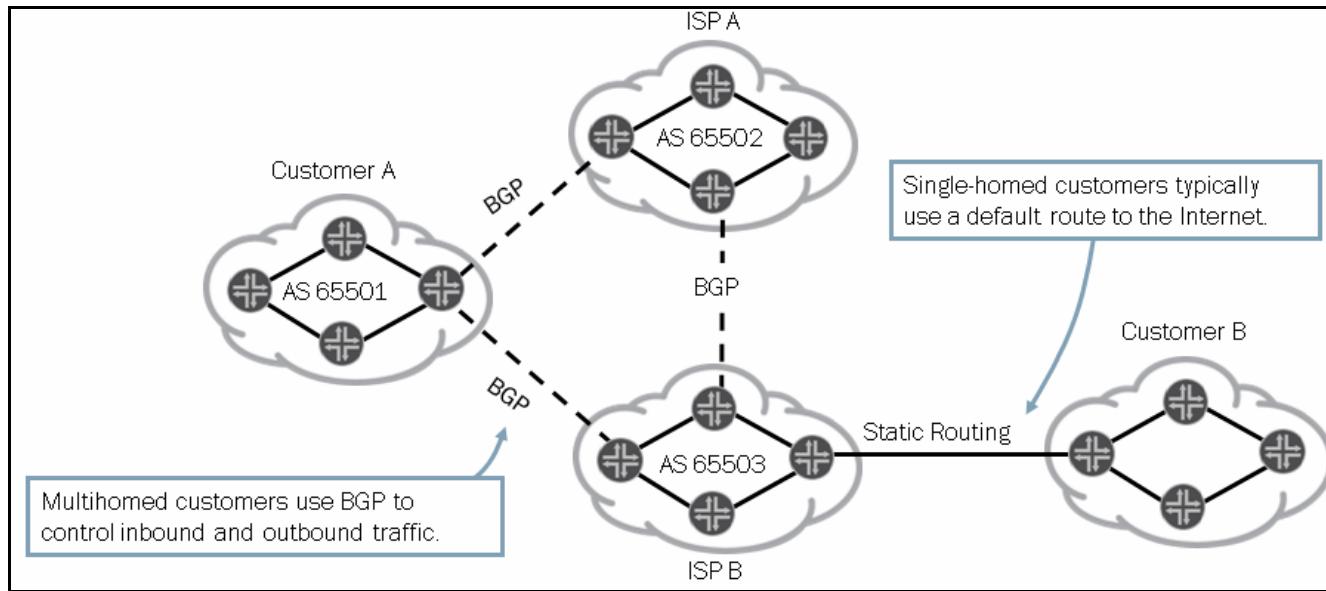
The Border Gateway Protocol (BGP) is a routing protocol between autonomous systems (ASes) and is sometimes referred to as a *path-vector routing protocol* because it uses an AS path, used as a vector, to prevent inter-domain routing loops. The term *path vector*, in relation to BGP, means that BGP routing information includes a series of AS numbers, indicating the path that a route takes through the network. Although BGP is primarily used for inter-AS routing, BGP is also used in large networks for MPLS-based VPNs and is used to separate large OSPF domains. BGP is much more scalable and offers a greater amount of control through policy than an IGP.

BGP exchanges routing information among ASes. An AS is a set of routers that operate under the same administration. BGP routing information includes the complete route to each destination. BGP uses the routing information to maintain an information base of network layer reachability information (NLRI), which it exchanges with other BGP systems.

BGP is a classless routing protocol, that supports prefix routing, regardless of the class definitions of IPv4 addresses. BGP routers exchange routing information between peers. The peers must be connected directly for inter-AS BGP routing (unless certain configuration changes are done). The peers depend on established TCP connections, which we address later in this chapter.

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

When Should I Use BGP?

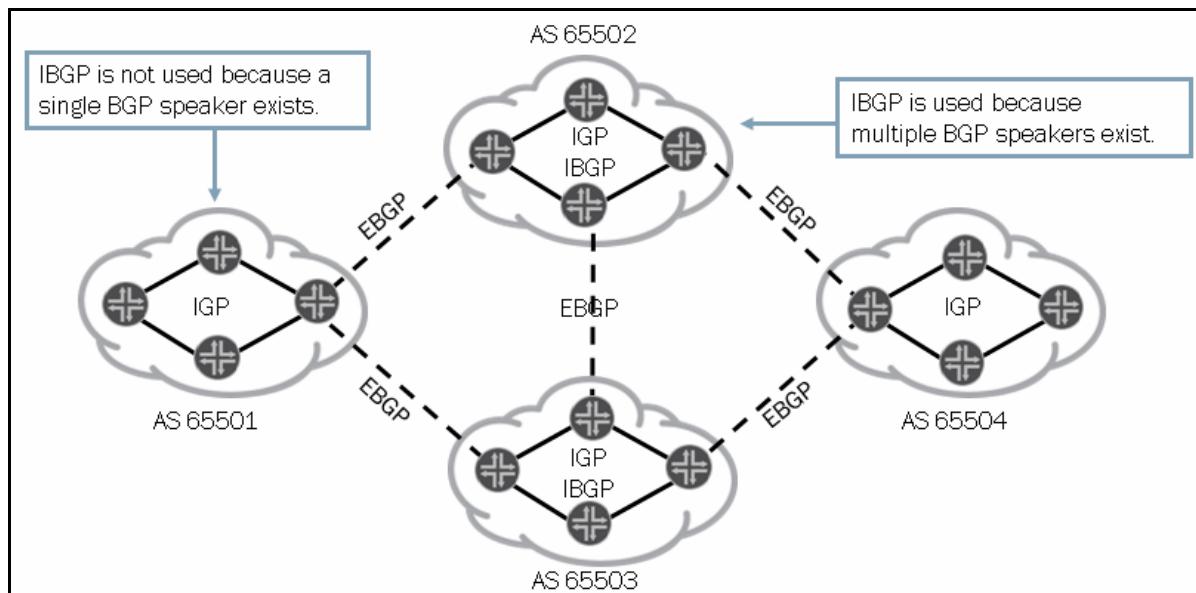


Networks with a single upstream connection receive little benefit from running a dynamic routing protocol with their Internet service provider (ISP). These customers typically use a static default route to send all external traffic toward the Internet. Their provider also typically uses a static route to direct traffic destined for the customer's addresses to the customer. Normally, a single-homed network uses addresses assigned by the provider from the provider's aggregate. Because these addresses are assigned to the provider and can only be used by the customer while they are a customer of the provider, they are known as *nonportable* addresses. Using these addresses allows the provider to announce a single aggregate route for many customer networks, reducing global routing table growth. Currently, the Internet routing table contains hundreds of thousands of routes, which highlights the need for a scalable and robust protocol such as BGP.

BGP is normally used when a network has multiple upstream connections, either to a single ISP or to multiple ISPs. BGP's policy controls provide the ability to optimize inbound and outbound traffic flows based on a network's technical and business constraints. Although BGP can detect and route around failures in redundant environments, BGP sessions within the same AS do not typically react as quickly as an IGP, and they often rely on the IGP used in the AS to remain operational when failures occur.

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

EBGP and IBGP Peers



BGP supports two different types of exchanges of routing information. Exchanges between ASes are known as *external BGP* or *EBGP* sessions and handle inter-AS routing. Exchanges within an AS are known as *internal BGP* or *IBGP* sessions, and handle intra-AS routing.

An EBGP peer connection is between a device in one AS and another device in a different AS. The connection between the two ASes consists of a physical connection and a BGP connection. The physical connection is a shared Data Link layer subnetwork between the two ASes. On this shared subnetwork, each AS has at least one border gateway belonging to that AS. The BGP connection exists between BGP speakers in each of the ASes. This session can communicate destinations that can be reached through the advertising AS. By default the EBGP connection is established between immediately connected devices located in two different ASes because the time-to-live (TTL) value of the EBGP packets is equal to 1. If an EBGP peer is more than one hop away, a multihop BGP session must be established.

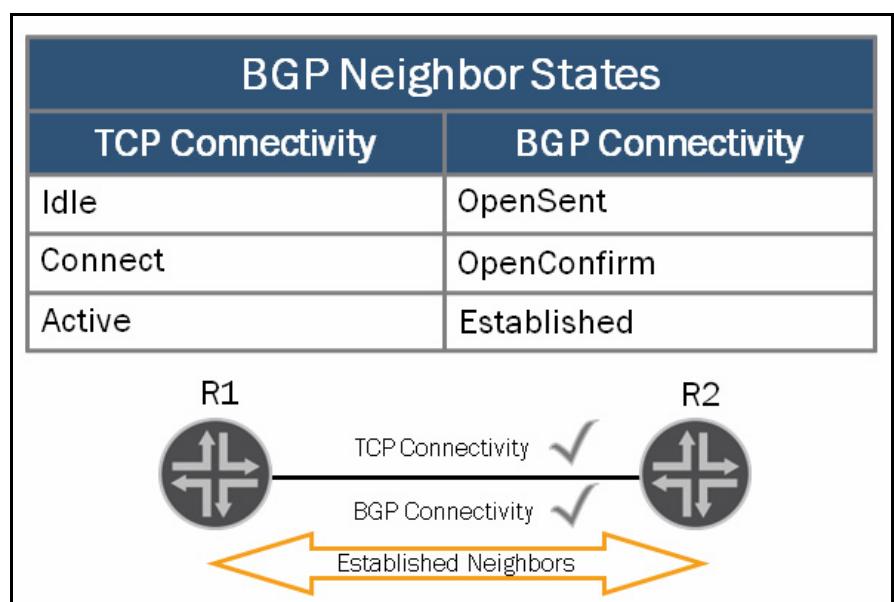
An IBGP connection is typically established between loopback interfaces of the routers not immediately connected (of course, everything depends on the AS's topology). BGP uses the loopback interfaces for stability reasons—these interfaces are always *alive*, unless the router itself *dies*. Because the IBGP connection typically exists between remotely connected routers, there is a requirement for an IGP within the AS. BGP's TCP session is established using regular routing tables.

BGP Peering Sessions

Unlike other dynamic protocols, BGP requires that you manually define the neighbors with which you want the local device to peer.

Because BGP peers must be manually defined, no automatic neighbor discovery exists as with other protocols.

BGP uses TCP as its transport protocol (port 179). TCP provides a full-duplex, connection-oriented, reliable, byte-stream service to BGP. BGP considers a connection between two peers to be idle until a TCP connection is established between them. With the TCP connection established, the endpoints are assured of a reliable connection. The following list describes the various BGP neighbor states:



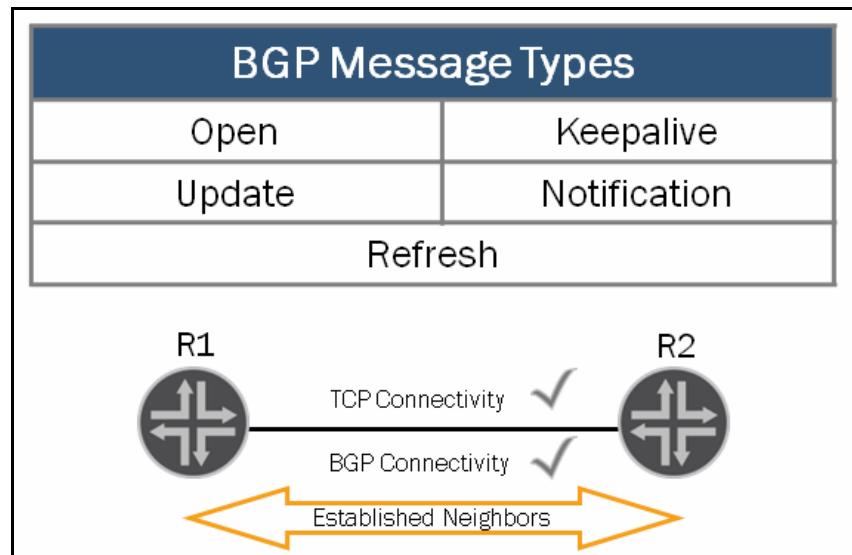
- Idle state:** The Idle state is the initial state when all incoming BGP connections are refused. A start event is required for the local system to initialize BGP resources and prepare for a transport connection with the other BGP peer.

- Connect state:** In the Connect state, BGP is waiting for the transport protocol connection to be completed. If the transport protocol connection succeeds, the local system sends an OPEN message and transitions to the OpenSent state. If the transport protocol connection fails, the local system restarts the ConnectRetryTimer, listens for a connection initiated by the remote BGP peer, and changes its state to Active.
- Active state:** In the Active state, BGP is trying to acquire a peer by initiating a transport protocol connection. If the transport protocol connection succeeds, the local system sends an OPEN message to its peer and transitions to the OpenSent state. If the local system's BGP state remains in the Active state, you should check physical connectivity as well as the configuration on both peers.
- OpenSent state:** In the OpenSent state, BGP waits for an OPEN message from its peer. When an OPEN message is received, it is checked and verified to ensure that no errors exist. If an error is detected, the system transitions back to the Idle state. If no errors are detected, BGP sends a Keepalive message.
- OpenConfirm state:** In the OpenConfirm state, BGP waits for a KEEPALIVE or NOTIFICATION message. If no KEEPALIVE message is received before the negotiated hold timer expires, the local system sends a NOTIFICATION message stating that the hold timer has expired and changes its state to Idle. Likewise, if the local system receives a NOTIFICATION message, it changes its state to Idle. If the local system receives a KEEPALIVE message, it changes its state to Established.
- Established state:** In the Established state, BGP can exchange UPDATE, NOTIFICATION, and KEEPALIVE messages with its peer. When the local system receives an UPDATE or KEEPALIVE message and when the negotiated hold timer value is nonzero, it restarts its hold timer. If the negotiated hold timer reaches zero, the local system sends out a KEEPALIVE message and restarts the hold timer.

BGP Message Types

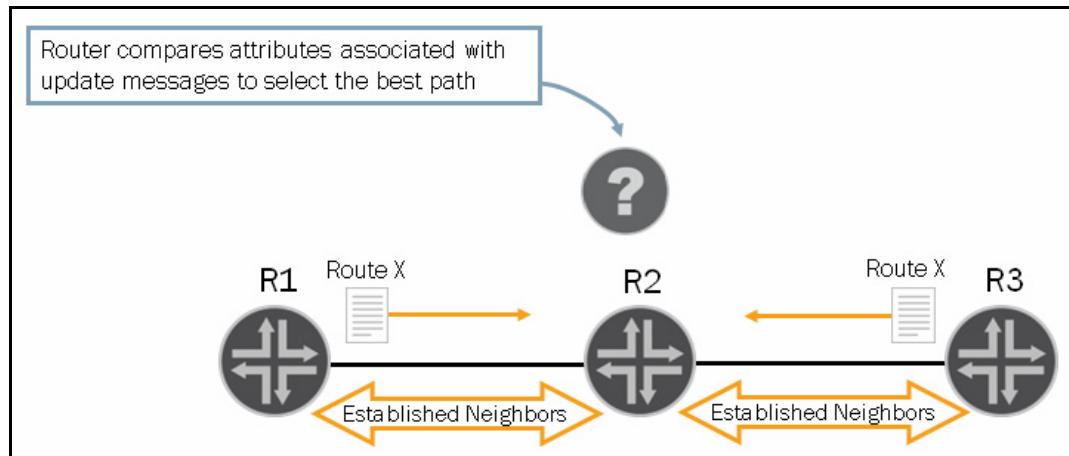
BGP processes a message only after the entire message is received. The maximum message size is 4096 octets; the smallest BGP message is a header without any data, or 19 octets. The following list details the BGP message types:

- Open message:** The open message is sent once the TCP three-way handshake is complete. The open message initiates the BGP session and contains details about the BGP neighbor and information about supported and negotiated options.
- Update message:** BGP uses update messages to transport routing information between BGP peers. Depending on the receiving device's routing policy, this routing information is either added to the routing table or ignored.
- Keepalive message:** BGP does not use keepalives at the Transport Layer. TCP fills this need. Instead, peers exchange keepalives as often as needed to ensure that the hold timer does not expire.
- Notification message:** BGP uses notification messages to signal when something is wrong with the BGP session. A notification is sent when an unsupported option is sent in an open message and when a peer fails to send an update or keepalive. When an error is detected, the BGP session is closed.
- Refresh:** Normally a BGP speaker cannot be made to readvertise routes that have already been sent and acknowledged (using TCP). The route refresh message supports soft clearing of BGP sessions by allowing a peer to readvertise routes that have already been sent. This soft clearing has some very specific uses when working with MPLS-based VPNs and adding new customer sites to existing customer VPN structures.



Each BGP message uses the same fixed size header, which is 19 bytes. BGP keepalive messages do not include any data portion following the header.

BGP Update Messages

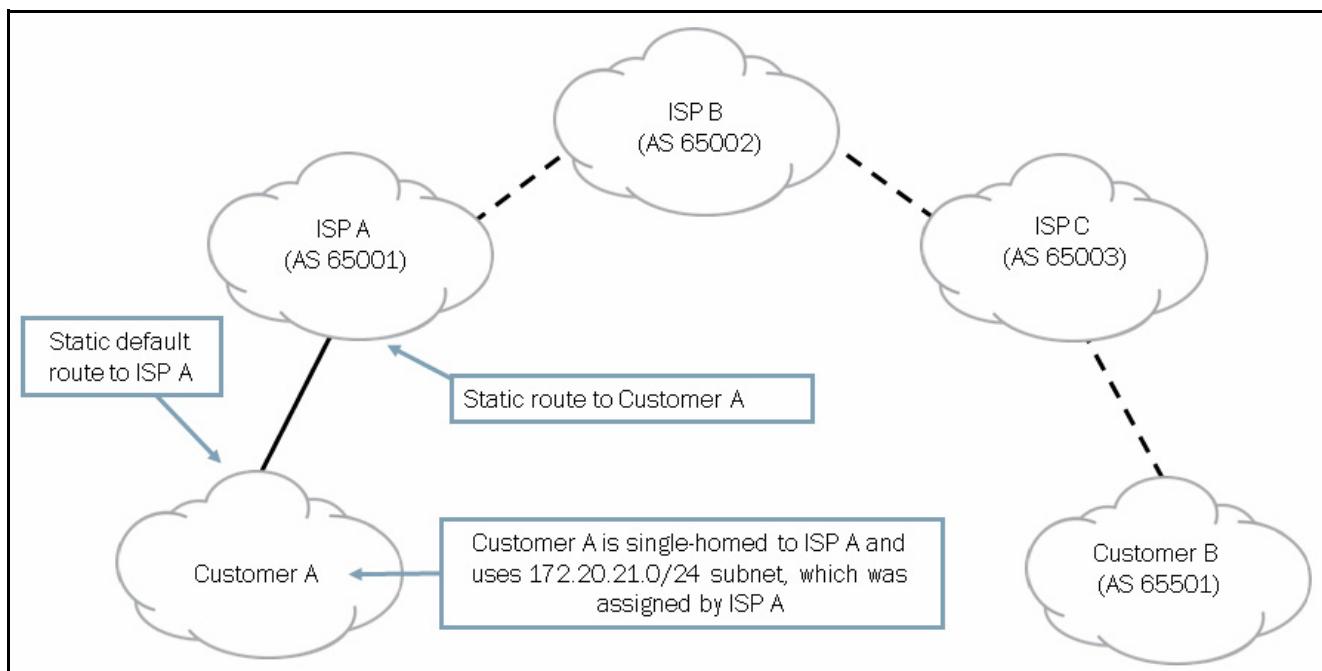


BGP update messages describe a single path and then list multiple prefixes that can be reached through this same path. BGP peers assume that this information is unchanged unless a subsequent update advertises a new path for a prefix or lists the prefix as unreachable. Updates can list any prefixes that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

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

BGP uses the information within the update messages, in particular the BGP attributes, to detect routing loops and determine the best path for a given destination prefix.

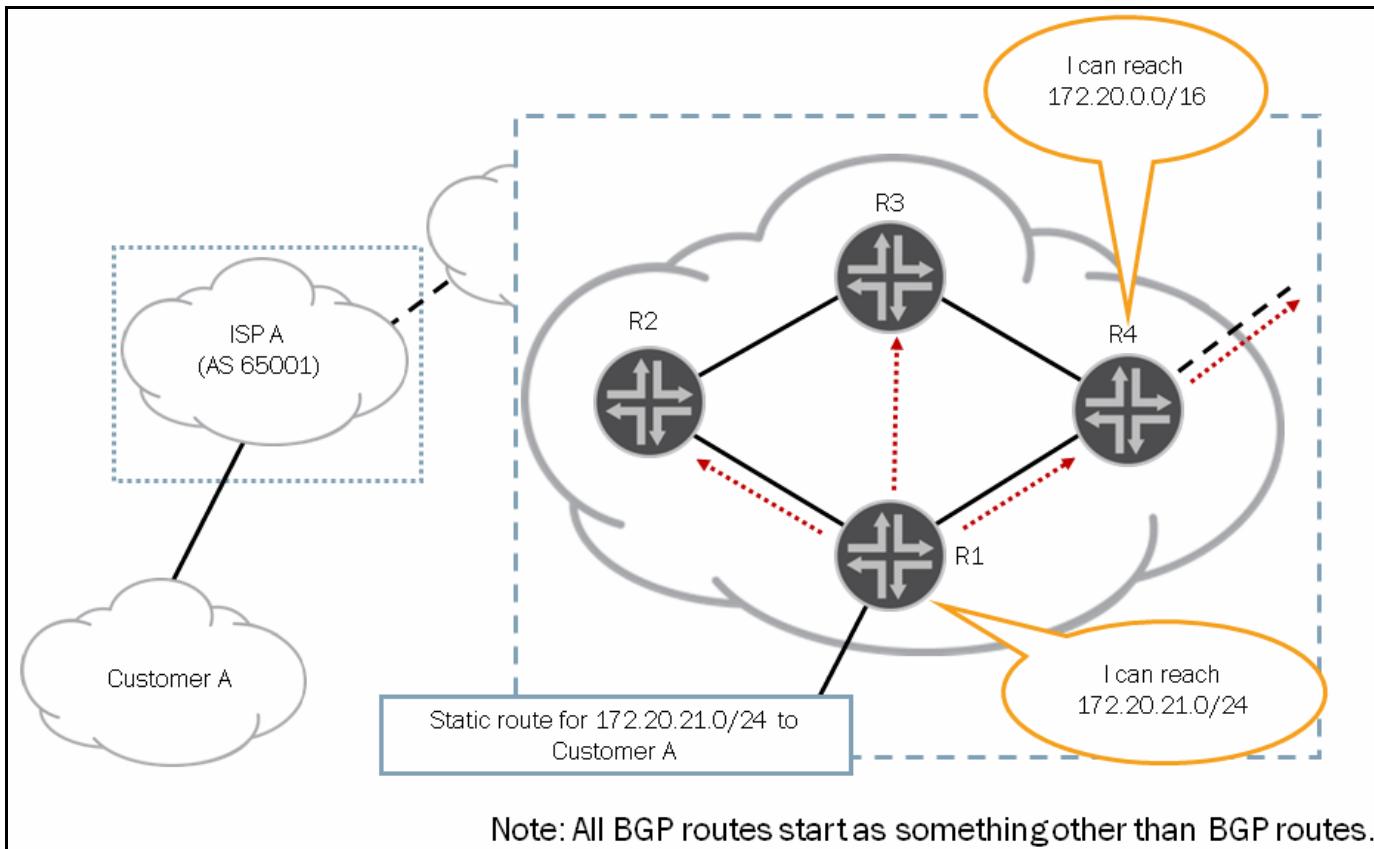
High-Level BGP Example



The example in the graphic explains the operation of BGP at a very high level. Consider the way traffic is routed to Customer A. Customer A has a single connection to ISP A. ISP A has assigned Customer A a prefix (172.20.21.0/24) from its aggregate address range (172.20.0.0/16).

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

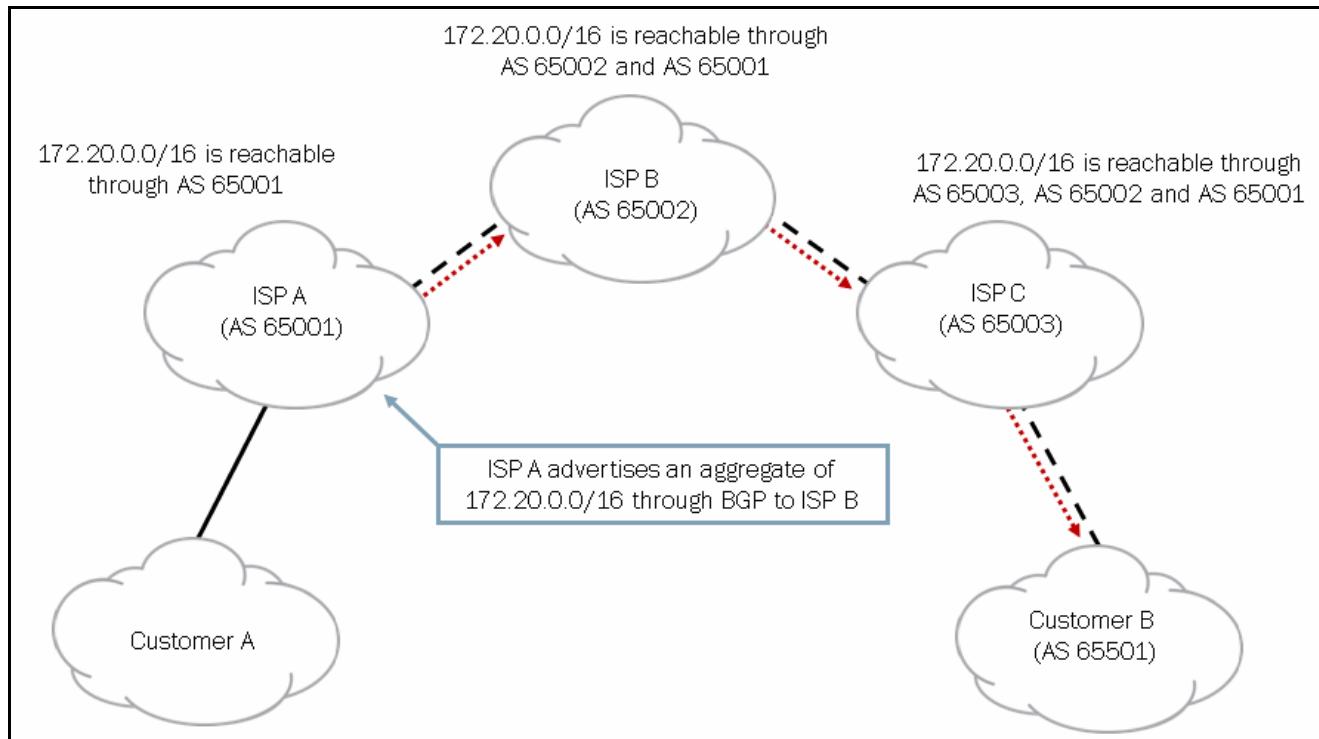
ISP A's Network



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

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

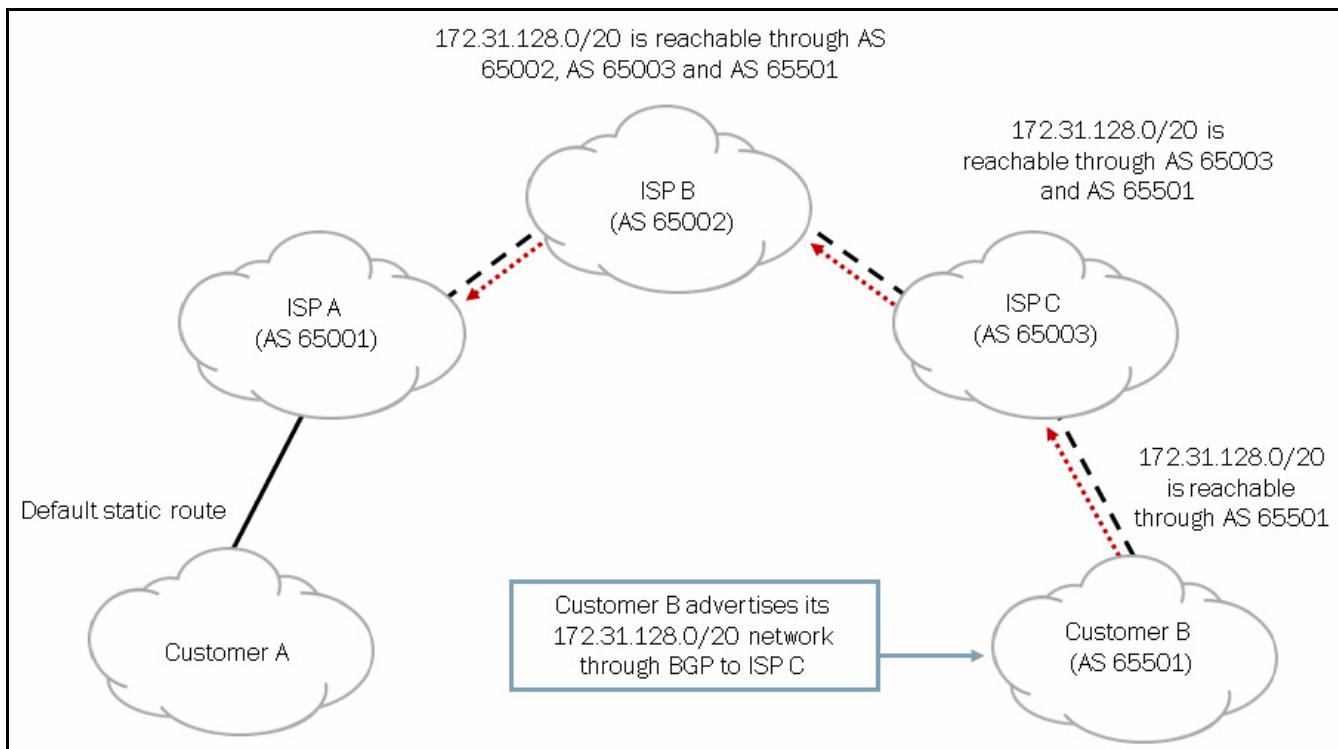
ISP A Advertises Its Aggregate



ISP A advertises its aggregate address range of 172.20.0.0/16 through BGP along with some information about the path to reach that route. One of these *path attributes* is the AS path, which is a list of the autonomous systems through which the path to this aggregate passes. By examining the AS path, ISP B knows that the 172.20.0.0/16 network was originated within ISP A.

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

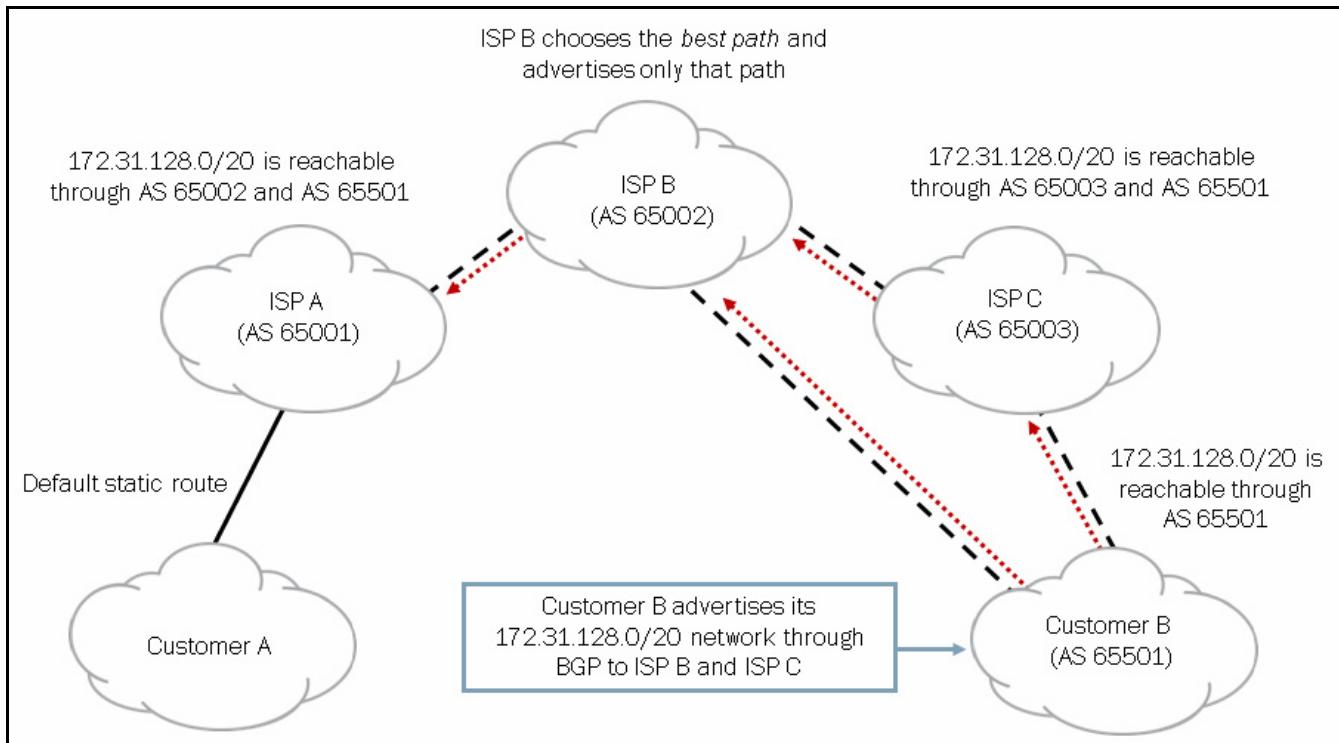
Customer B's Aggregate



Customer B is currently a single-homed network but is planning on adding a second connection to another ISP in the near future. Customer B advertises its portable /20 prefix to ISP C with an AS path, indicating that it was originated locally. ISP C sends the advertisement to ISP B, who sends it to ISP A, with each ISP updating the path attributes as it sends the route.

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

Customer B Becomes Multihomed



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

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

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

We cover the path selection process and many of the BGP attributes in greater detail later in this chapter.

BGP Attributes Table

The BGP attribute classes are described in RFC 1771 and include the well-known mandatory, well-known discretionary, optional transitive, and optional nontransitive classes.

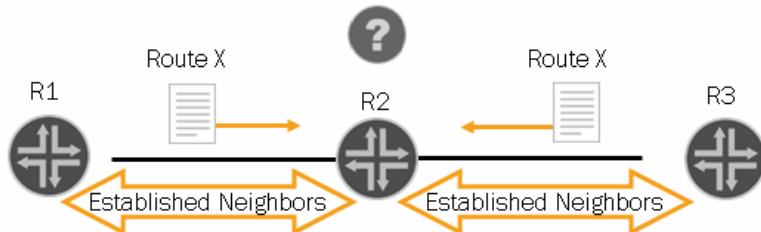
- **Well-known mandatory:** Must be supported by all BGP implementations and must be included in every BGP update.
- **Well-known discretionary:** Must be supported in all BGP implementations, but do not have to be included in every BGP update.
- **Optional transitive:** Not required to be supported by all BGP implementations but, if they are, they should be passed along, unchanged, to other BGP peers.
- **Optional nontransitive:** Not required to be supported by all BGP implementations. If an optional nontransitive attribute is not recognized, it is ignored and not passed to other peers.

BGP Attributes	
Name	Type
AS Path	Well-known mandatory
Local Preference	Well-known discretionary
MED	Optional nontransitive
Origin	Well-known mandatory
Next Hop	Well-known mandatory
Community	Optional transitive
Aggregator	Optional transitive
Atomic Aggregator	Well-known discretionary
Cluster List	Optional nontransitive
Originator ID	Optional nontransitive

The table lists most BGP attributes. We cover the more common attributes in greater detail on subsequent pages.

BGP Attributes

- Attributes are used to select the best path



- Some common examples include:

Common BGP Attributes

Next Hop	Local Preference	AS Path	Origin	MED	Community
----------	------------------	---------	--------	-----	-----------

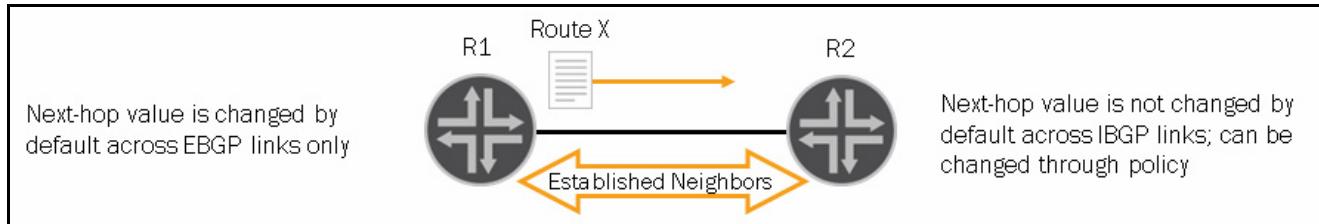
The primary purpose of BGP is *not* to find the shortest path to a given destination; rather, its purpose is to find the best path. Each AS determines the best path to a prefix by taking into account its own outbound routing preferences, the inbound routing preferences of the route's originator (as updated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

The Next-Hop Attribute

Common BGP Attributes						
Next Hop	Local Preference	AS Path	Origin	MED	Community	

The BGP next-hop attribute is an IP address of a BGP peer. It is used to verify connectivity of a remote BGP peer. A BGP peer can be an immediately connected host or a remote host. Recursive lookups into routing tables accomplish the peer connectivity.

The IP address specified in the next-hop field must be reachable by the local router before the route becomes active in the routing table.



By default, the router that originally sourced the route into BGP places its peer address into the attribute field. The next-hop value is then typically changed when the route is transmitted across EBGP links. IBGP peers do not alter the next-hop value between themselves. Policy controls can be put into place on any peering links to change the value of the next-hop field.

The BGP next-hop attribute is always present for all BGP routes. If the routes are active, this attribute always is passed across all BGP links with its corresponding route. BGP routes for which the next hop is not reachable are placed into the routing table as hidden routes. You can view these routes with the **show route hidden** CLI command.

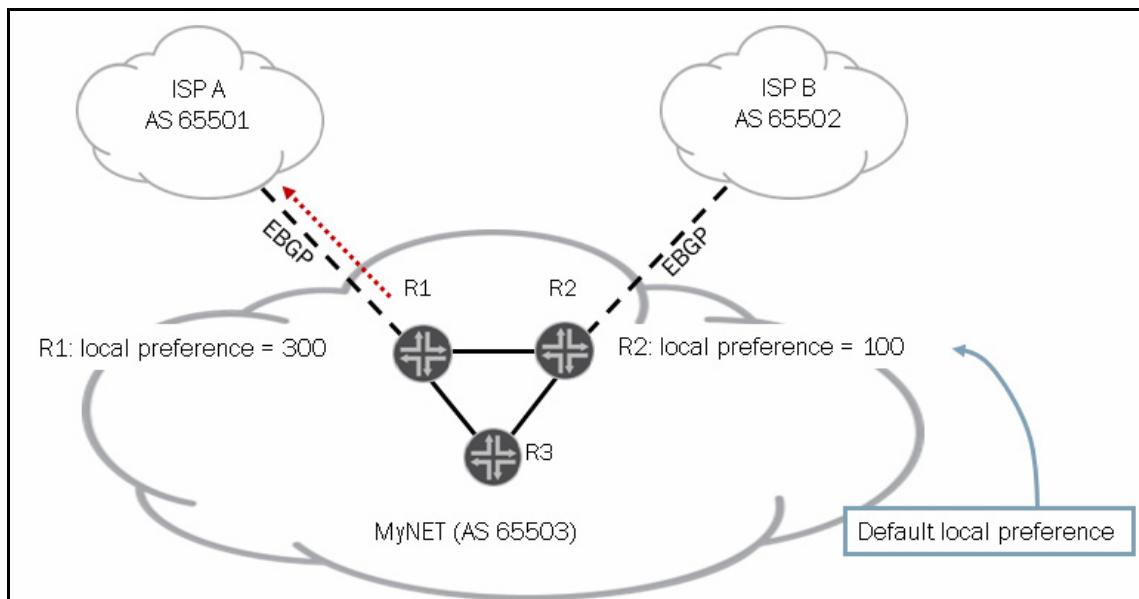
The Local-Preference Attribute

Common BGP Attributes						
Next Hop	Local Preference	AS Path	Origin	MED	Community	

You can use the local-preference attribute to direct all outbound traffic through a specific peer. Before sending the traffic to the internal peers, the designated peer sets the local-preference value on all routes received. Then all the peers use those routes in their RIB-local tables. The local-preference attribute is a numeric value; higher values indicate a better metric. The Junos operating system allows you to set the local-preference value using BGP configuration or through routing policy. If you set a local-preference value through both configuration and routing policy, the system uses the value assigned through routing policy.

BGP uses the local-preference attribute only within an AS. Local-preference values are not transmitted across EBGP links.

Local-Preference Example



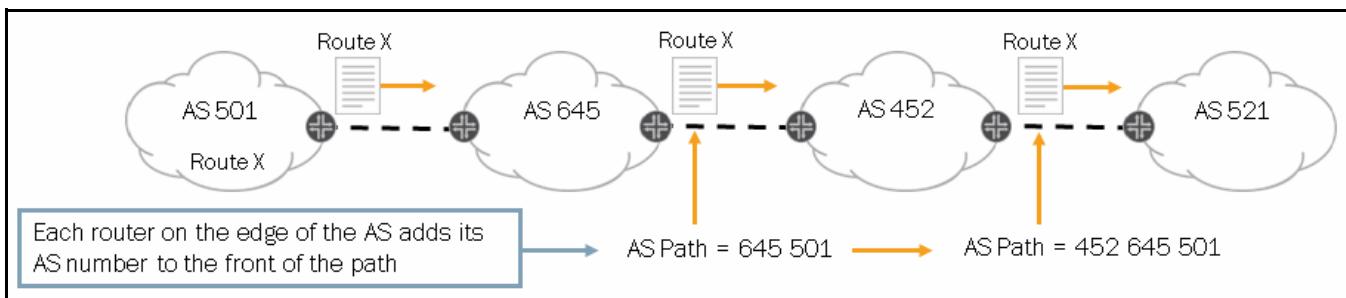
The graphic shows an example of how the local-preference attribute is used within an AS. The network administrators in MyNET have decided that the routers in MyNET should use ISP A whenever possible and ISP B as its secondary path. This decision might be made for several reasons, including cost of service and performance factors.

To use the ISP A path, the MyNET network administrators set the local-preference attribute for all routes received from ISP A to 300 while all routes received from ISP B use the default local-preference value of 100. We assume that ISP A and ISP B are advertising a similar set of routes. Because of the higher local-preference value associated with the routes received from ISP A, the MyNET routers choose the path through ISP A rather than the path through ISP B.

In contrast to other BGP attribute values, higher local-preference values are preferred.

The AS-Path Attribute

Common BGP Attributes						
Next Hop	Local Preference	AS Path	Origin	MED	Community	



The AS-path attribute describes the path of autonomous systems that the route has been through since it was sourced into BGP. When a BGP router receives routes in an update message, the first action is to examine the current AS path to see if the local AS number is in the path. If it is in the path, it indicates that the route has been through the AS already; accepting the route would cause a loop. Therefore, BGP drops the route. The Junos OS performs a verification to ensure the receiving router's AS number is not listed in the AS path. If the receiving router's AS number is listed in the AS path, the router does not advertise the route.

By default, the AS-path value is changed as a route transitions between autonomous systems. The AS-path value is null until the associated route is advertised out of the originating AS. As the route leaves an AS, the BGP router adds the local AS number to the front of the path before sending it to a peer. Using routing policy, you can prepend your ASN information to the AS-path attribute. By prepending your ASN information to the AS-path multiple times, you can affect the routing decision made by routers in other autonomous systems and discourage those routes from using that path because of the longer AS-path.

The AS-path attribute is mandatory; thus, it must always be present for all BGP routes.

The Origin Attribute

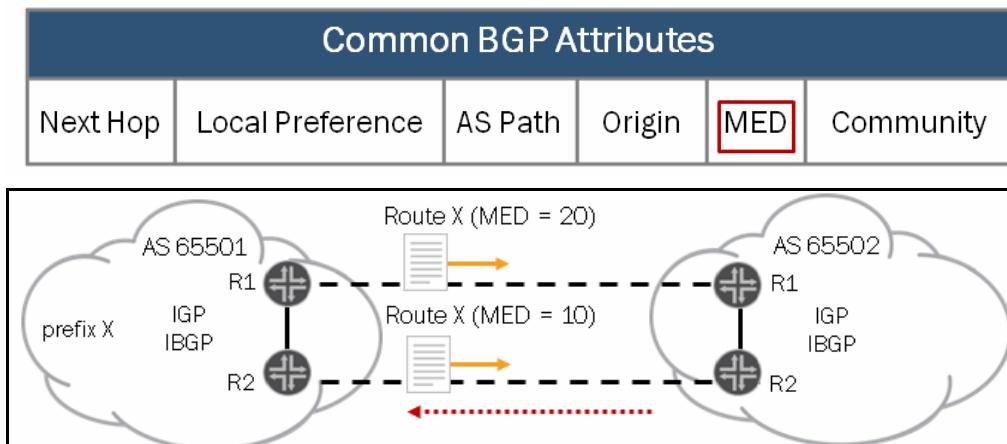
Common BGP Attributes					
Next Hop	Local Preference	AS Path	Origin	MED	Community

The BGP router that injects a route into the BGP process is responsible for placing the origin attribute into the route. The origin attribute describes where the original router received the route. The following are the possible choices:

- *IGP*: BGP assigns an IGP route a value of 0. Examples of IGP routes include OSPF, IS-IS, static, and aggregate.
- *EGP*: BGP assigns EGP routes a value of 1. EGP routes are from the original EGP protocol, which was the predecessor to BGP.
- *Incomplete*: BGP assigns unknown routes a value of 2. Unknown routes are those that did not come from an IGP or from EGP.

The origin attribute is a mandatory attribute; it is transmitted across all BGP links. By default, the Junos OS assigns all routes injected into BGP an origin value of *I* for IGP. You can alter this default value using routing policy.

The MED Attribute



By default, BGP uses the multiple exit discriminator (MED) value only when the BGP router's AS has two or more connections to the same upstream AS. You can determine the existence of multiple connections by examining the AS-path attribute to find that multiple routes from the same AS being advertised by multiple, different peers exist.

An AS uses the MED value in an attempt to influence data traffic headed back toward the AS. The local AS sets the MED value differently on separate peers headed toward the same remote AS. The remote AS picks routes based on the lowest MED value it finds. The remote AS then uses that local peer to route traffic back to the local AS.

BGP routes do not require the MED attribute. If it is missing, BGP assumes the MED value to be 0. To configure a MED metric, you can either use the **metric-out** statement at the BGP protocol, group or neighbor level or define and apply a routing policy that alters the MED value using **metric** as an action in the **then** statement.

The Community Attribute

Common BGP Attributes					
Next Hop	Local Preference	AS Path	Origin	MED	Community

A BGP community is an identifier that represents a group of destination prefixes that share a common property. Communities are used to tag specific routes that can be identified easily later for a variety of purposes. BGP includes community attribute information as a path attribute in BGP update messages.

```
[edit policy-options]
user@R1# show
policy-statement ibgp-export {
    from neighbor 172.25.125.2;
    then {
        community set customer-routes;
    }
}
community customer-routes members 64700:133;
```

You define BGP communities under the [edit policy-options] hierarchy and typically reference those communities in a routing policy. The inset illustrates the typical format for user-defined BGP communities. You can define routing policy matches based on the BGP communities. You can associate multiple communities with the same BGP route. BGP communities can be set, added, or deleted in a routing policy, as shown in the following sample output:

```
[edit policy-options policy-statement ibgp-export]
user@R1# set then community ?
Possible completions:
<community_name>      Name to identify a BGP community
+                      Add BGP communities to the route
-                      Remove BGP communities from the route
=                      Set the BGP communities in the route
add                   Add BGP communities to the route
delete                Remove BGP communities from the route
set                   Set the BGP communities in the route
```

Detailed coverage of BGP communities is outside the scope of this class.

Summary of BGP Active Route Selection

BGP Route Selection Summary	
1. Prefer the highest local-preference value	6. Prefer best exit from AS
2. Prefer the shortest AS-path length	7. For EBGP-received routes, prefer the current active route; otherwise, prefer routes from the peer with the lowest RID
3. Prefer the lowest origin value	8. Prefer paths with the shortest cluster length
4. Prefer the lowest MED value	9. Prefer routes from the peer with the lowest peer ID
5. Prefer routes learned from an EBGP peer over an IBGP peer	

Before the router installs a BGP route, it must make sure that the BGP next-hop attribute is reachable and that no routing loops exist. If the BGP next hop cannot be resolved or if a loop is detected, the route is not evaluated through the BGP route selection process or installed in the route table.

Before the Junos OS installs a BGP route in the route table the route preference is evaluated. Remember that the route preference can be changed through policy so the route preference can differ for the same prefix learned through different BGP paths. If the route preference for a BGP prefix learned through different BGP paths differs, the BGP route with the lower route preference is selected. Note that this is prior to the BGP selection process outlined in the table.

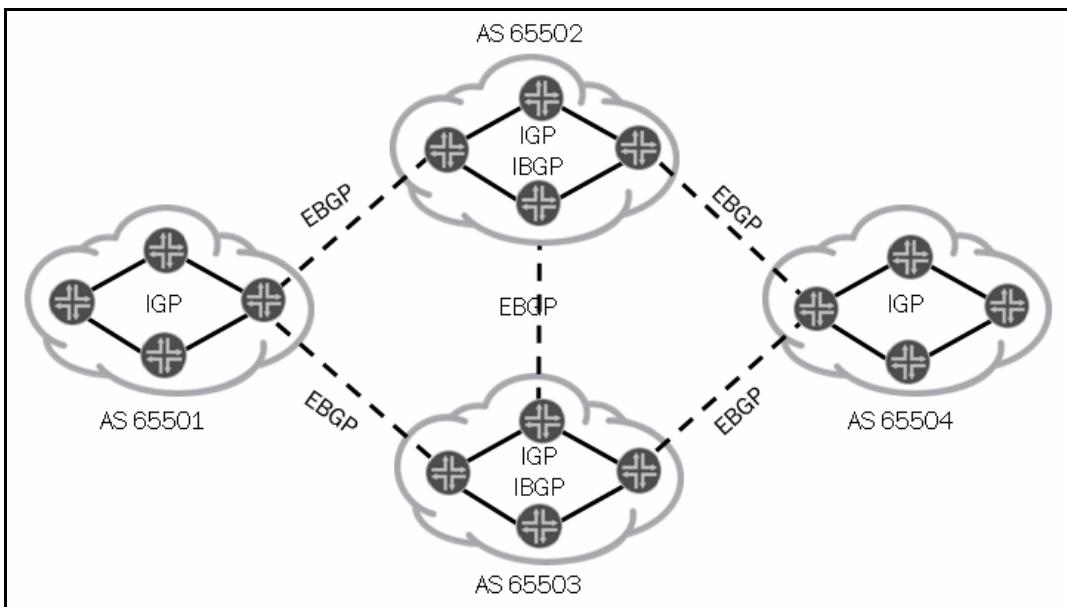
When a BGP route is installed in the routing table, it must go through a path selection process if multiple routes exist to the same destination prefix and the route preference is the same. The BGP path selection process proceeds in the following order:

1. The router then compares routes for the *highest* local preference (the only choice based on a higher, rather than lower, value).
2. The router evaluates the AS-path attribute next, where a shorter path is preferred. This attribute is often a common tiebreaker for routes.
3. The router evaluates the origin code. The lowest origin code is preferred (I [IGP] < E [EGP] < ? [Incomplete]).
4. If any of the remaining routes are advertised from the same neighboring AS, the router checks the MED attributes for a lowest value. The absence of a MED value is interpreted as a MED of 0.
5. If multiple routes remain, the router prefers any routes learned through an EBGP peer over routes learned through an IBGP peer. If all remaining routes were learned through EBGP, the router skips to Step 9.
6. If the remaining routes were learned through IBGP, use the path with the lowest IGP cost to the IBGP peer. For each IBGP peer, install a physical next hops based on the following three rules:
 - a. BGP examines both the `inet.0` and the `inet.3` routing tables for the BGP next-hop value. The physical next hops of the instance with the lowest JUNOS software preference is used. Often, this means that BGP uses the `inet.3` version of the next hop, via an MPLS LSP.
 - b. Should the preference values in the `inet.0` and the `inet.3` routing tables tie, the physical next hops of the instance in `inet.3` is used.
 - c. When a preference tie exists and the instances are in the same routing table, the number of equal-cost paths of each instance are examined. The physical next hops of the instance with more paths is installed. This tie might occur when the **traffic-engineering bgp-igp** knob is used for MPLS.
7. BGP then uses the route advertised from the peer with the lowest router ID (usually the loopback IP address). When comparing external routes from two distinct neighboring ASs, if the routes are equal up to the router ID comparison step, the currently active route is preferred. This preference helps prevent issues with MED-related route oscillation. The **external-router-id** command overrides this behavior and prefers the external route with the lowest router ID, regardless of which route is currently active.

8. The router then examines the cluster-list attribute for the shortest length. The cluster list is similar in function to an AS path.
9. The router prefers routes from the router with the lowest peer ID.

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

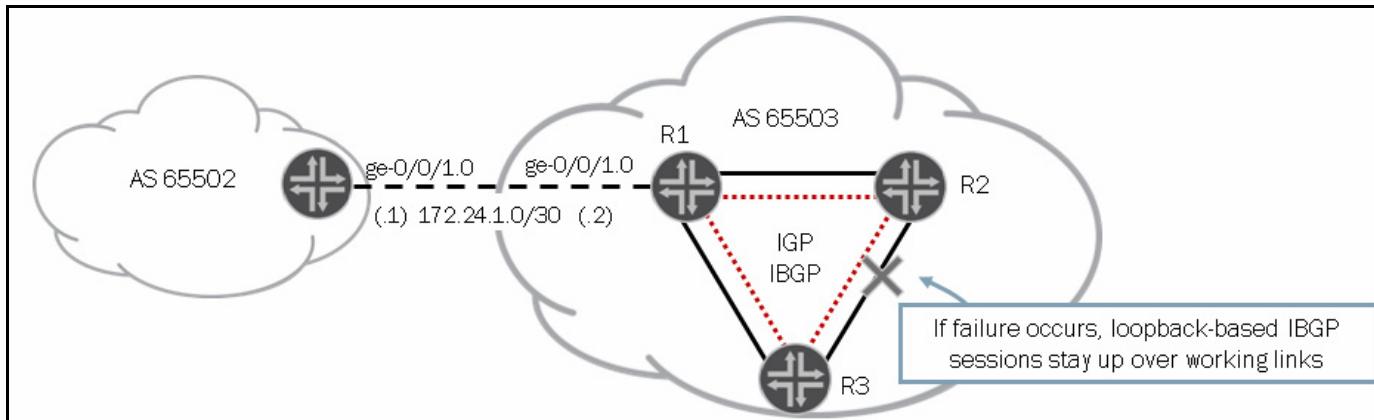
IBGP Versus EBGP



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

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

Loopback Peering



You maintain only one IBGP session between each internal peer. The IGP is used to maintain reachability between the loopback addresses regardless of the physical topology, allowing the IBGP sessions to stay up even when the physical topology changes.

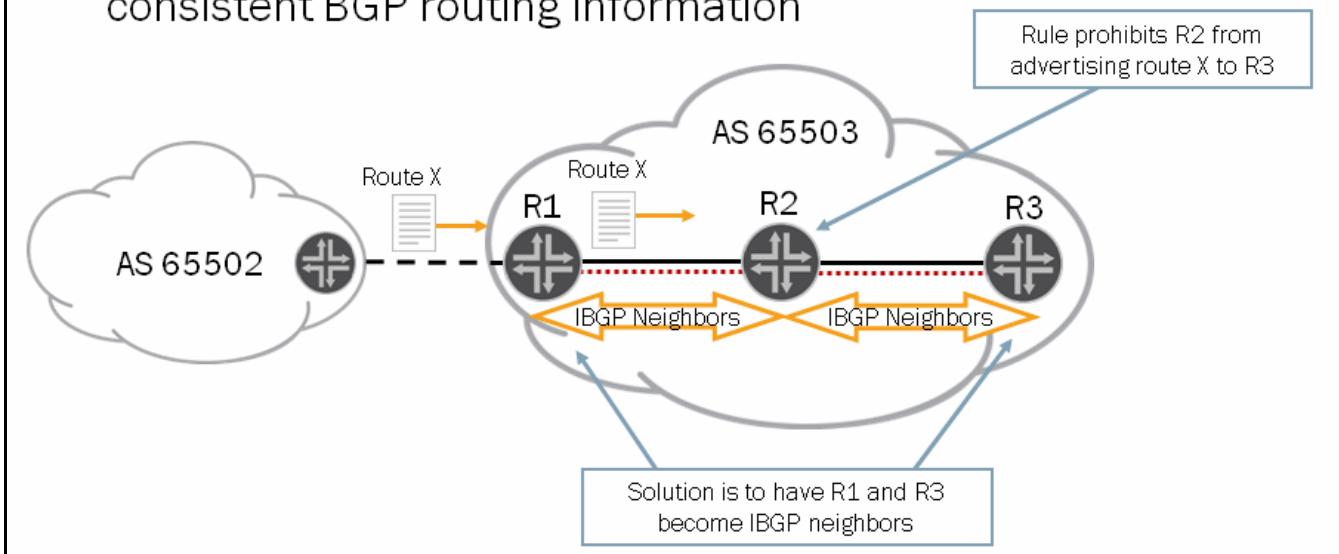
The physical topology is relevant in one respect: each router along the path between BGP speakers must have enough information to make consistent routing decisions about packet forwarding. In many cases, this requirement means that all routers along all possible physical paths between BGP speakers must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

Interface Peering

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

IBGP Route Propagation

- A full mesh is required to ensure all IBGP speakers have consistent BGP routing information



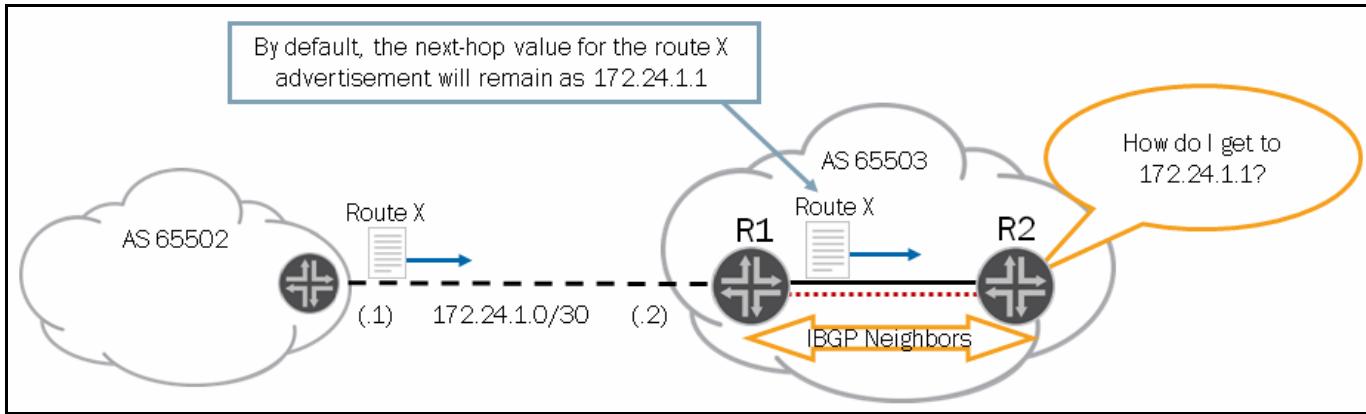
IBGP speakers send routes to their IBGP peers that they received from EBGP peers and routes that they originated themselves. IBGP speakers never send routes to IBGP peers that they learned from other IBGP peers. For all IBGP speakers in an AS to have consistent routing information, there must be a full mesh of IBGP sessions between all BGP speakers. Without this full mesh, some BGP speakers might not receive all the required routing information.

In the example on the graphic, there is not a full mesh of IBGP sessions. R1 receives the announcement through an EBGP session. Because it is the best route it has for that prefix, it propagates the route to its IBGP peer R2. R2 also determines that route to be its best path for the prefix; however, it does *not* send the route to its IBGP peer R3. Because it received the route through IBGP, it cannot send the route to any IBGP peers. Therefore, R3 does not receive or install a route for the prefix advertised from AS 65502. This situation can be alleviated by adding an IBGP session between R1 and R3. (It is irrelevant whether the two routers are directly connected.)

If IBGP routers readvertised IBGP routes to other IBGP peers, a loop would form. Because the AS path is not updated by each router, but rather only when the associated prefix is advertised to an EBGP peer, the AS path cannot be used to detect loops for BGP routes advertised within an AS. For this reason, BGP enforces advertisement rules that require the full-mesh peering of IBGP routers to ensure consistent routing information on all IBGP routers within the AS.

Using route reflectors or confederations can also alleviate this situation, both of which can reduce or alleviate the full-mesh requirement. Coverage of route reflectors and confederations is beyond the scope of this class.

IBGP Next-Hop Propagation



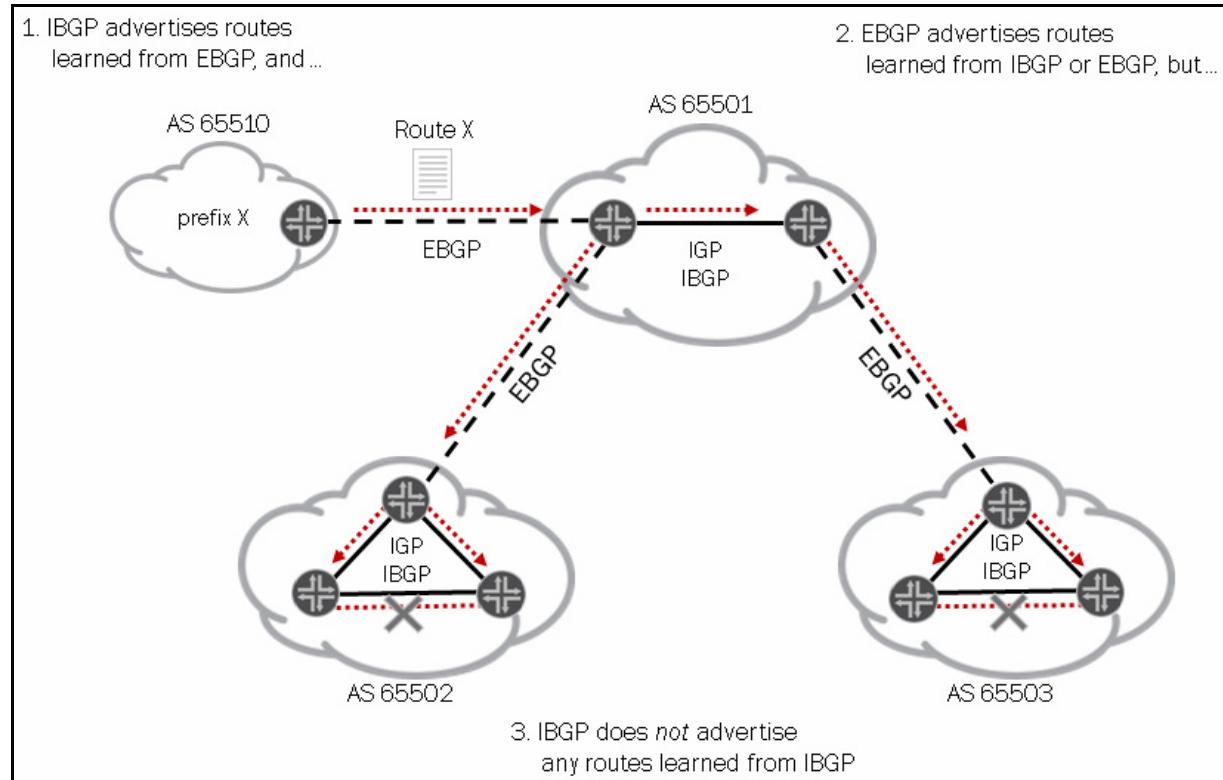
By default, the next-hop attribute attached to a route is unchanged as it passes through an AS. Because routers can use the BGP routes only if they already have a route to the next hop, you must either configure the routers to advertise external interfaces through the IGP or configure the routers to change the next-hop attribute attached to BGP routes using policy.

When EBGP speakers send routes to a peer, they set the next-hop attribute to the interface they share with that peer. In this example, R1 receives a route from its EBGP peer with the next-hop attribute set to 172.24.1.1. R1 sends this route to R2 without changing the next-hop attribute. Therefore, to use this route, R2 either must know how to reach 172.24.1.1 through the IGP or static routing, or R1 must send the routes with a different next hop.

You can send the appropriate external routes into the IGP, if wanted; however, using the `next-hop self` action in a policy has some advantages. Using the `next-hop self` action in a policy causes the router to send BGP routes to its peers using the same IP address it uses to establish that BGP session. For the BGP session to remain established, the peer must have a route to that IP address. Therefore, using `next-hop self` guarantees that a router's peers can reach the next hop of the routes that router sends, as long as the BGP session remains established.

Using the `next-hop self` action is fairly easy; we provide a sample configuration on a subsequent page.

Default BGP Advertisement Rules



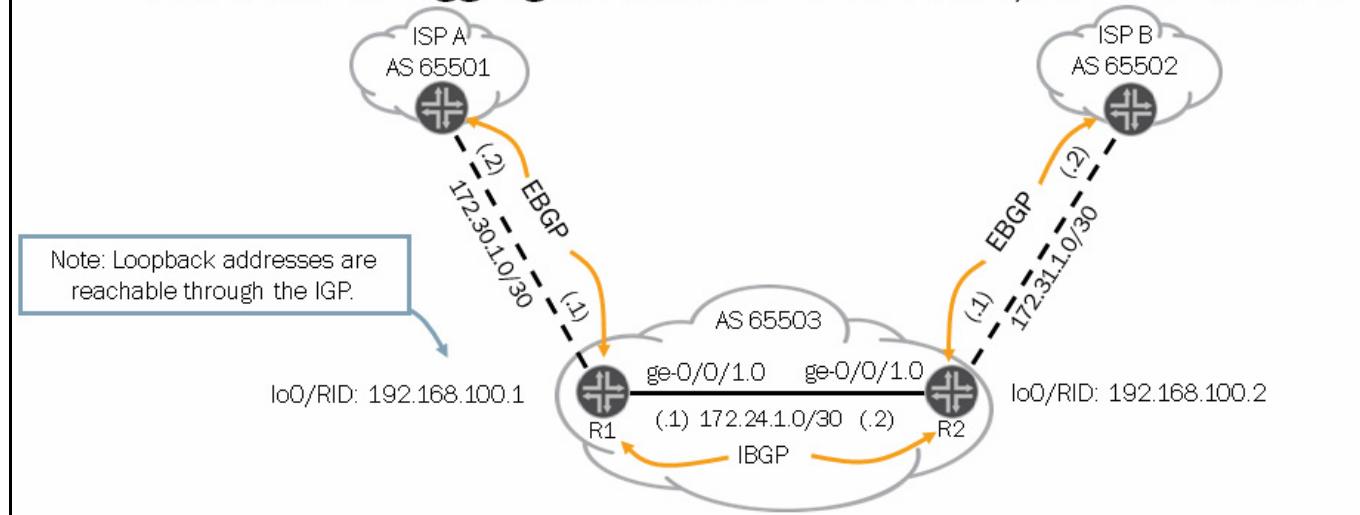
By default, only active BGP routes are advertised. The graphic illustrates the default BGP advertisement rules. The rules are as follows:

1. IBGP peers advertise routes received from EBGP peers to other IBGP peers.
2. EBGP peers advertise routes learned from IBGP or EBGP peers to other EBGP peers.
3. IBGP peers do not advertise routes received from IBGP peers to other IBGP peers.

The purpose of the advertisement rules is to prevent routing loops on a BGP network.

Case Study: Objectives and Topology

- Configure IBGP and EBGP as outlined on the diagram
 - Use loopback-based IBGP peering sessions
 - Ensure next-hop reachability for both EBGP peers
 - Advertise an aggregate route of 172.24.0.0/22 to both ISPs



The graphic lists the objectives for our case study and illustrates the relevant topology.

Case Study: Configuring BGP

```
[edit]
user@R1# show routing-options
router-id 192.168.100.1;
autonomous-system 65503; ← Device's assigned AS number

[edit]
user@R1# show protocols bgp
group int-65503 {
    type internal; ← BGP group names are user-defined
    local-address 192.168.100.1;
    neighbor 192.168.100.2;
}
group ext-65501 {
    type external; ← BGP session type determines if the
    peer-AS 65501; ← peering session is IBGP or EBGP
    neighbor 172.30.1.2;
}
```

The inset illustrates the sample configuration used by R1. A similar configuration is required on R2.

In this configuration example, we see some parameters defined under the [edit routing-options] and [edit protocols bgp] hierarchies. Under the [edit routing-options] hierarchy, we defined the system's router identifier (RID) and the local AS number. Optionally, you can configure the system's local AS number under the global BGP hierarchy for a specific BGP group, or, for a specific BGP neighbor, use the **local-as** configuration option. When the AS number is configured at multiple hierarchy levels, the AS number specified at the most specific hierarchy level is used. The ability to specify different AS numbers at different hierarchy levels can be quite useful, especially when merging networks with different AS numbers.

Because R1 and R2 are using loopback-based peering for their internal BGP group, they must reference loopback addresses in the related BGP configuration. In this case, the **neighbor** address is the remote peer's loopback address (R2 in our example). The **local-address** is the local device's loopback address (R1 in our example). If the local address is not specified, the system uses the interface address of the egress interface used to reach the referenced peer address. Because R2 is expecting to form an IBGP peering session with R1 using the 192.168.100.1, you must specify that address as the **local-address** in the configuration.

As mentioned in the inset, the session type determines if the peering session is IBGP or EBGP. You specify an **external** session type for EBGP and an **internal** session type for IBGP. If you omit the session type, you must specify the **peer-as** number, which can be a remote AS number or the local AS number. If the specified AS number does not match the AS number defined on the router, BGP assumes the session type is external. If the specified AS number does match the AS number defined on the router, BGP assumes the session type is internal. The software notifies you if you did not include the required details, as shown in the following sample output:

```
[edit protocols bgp]
user@R1# show
group x100 {
    neighbor 10.1.1.1;
}

[edit protocols bgp]
user@R1# commit
[edit protocols]
'bgp'
    Error in neighbor 10.1.1.1 of group x100:
    peer AS number must be configured for an external peer
    error: configuration check-out failed
```

Case Study: Changing the Next Hop

```
[edit]
user@R1# show policy-options
policy-statement next-hop-self-policy {
    term alter-next-hop {
        then {
            next-hop self; ← Using the self option alters the next-hop value to the
                           address used for the respective peering session
        }
    }
}

[edit]
user@R1# show protocols bgp
group int-65503 {
    type internal;
    local-address 192.168.100.1;
    export next-hop-self-policy; ← Policy is applied as an export
                                 policy for the internal BGP group
    neighbor 192.168.100.2;
}
group ext-65501 {
    type external;
    peer-AS 65501;
    neighbor 172.30.1.2;
}
```

The inset illustrates a sample policy used to alter the next-hop value. The illustrated policy matches on all BGP routes received from R1's EBGP peer and changes the next-hop address for those routes to the IP address used for the IBGP peering session (R1's loopback address). This policy is applied as an export policy to the internal BGP group.

We recommend that you always apply the `next-hop self` policy as an export policy to the internal peers or to the BGP group to which those peers belong. Improper application of a `next-hop self` policy can cause suboptimal routing or result in hidden routes.

When defining a `next-hop self` policy, ensure that you do not include the `accept` action in conjunction with the `next-hop` action. Using the `accept` action in conjunction with the `next-hop` action effectively matches all routes, BGP and otherwise, and advertises those routes to the configured IBGP peers.

Case Study: Advertising the Aggregate Route

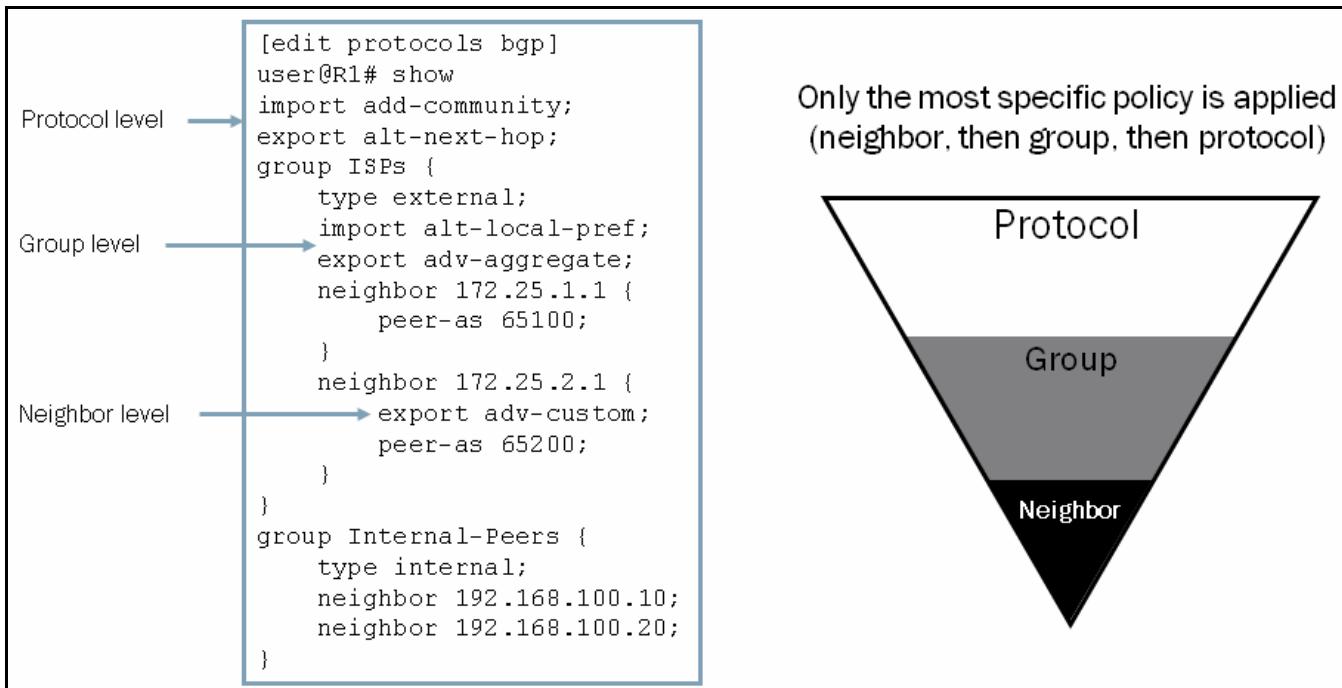
```
[edit]
user@R1# show routing-options aggregate
route 172.24.0.0/22; ← Aggregate route defined

[edit]
user@R1# show policy-options policy-statement adv-aggregate
term match-aggregate {
    from {
        protocol aggregate;
        route-filter 172.24.0.0/22 exact;
    }
    then accept;
}

[edit]
user@R1# show protocols bgp group ext-65501
type external;
export adv-aggregate; ← Redistribution policy applied as an
peer-AS 65501;           export policy to the EBGP group
neighbor 172.30.1.2;
```

The inset illustrates the required configuration for defining and advertising an aggregate route. We covered aggregate routes in a previous chapter in this study guide.

Applying Policy to BGP: Part 1



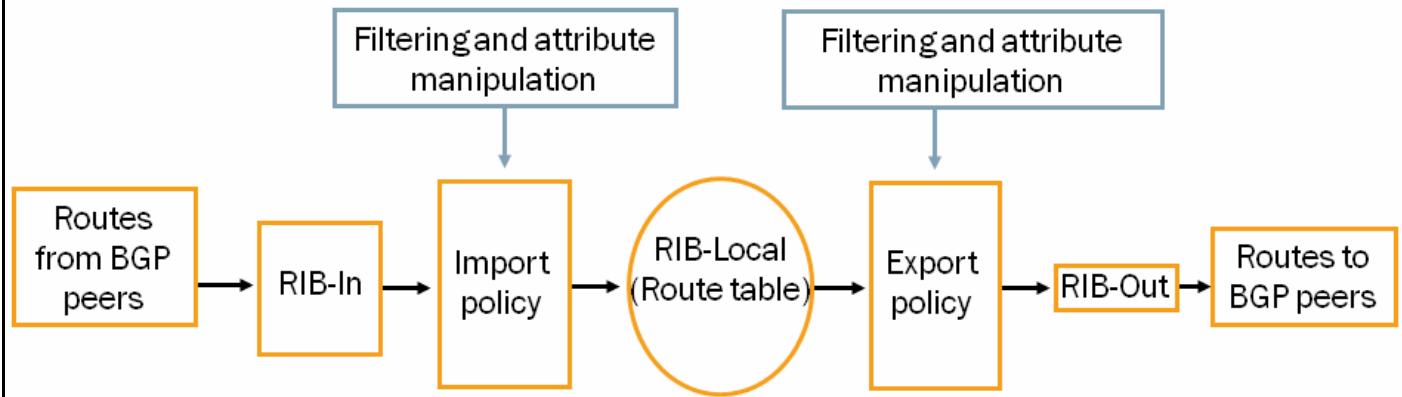
You can apply import and export policies to BGP sessions at the neighbor, group, or protocol level of the configuration hierarchy. The router always applies the most specific (and only the most specific) import or export policy. Therefore, import or export policies applied at higher levels of the configuration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

Applying Policy to BGP: Part 2

Import Policy Versus Export Policy

Import policies affect routes passed between the RIB-In and the RIB-Local tables

Export policies affect routes passed between RIB-Local and the RIB-Out tables



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

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

For example, suppose you have an OSPF route (preference 10) and a BGP route (preference 170) for the same prefix. An export policy can determine whether to send the active OSPF route and can modify attributes of the route as it is sent, but it cannot, by default, cause the inactive BGP route to be sent. You can override this default behavior and advertise inactive BGP routes that are inactive because of route preference by using the **advertise-inactive** configuration option.

Export policies are applied as routes are exported from the routing table, so attribute changes do not affect the local routing table; rather, they are applied to the route as it is exported. We illustrate the monitoring commands to view the BGP routes in the RIB-In, Local-RIB, and RIB-Out tables later in this section.

Displaying BGP Summary Information

```

user@R1> show bgp summary
Groups: 2 Peers: 2 Down peers: 0
Table      Tot Paths  Act Paths Suppressed   History Damp State     Pending
inet.0        12          6          0          0          0          0
Peer          AS InPkt OutPkt  OutQ Flaps Last Up/Dwn State |#Act/Rec/Acc/Damped.
172.30.1.2    65501    914     915      0      0    6:51:16 6/6/6/0           0/0/0/0
192.168.100.2 65503    978     983      0      0    7:19:03 0/6/6/0           0/0/0/0
  
```

The output fields of the **show bgp summary** command are the following:

- Groups: Displays the number of BGP groups;
- Peers: Displays the number of BGP peers;
- Down peers: Displays the number of unestablished BGP peers;

- **Peer:** Displays the address of each BGP peer; each peer has one line of output;
- **AS:** Displays the peer's AS number;
- **InPkt:** Displays the number of packets received from the peer;
- **OutPkt:** Displays the number of packets sent to the peer;
- **OutQ:** Displays the count of the number of BGP packets queued to be transmitted to a particular neighbor; it usually is 0 because the queue is emptied quickly;
- **Last Up/Down:** Displays the last time since the neighbor transitioned to or from the established state; and
- **State:** Displays either the BGP state or, if the neighbor is connected, the number of paths received from the neighbor, the number of these paths that have been accepted as active and are being used for forwarding, and the number of routes being damped.

Displaying BGP Neighbors

```
user@R1> show bgp neighbor
Peer: 172.30.1.2+62790 AS 65501 Local: 172.30.1.1+179 AS 65503
  Type: External    State: Established    Flags: <Sync>
  Last State: OpenConfirm   Last Event: RecvKeepAlive
  Last Error: None
  Export: [ adv-aggregate ]
  Options: <Preference PeerAS Refresh>
  Holdtime: 90 Preference: 170
  Number of flaps: 0
  Peer ID: 172.18.1.1      Local ID: 192.168.100.1      Active Holdtime: 90
  Keepalive Interval: 30      Peer index: 0
  BFD: disabled, down
  Local Interface: ge-0/0/3.0
...
...
```

Some of the output fields of the **show bgp neighbor** command are the following:

- **Peer:** Displays the address of each BGP peer; each peer has one line of output.
- **Type:** Displays the type of peer (Internal or External).
- **State:** Displays the BGP state for this neighbor.
- **Flags:** Displays the internal peer-specific flags for this neighbor.
- **Last State:** Displays the BGP state of this neighbor prior to the current state.
- **Last Event:** Displays the last BGP state transition event.
- **Last Error:** Displays the last notification message sent to the neighbor.
- **Options:** Displays the configuration options in effect for this neighbor.
- **Holdtime:** Displays the configured hold time for this neighbor.
- **Preference:** Displays the configuration preference for routes learned from the neighbor.
- **Peer ID:** Displays the neighbor's router ID.
- **Local ID:** Displays the local system's router ID.
- **Active Holdtime:** Displays the hold-time value that was negotiated during the BGP open.
- **Table inet.0 Bit:** Displays the Internal bit used for the peer group.
- **Send state:** Displays whether all peers in the group have received all their updates (in sync or out of sync).
- **Active Prefixes:** Displays the number of prefixes accepted as active from this neighbor.

- **Last traffic (seconds):** Displays how recently a BGP message was sent or received between the local system and this neighbor.
- **Output Queue:** Displays the number of BGP update messages pending for transmission to the neighbor.
- **Deleted routes:** Displays the prefixes queued for withdrawal through pending update messages.
- **Queued AS Path:** Displays an AS path queued for transmission in an update message.

Displaying BGP Group Information

```
user@R1> show bgp group
Group Type: Internal      AS: 65503          Local AS: 65503
  Name: int-65503        Index: 0            Flags: <Export Eval>
  Export: [ next-hop-self-policy ]
  Holdtime: 0
  Total peers: 1         Established: 1
  192.168.100.2+51067
  inet.0: 0/6/6/0

Group Type: External       Local AS: 65503
  Name: ext-65501        Index: 1            Flags: <Export Eval>
  Export: [ adv-aggregate ]
  Holdtime: 0
  Total peers: 1         Established: 1
  172.30.1.2+62790
  inet.0: 6/6/6/0
...
...
```

Use the **show bgp group** command to view BGP group information. Some of the output fields of this command are the following:

- **Group Type:** Displays the type of BGP group. It can be either **Internal** or **External**.
- **AS:** Displays the number of the remote AS. For IBGP, this number should be the same as the local AS number.
- **Local AS:** Displays the number of the local AS.
- **Export:** Displays the export policies configured for the BGP group with the **export** statement.
- **Total peers:** Displays the total number of peers in the group.
- **Established:** Displays the number of peers within the group that are in the established state.
- **ip addresses:** Displays the list of peers that are members of the group; the address is followed by the peer's port number.
- **Options:** Displays configured BGP options; these options can be one or more of the following:
 - **Local address:** Displays the address configured with the **local-address** statement.
 - **NLRI:** Displays the configured MBGP state for the BGP group; it can be either multicast or unicast, or both if you have configured **nlri any**.
 - **Hold time:** Displays the hold time configured with the **hold-time** statement; the default hold time is 90 seconds.
 - **Preference:** Displays the preference value configured with the **preference** statement; the default preference value is 170.

Displaying Installed BGP Routes

```
user@R1> show route protocol bgp

inet.0: 15 destinations, 21 routes (15 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.0/16      *[BGP/170] 1d 21:43:42, localpref 100
                  AS path: 64501 65500 65501 65502 65503 I
                  > to 172.30.1.2 via ge-0/0/3.0
                  [BGP/170] 1d 21:43:42, localpref 100, from 192.168.100.2
                  AS path: 64502 65400 65501 65502 65503 I
                  > to 172.24.1.2 via ge-0/0/1.0
...
...
```

BGP routes received from other neighbors are stored in the main routing table, along with routes from other sources. Use the **show route protocol bgp** command to display only BGP routes. You can combine this command with the **extensive** or **detail** switches to display the various attributes associated with a route. You can also filter BGP routes based on AS path or community regular expressions.

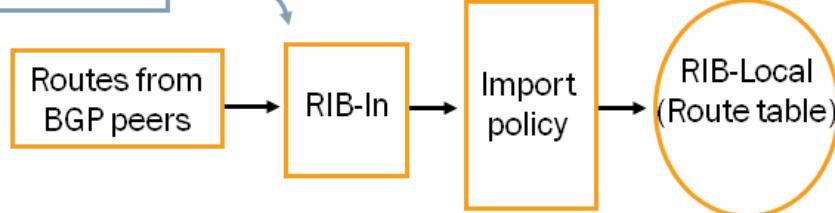
Add the **hidden** switch to see prefixes that are hidden. A BGP route might be hidden for various reasons such as import policy filtering, the next hop is unreachable, or route damping.

Displaying Received BGP Routes

```
user@R1> show route receive-protocol bgp 172.30.1.2

inet.0: 14 destinations, 20 routes (14 active, 0 holddown, 0 hidden)
  Prefix      Nexthop      MED    Lclpref    AS path
* 10.0.0.0/16    172.30.1.2                      65501 65510 65515 65520 65525 I
* 10.1.0.0/16    172.30.1.2                      65501 65510 65515 65520 65525 I
* 10.2.0.0/16    172.30.1.2                      65501 65510 65515 65520 65525 I
...
...
```

Displays route entries in the RIB-In table that have not yet been filtered

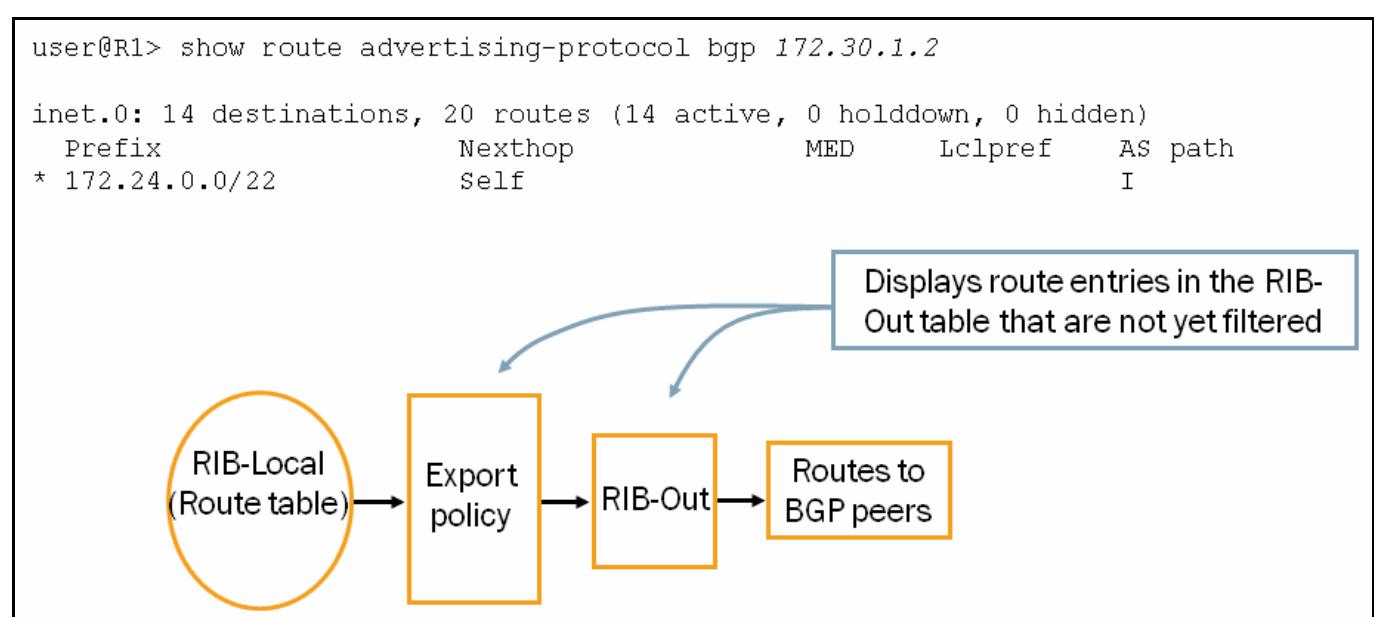


The **show route receive-protocol bgp neighbor** command displays the BGP routing information as it was received from the specified BGP neighbor and before any import policy processing occurs. For example, if you apply an import policy to remove all community values from BGP routes received from a given peer and that peer advertises routes to your device with a community value set, the generated output for the **show route receive-protocol bgp neighbor** command will display the community value set by the advertising peer.

The output for the **show route receive-protocol bgp neighbor** command does not, however, display BGP routes rejected by an import policy. You can add the **hidden** command option to display routes that are hidden because of an import policy.

To verify the effects of your import policy, you can display the routes as they exist in the routing table using the **show route** command.

Displaying Advertised BGP Routes



The **show route advertising-protocol bgp neighbor** command displays the routing information as it has been prepared for advertisement to the specified BGP neighbor. The information reflects the routes that the routing table exported into the routing protocol, along with any attributes that have been attached. Note that the local AS-number is not shown in the output of this command.

Review Questions

1. What happens if a router receives a BGP route with its own AS number in the AS path?
2. What are the advantages of loopback peering for IBGP sessions?
3. Describe the default BGP advertisement rules.
4. Which Junos OS CLI commands allow you view BGP routes received from and advertised to a BGP peer?

Answers

1.

One of the primary purposes of the AS path is to prevent routing loops. If a router receives a BGP route that includes that router's AS number in the AS path list, that router immediately drops the route. Currently, Junos devices do not even send an update to a remote peer if that remote peer's AS number appears in the AS path list.

2.

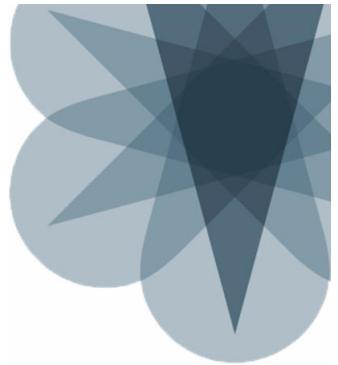
Using the loopback addresses as the endpoints for IBGP sessions ensures that the session stays up as long as a route between the two endpoints exists. This setup allows the IBGP session to stay up irrespective of the state of its local physical interfaces or topology changes between the two routers, promoting stability of the IBGP session.

3.

IBGP peers advertise routes received from EBGP peers to other IBGP peers; EBGP peers advertise routes learned from IBGP or EBGP peers to other EBGP peers; IBGP peers do not advertise routes received from IBGP peers to other IBGP peers.

4.

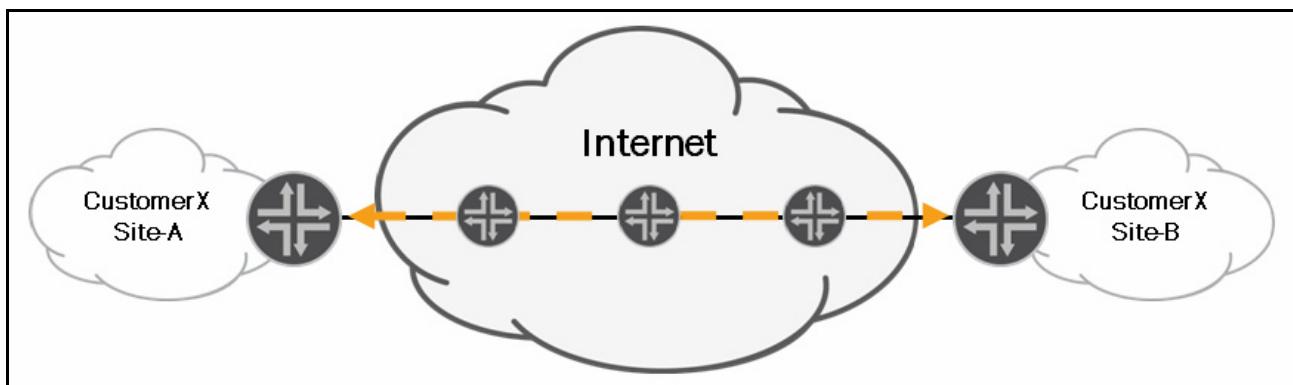
The **show route receive-protocol bgp neighbor** command displays the BGP routing information that was received from the specified BGP neighbor. The **show route advertising-protocol bgp neighbor** command displays the routing information as it has been prepared for advertisement to the specified BGP neighbor.

**JNCIS-SP Study Guide—Part 1****Chapter 5: IP Tunneling**

This Chapter Discusses:

- IP tunneling concepts and applications;
- Basic operations of generic routing encapsulation (GRE) and IP over IP (IP-IP) tunnels; and
- Configuration and monitoring of GRE and IP-IP tunnels.

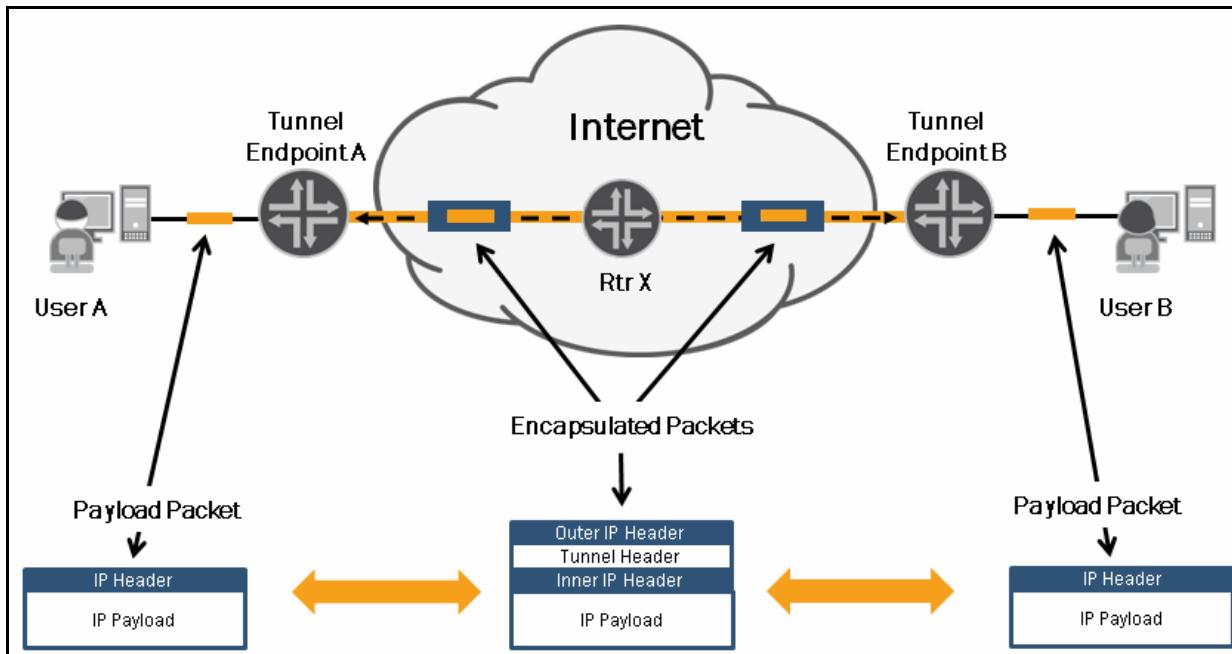
What Is an IP Tunnel?



In its most basic form, an IP tunnel is a communications channel between two different networks. The networks joined by an IP tunnel are often geographically dispersed and have no native routing path between one another. Because no native routing paths exist between the remote networks, a transport network, such as the Internet, is required to allow the tunnel to form and IP communications to occur.

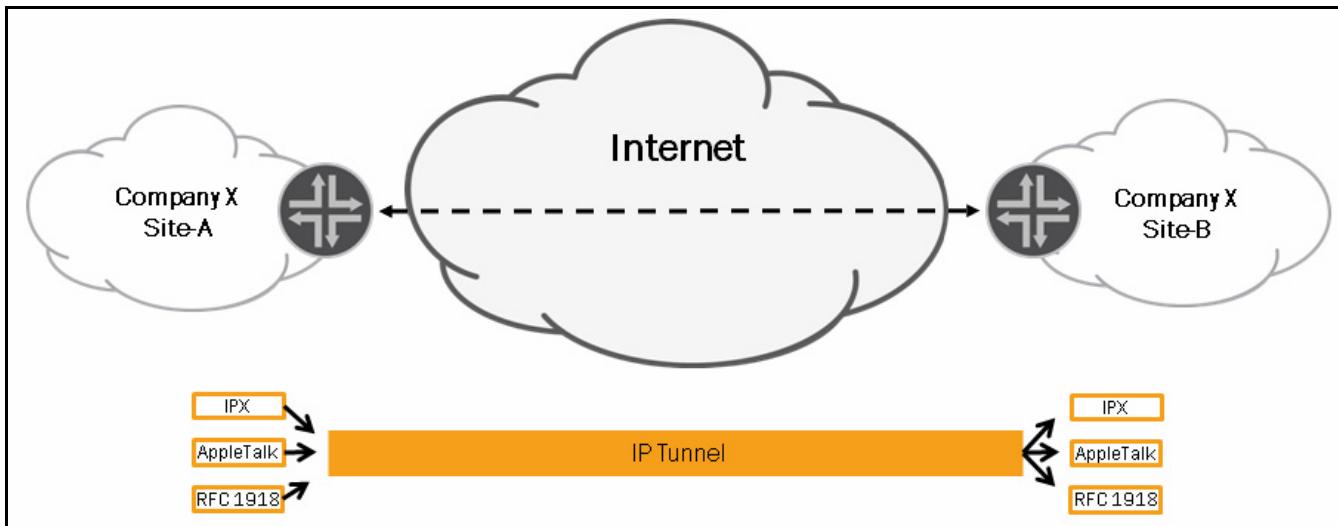
IP tunnels require a tunneling protocol, which can provide secure or unsecure communications, depending on the protocol selected. Tunneling protocols can use data encryption to transport insecure payload protocols over a public network (such as the Internet), thereby providing VPN functionality. IP Security (IPsec) has an end-to-end transport mode but can also operate in a tunneling mode through a trusted security gateway. We cover IPsec in detail in other training courses. In this chapter, we highlight GRE and IP-IP, which are unsecure tunneling protocols.

Tunneling IP Packets



As a packet enters an IP tunnel, the tunneling protocol encapsulates the entire packet, including the header. Once the packet is encapsulated, it is forwarded by the encapsulating router, known as the encapsulator, to the next hop in the path toward the remote tunnel endpoint. Once the packet is received by the remote tunnel endpoint, known as the decapsulator, the packet is de-encapsulated and forwarded based on the original header contents. The tunnel endpoints typically reside at the edge of the source and destination networks and between the source or destination network and the transport IP network (often the Internet). A router can serve as an encapsulator and decapsulator at the same time.

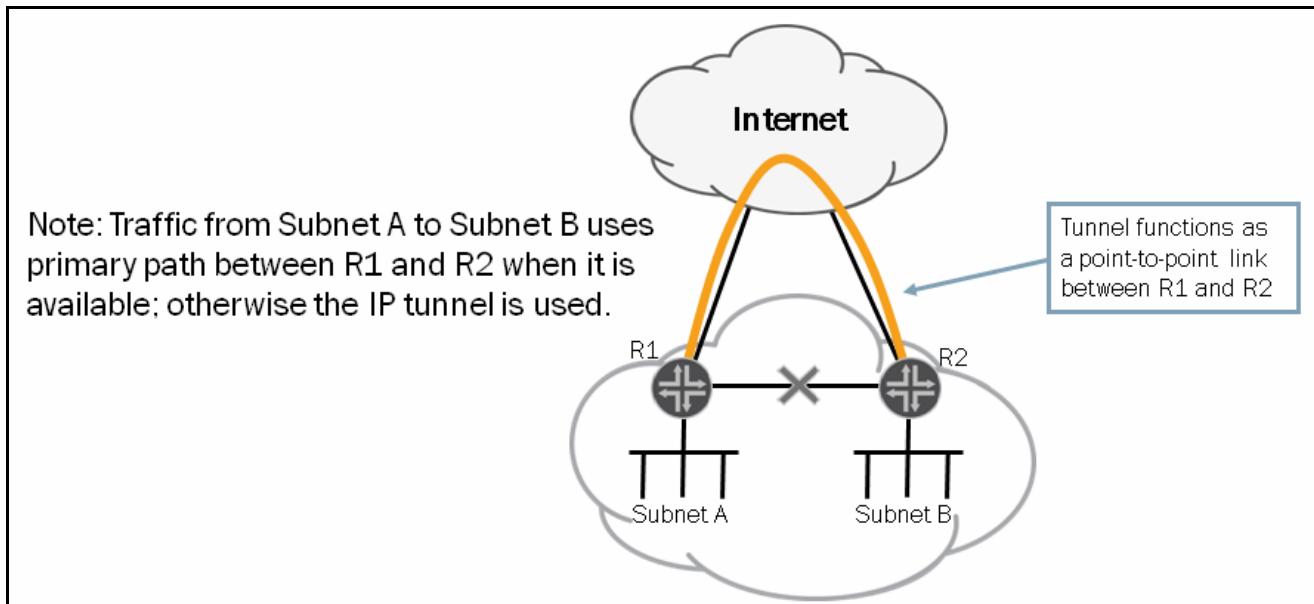
Uses of IP Tunnels: Part 1



By default, only IP traffic using routable IP addresses is permitted through the Internet. Routable IP addresses are defined by the Internet Assigned Numbers Authority (IANA). Examples of nonroutable IP addresses include the private IP address ranges defined in RFC 1918.

As the graphic indicates, you can use IP tunnels to route non-IP traffic, such as IPX or AppleTalk, or IP traffic with nonroutable IP addresses over a public network, such as the Internet. Note that Junos operating system does not support IPX or AppleTalk. We highlight other uses of IP tunnels in the next section.

Uses of IP Tunnels: Part 2

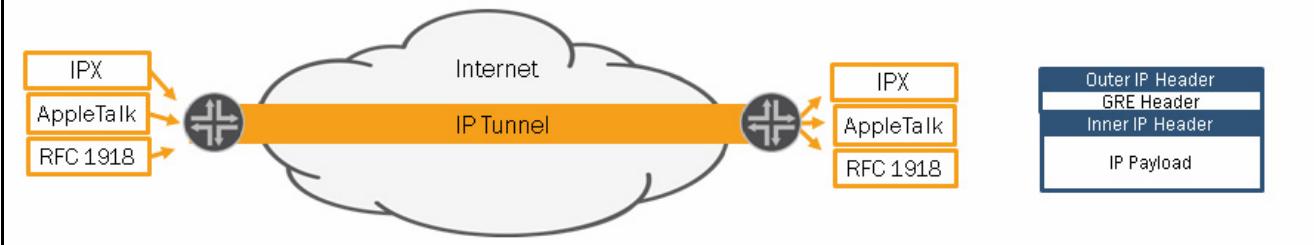


Another use of IP tunnels is to incorporate a redundant link to be used during failure scenarios. An IP tunnel is a logical point-to-point link between the tunnel end-points and is capable of the same functions as any other link. In other words, a tunnel interface can serve as a next hop for static routes or participate in the network's IGP. Note that routing protocols might prefer a tunnel over a physical link because the tunnel appears to be a one-hop link with the lowest cost path. In reality, however, tunnels involve multiple hops and are generally more costly than other paths. You can ensure the IP tunnel serves as the backup path through administrative settings such as cost and route preference.

Other uses of IP tunnels exist; however, they are not covered in the graphic. You can use IP tunnels to bypass simple firewall rules because the original IP header contents are hidden from most transit devices (those devices between the tunnel end-points). You can also use IP tunnels to tunnel IPv6 traffic between remote IPv6 networks.

Generic Routing Encapsulation

- GRE is an IP tunneling protocol that can encapsulate a wide variety of Network Layer protocol packet types



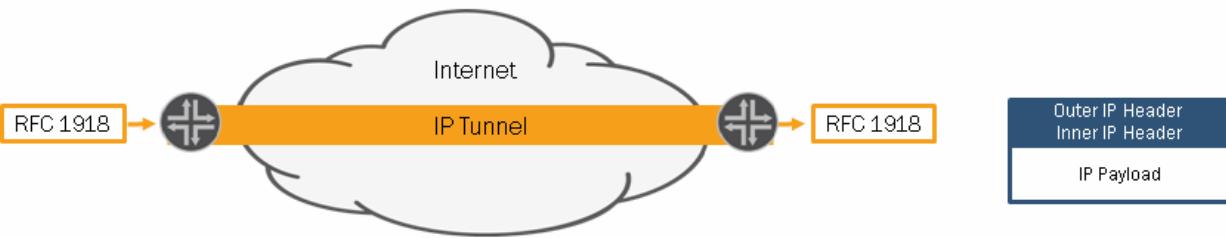
GRE is a tunneling protocol that can encapsulate a wide variety of Network Layer protocol packet types inside IP tunnels, creating a virtual point-to-point link between the routers at remote ends of the tunnel. As indicated in the graphic, GRE tunnels are used to tunnel IP traffic as well as non-IP traffic, such as IPX and AppleTalk. In addition to the referenced protocol packet types, GRE tunnels are also commonly used to tunnel IPv6 and MPLS protocol traffic over an IP transport network. In these situations, GRE tunnels connect isolated IPv6 networks and serve in place of the tunnels normally created by the RSVP and the LDP, respectively. Detailed coverage of tunneling IPv6 and MPLS traffic is beyond the scope of this chapter.

To encapsulate a GRE packet in an IP packet, a GRE header and an outer IP header are added. The GRE header and outer IP header add an additional 24 bytes of overhead to the packet. The outer header includes the source IP address, the entry point of the tunnel, and the destination IP address—the exit point of the tunnel. The inner packet (also known as the payload packet) is not modified, except the time-to-live (TTL) field, which is decremented. The TTL field must be decremented to ensure that the packet does not live forever. GRE packets that are encapsulated in IP packets use IP protocol type 47.

Although GRE was originally developed by Cisco Systems, it is now a standard and is supported by many vendors, including Juniper Networks. RFC 1702 defines GRE over IPv4 networks.

IP-IP

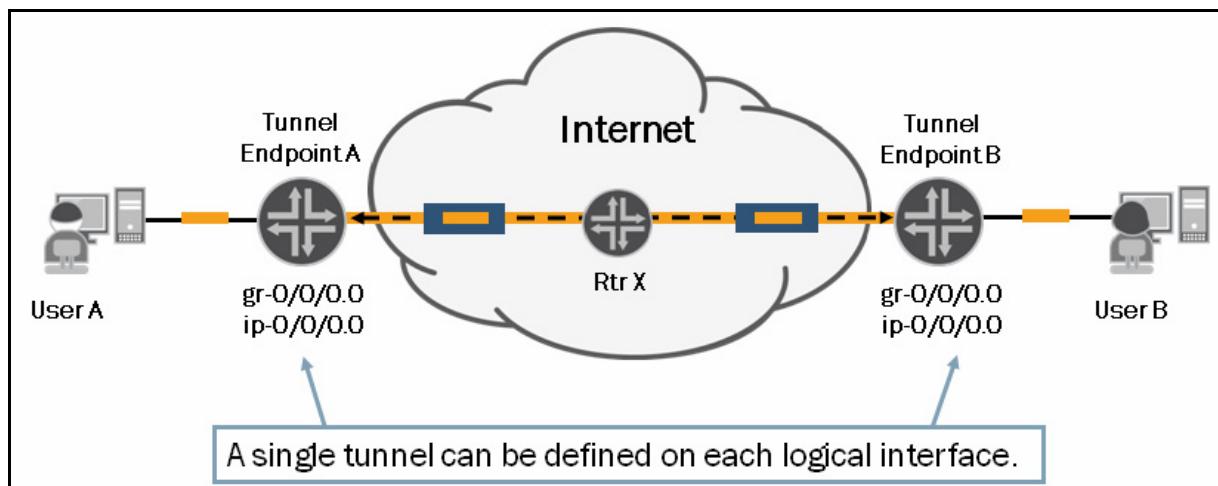
- IP-IP is an IP tunneling protocol that can encapsulate one IP packet inside another IP packet



IP-IP is a tunneling protocol used to encapsulate one IP packet in another IP packet. To encapsulate an IP packet using IP-IP encapsulation, an outer IP header is inserted before the packet's existing IP header. The additional IP header adds 20 bytes of overhead to the packet. The outer IP header source and destination addresses identify the tunnel endpoints. The inner IP header source and destination addresses identify the original sender and recipient of the datagram, respectively. The encapsulator does not change the inner IP header, except to decrement the time to live (TTL), and it remains unchanged during its delivery to the tunnel endpoint.

Because IP-IP tunnels can tunnel only IP traffic, they are not as commonly used as GRE tunnels, which can tunnel both IP and non-IP traffic. IP-IP encapsulation is defined in RFC 2003.

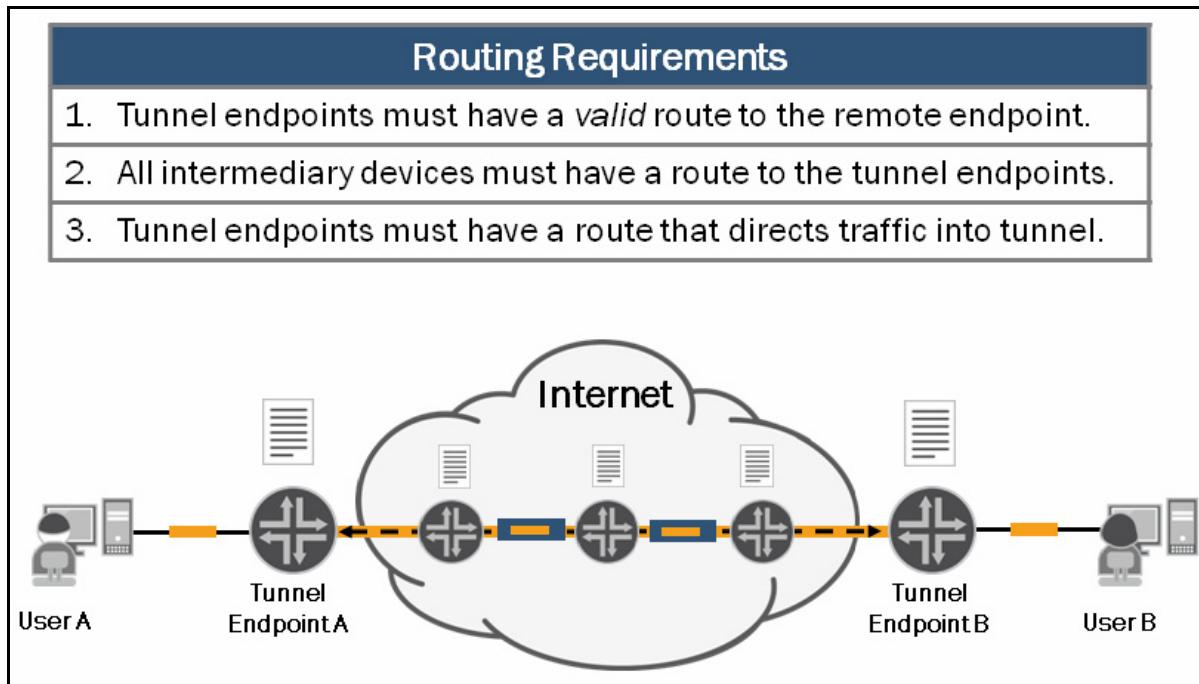
Tunnel Requirements: Part 1



You must define a tunnel interface on each endpoint of the tunnel. If you use a GRE tunnel, you define a tunnel interface in the form of $gr-x/y/z$, where $x/y/z$ represents the placement of the interface card used for tunneling services. On some Junos devices, the tunneling services are performed by the software rather than a hardware component, such as a physical interface card (PIC). In platforms where the tunneling services are performed in software, use $gr-0/0/0$ as the tunnel interface. Similarly, if you use an IP-IP tunnel, you must define a tunnel interface in the form of $ip-x/y/z$, where $x/y/z$ represents the placement of the tunneling services interface card. Not all Junos devices support GRE and IP-IP tunnel interfaces; in some cases, they might have special hardware requirements or configuration considerations. Check your product-specific documentation for support information.

You can configure multiple logical units for each GRE or IP-IP interface, and you can configure only one tunnel per unit. Each tunnel interface is a point-to-point interface. You must specify the tunnel's destination and source addresses. All other statements are optional. We cover tunnel interface configuration later in this chapter.

Tunnel Requirements: Part 2

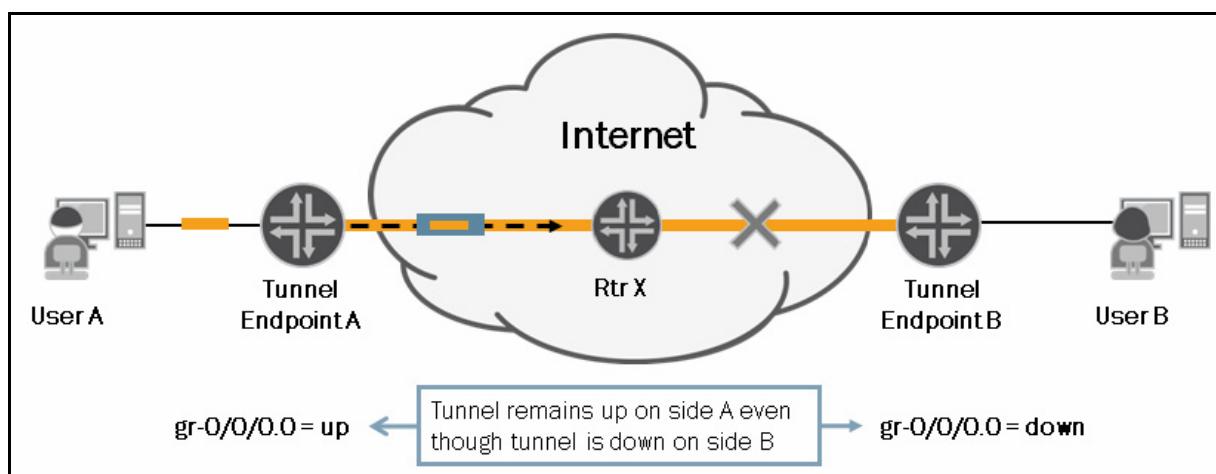


The graphic illustrates other requirements for proper GRE and IP-IP tunnel operations. As indicated in the graphic, the tunnel endpoints must have a valid route to the remote tunnel endpoint. This route must resolve to the physical next hop in the path toward the remote tunnel endpoint and cannot use the tunnel as the next hop.

Similarly, the intermediary devices, located in the forwarding path between the tunnel endpoints, must be able to route to each of the tunnel endpoints. Note that the tunnel is often established using the loopback address assigned to the tunnel endpoints. Using loopback addresses rather than IP addresses assigned to physical interfaces on the tunnel endpoints provides added some redundancy when multiple paths between the tunnel endpoints exist.

The tunnel endpoints also need a route to direct traffic destined to the remote subnet into the tunnel. This route uses the tunnel interface as the next hop. We provide a sample configuration for a route that directs traffic through a tunnel on a subsequent page.

Tunnel Considerations: Part 1



By default, GRE and IP-IP tunnels are completely stateless, which means the tunnel endpoints do not know about the state or availability of the remote tunnel endpoint. Consequently, the local tunnel endpoint does not have the ability to bring the GRE tunnel interface down if the remote endpoint is unreachable. The inability to mark an interface down when the remote tunnel endpoint is unavailable means that any associated routes dependant on that interface (such as static routes) remain in the routing table whether or not the tunnel is actually usable. The described behavior can cause problems in certain situations.

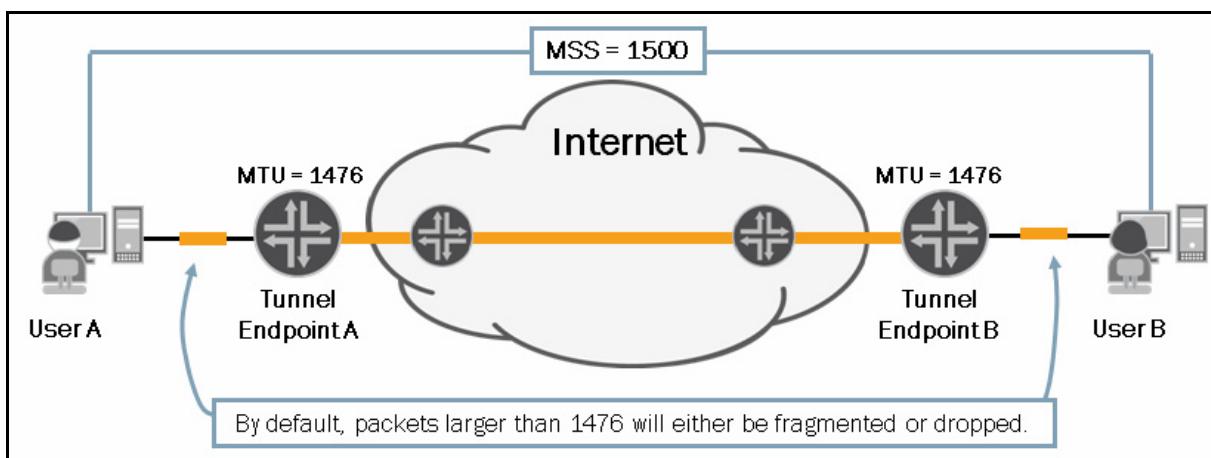
Some implementations of GRE support a keepalive mechanism to monitor the health and availability of the tunnel endpoints. Using keepalives to monitor the tunnel can greatly reduce the potential risks of sending traffic through stateless tunnels. The following is a configuration example of a GRE keepalive:

```
[edit]
user@host# show
protocols {
    oam {
        gre-tunnel {
            interface gr-1/1/10.1 {
                keepalive-time 10;
                hold-time 30;
            }
        }
    }
}
```

When configuring the keepalive, you must specify the `family inet` statement at the `[edit interfaces interface-name unit unit]` hierarchy level. The hold time must also be twice the keepalive time. GRE keepalives are supported on the M Series and MX Series devices.

You can also use Bidirectional Forwarding Detection (BFD) in conjunction with GRE tunnels to help accomplish the same basic functionality. We cover BFD in detail in the “High Availability” chapter.

Tunnel Considerations: Part 2



The IP protocol was designed to be used over a variety of interface types. Although the maximum length of an IP packet is 64 KB, most interface types enforce a significantly smaller maximum packet size, known as a maximum transmission unit (MTU). The MTU used depends on the interface type. The IP protocol accommodates different MTU values by allowing routers to fragment IP packets as necessary. The host receiving fragmented packets is responsible for reassembling the fragments back into the original packet.

IP fragmentation involves breaking a packet into several pieces that can later be reassembled. The IP source, destination, identification, total length, and fragment offset fields, along with the *more fragments* and *don't fragment* flags in the IP header, are used for IP fragmentation and reassembly.

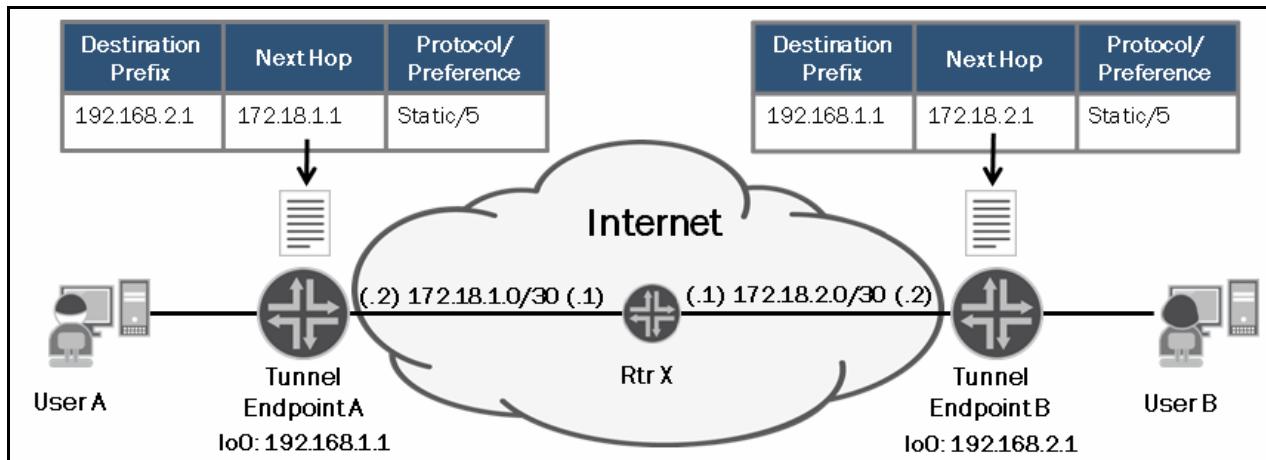
When two devices communicate using TCP, they determine the maximum segment size (MSS) permitted over the end-to-end communications path. Typically, the MSS turns out to be 1500 bytes, which is the maximum amount of payload in an Ethernet frame. This approach works well until overhead is added somewhere between the communicating devices, such as in the case of IP-IP and GRE tunnels where additional headers are added. When packets traverse a GRE tunnel, for example, the maximum packet size is reduced from 1500 to 1476 to account for the additional overhead of 24 bytes (4-byte GRE header + 20-byte IP header).

In this situation, if either host sends a packet that is larger than 1476, the packet must be fragmented (typically by one of the tunnel endpoints) to cross the tunnel. Unfortunately, the hosts do not realize this fact, because they have only communicated with one another and agreed upon the 1500 value.

If the transmitting host sets the *don't fragment* (DF) bit, the router drops the packet and typically sends an Internet Control Message Protocol (ICMP) message informing the sender to use smaller packets.

To avoid this problem, you can increase the MTU for the tunnel. We recommend that you set the MTU for a tunnel to 1524 or better. On some Junos devices, you can also enable the **clear-dont-fragment** configuration option, which clears the DF bit and allows fragmentation to occur. For support details, refer to the technical publications for your specific product. We cover tunnel interface configuration options later in this chapter.

Tunnel Considerations: Part 3

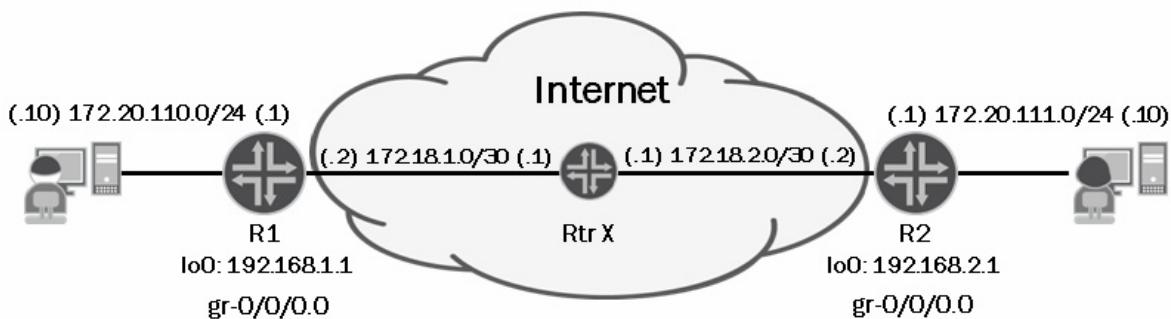


Each tunnel endpoint must have a valid route to the remote tunnel endpoint. For proper tunnel operations, this route must resolve to the physical next hop in the end-to-end communications path and should never use a recursive route, which is a route for which the best path to the remote tunnel endpoint is through the tunnel itself. If a tunnel endpoint uses the tunnel as the best path to the remote tunnel endpoint, the tunnel interface bounces and becomes unusable.

We recommend you configure static routes on the tunnel endpoints for the destination tunnel endpoint to avoid problems caused by recursive routing.

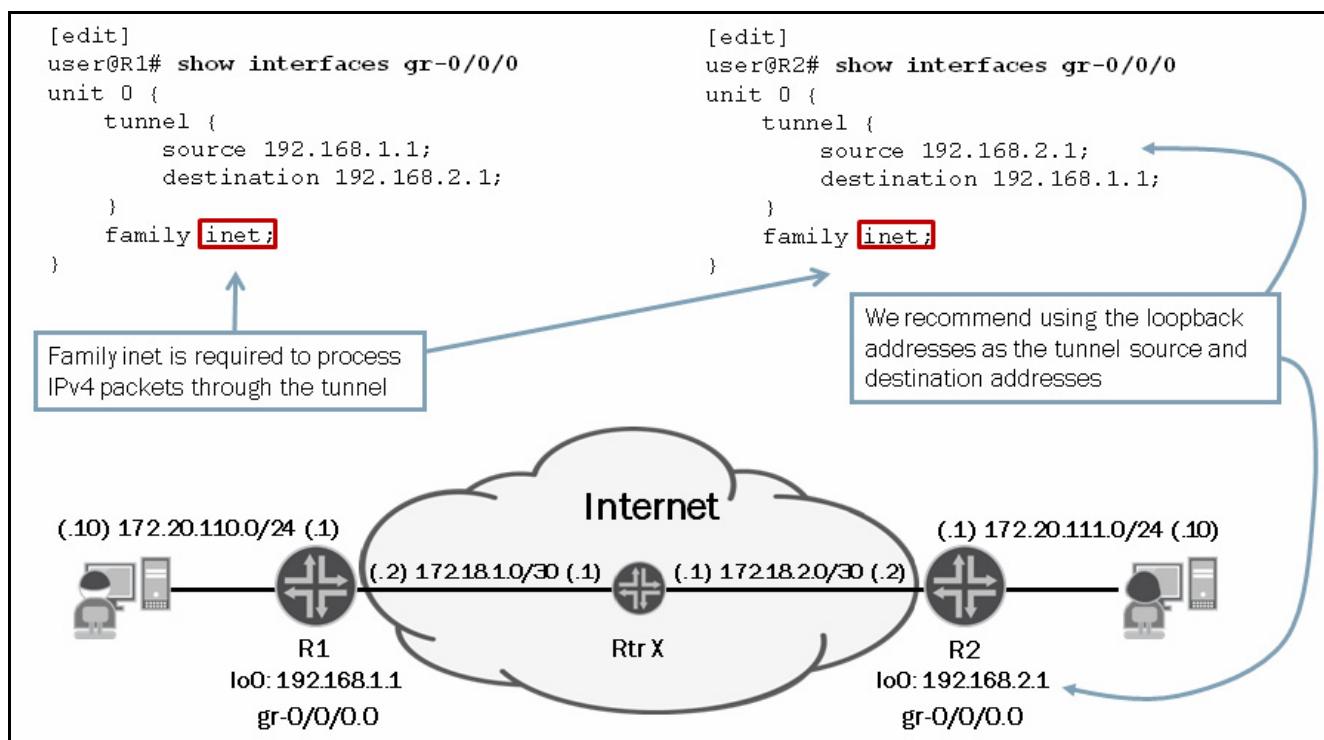
Case Study: Objectives and Topology

- Use the sample topology and implement a GRE tunnel between R1 and R2 that carries traffic destined to the remote 172.20.11x.0/24 subnet



The graphic introduces the case study for implementing a GRE tunnel. Note that the steps outlined in this case study are equally applicable to IP-IP tunnels. We cover the configuration and verification steps on subsequent pages.

Case Study: Defining the Tunnel Interface



The graphic illustrates basic GRE interface configurations used to process IPv4 packets. In addition to the basic configuration options listed on the graphic, you can also configure some additional options. The following is a list of some of the tunnel options:

- `copy-tos-to-outer-ip-header`: Unlike IP-IP tunnels, GRE tunnels do not copy the type of service (ToS) bits to the outer IP header by default. To have the RE copy the inner ToS bits to the outer IP header on packets sent by the RE, include the `copy-tos-to-outer-ip-header` statement at the logical unit hierarchy level of a GRE interface.
- `allow-fragmentation`: By default, the GRE-encapsulated packets are dropped if the packet size exceeds the MTU setting of the egress interface. Use the `allow-fragmentation` option to enable fragmentation of GRE-encapsulated packets regardless of MTU value.
- `reassemble-packets`: On GRE tunnel interfaces only, you can enable the reassembly of fragmented tunnel packets.
- `key`: You can assign a key value to identify an individual traffic flow within a GRE tunnel, as defined in RFC 2890. However, only one key is allowed for each tunnel source and destination pair. Each IPv4 packet entering the tunnel is encapsulated with the GRE tunnel key value. Each IPv4 packet exiting the tunnel is verified by the GRE tunnel key value and de-encapsulated. Packets that do not match the configured key value are dropped by the router. The key number can be 0 through 4,294,967,295. You must configure the same GRE tunnel key value on both tunnel endpoints.
- `clear-dont-fragment-bit`: By default, IPv4 traffic transmitted over GRE tunnels is not fragmented. To enable fragmentation of IPv4 packets in GRE tunnels, include the `clear-dont-fragment-bit` statement. This option clears the DF bit on all packets—even packets that do not exceed the tunnel MTU. If the packet's size exceeds the tunnel's MTU value, the packet is fragmented before encapsulation. If the packet's size does not exceed the tunnel's MTU value, the packet is not fragmented.

In addition to the tunneling configuration options previously shown, you can also enable or disable path MTU discovery for GRE and IP-IP tunnels under the `[edit system internet-options]` hierarchy level, as follows:

```
[edit system internet-options]
user@R1# set ?
Possible completions:
...
gre-path-mtu-discovery  Enable path MTU discovery for GRE tunnels
```

```

ipip-path-mtu-discovery    Enable path MTU discovery for IP-IP tunnels
no-gre-path-mtu-discovery  Don't enable path MTU discovery for GRE tunnels
no-ipip-path-mtu-discovery Don't enable path MTU discovery for IP-IP tunnels
...

```

Path MTU discovery (PMTUD) is a technique used to determine the MTU size on a path between two endpoints so that IP fragmentation does not occur. PMTUD sets the DF bit in the IP header of outgoing packets. If an intermediary device on the path has an MTU smaller than the packet sent by the tunnel endpoint, that device drops the packet and sends back an ICMP *Fragmentation Needed* (type 3, Code 4) message containing its MTU. The received message allows the tunnel endpoint to reduce its path MTU. The tunnel endpoint repeats this discovery process until the MTU is small enough that packets can pass through the entire path without fragmentation.

Note that the configuration options for GRE and IP-IP tunnels can vary, and in some cases, might be platform specific. Refer to the technical publications for your specific product for support information.

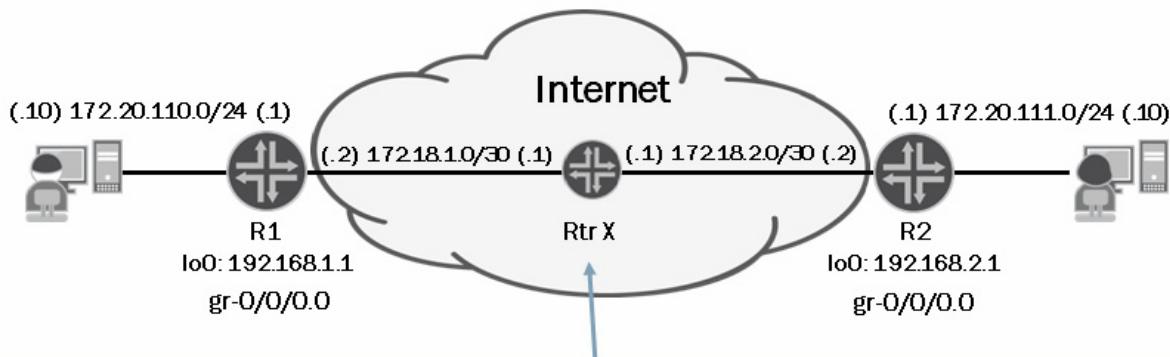
Case Study: Defining the Required Routes

```

[edit]
user@R1# show routing-options static
route 192.168.2.1/32 next-hop 172.18.1.1;
route 172.20.111.0/24 next-hop gr-0/0/0.0;

[edit]
user@R2# show routing-options static
route 192.168.1.1/32 next-hop 172.18.2.1;
route 172.20.110.0/24 next-hop gr-0/0/0.0;

```



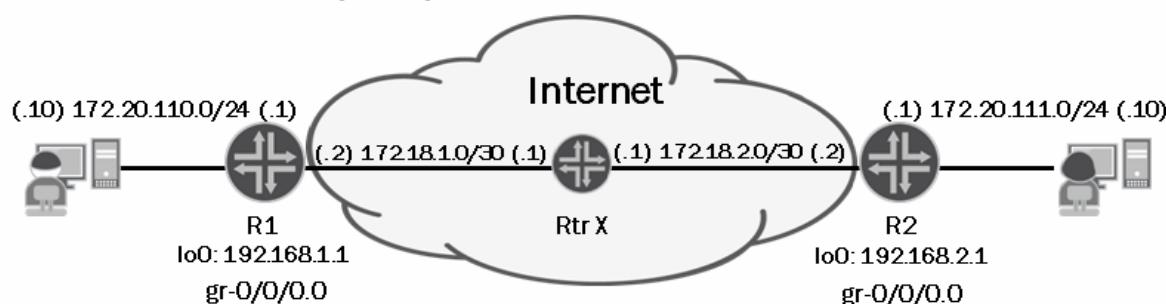
Note: Remember that all intermediary routers must have a route to the loopback addresses.

The graphic illustrates the required routes on R1 and R2. As previously mentioned, each tunnel endpoint must have a valid route to the remote tunnel endpoint. This route must resolve to the physical next hop in the path toward the remote tunnel endpoint and cannot use the tunnel as the next hop. The tunnel endpoints also need a route to direct traffic destined to the remote subnet in to the tunnel. In addition to the routing requirements for the tunnel endpoints, the intermediary devices must be able to route to each of the tunnel endpoints, which often (as in this example) is the loopback address assigned to the tunnel endpoints.

Case Study: Verifying Operations—Part 1

```
user@R1> show interfaces gr-0/0/0 terse
Interface          Admin Link Proto  Local                  Remote
gr-0/0/0           up    up      inet

user@R2> show interfaces gr-0/0/0 terse
Interface          Admin Link Proto  Local                  Remote
gr-0/0/0           up    up      inet
```



The graphic illustrates the first step for verifying proper operation of a GRE or IP-IP tunnel. As shown in the sample output on the graphic, the gr-0/0/0.0 interface is up on both tunnel endpoints. Because GRE and IP-IP tunnels are stateless, you should always check both sides to ensure that the tunnel interfaces are up. In addition to checking the status of the tunnel interfaces, you should also perform the remainder of the verification steps outlined on the subsequent pages in this section.

Case Study: Verifying Operations—Part 2

```
user@R1> show route 192.168.2.1

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

192.168.2.1/32      *[Static/5] 01:19:00
                     > to 172.18.1.1 via ge-0/0/3.0

user@R1> show route 172.20.111.0/24

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

172.20.111.0/24     *[Static/5] 01:17:44
                     > via gr-0/0/0.0
```

The inset illustrates the next step for verifying proper tunnel operations. We list the required routes for the R1 device in the CLI. The following is a similar output for R2:

```
user@R2> show route 192.168.1.1

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

192.168.1.1/32      *[Static/5] 5d 21:02:26
                     > to 172.18.2.1 via ge-0/0/3.0

user@R2> show route 172.20.110.0/24

inet.0: 19 destinations, 19 routes (19 active, 0 holddown, 0 hidden)
Restart Complete
```

+ = Active Route, - = Last Active, * = Both

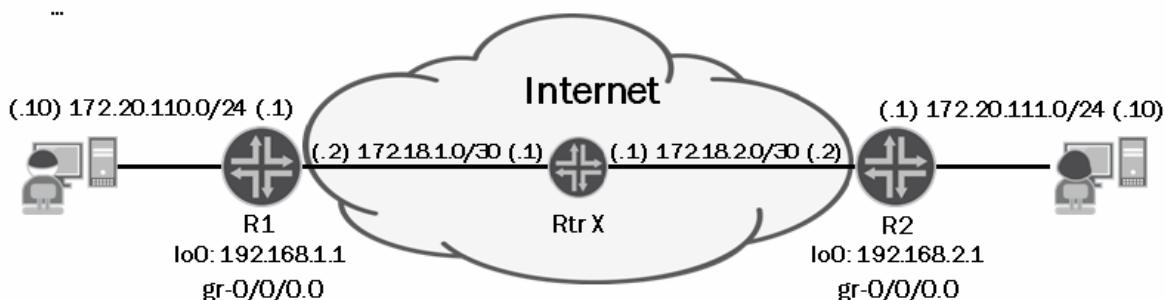
```
172.20.110.0/24      *[Static/5] 10:43:01
                      > via gr-0/0/0.0
```

Verifying Operations: Part 3

```
user@R1> ping 172.20.111.10 rapid count 25 source 172.20.110.1
PING 172.20.111.10 (172.20.111.10): 56 data bytes
!!!!!!!--- 172.20.111.10 ping statistics --- 25 packets transmitted, 25 packets received, 0% packet loss round-trip min/avg/max/stddev = 1.492/3.185/8.064/2.521 ms
```

```
user@R1> show interfaces gr-0/0/0.0 detail | find "traffic statistics"
Traffic statistics:
  Input  packets:          25
  Output  packets:          25
  ...

```



The graphic illustrates the third and final recommended step for verifying tunnel operations. We recommend you send traffic through the tunnel and confirm that the interface statistics increment accordingly on the respective tunnel interface. In the example, we send traffic from R1 to a known user device on the remote subnet (172.20.111.10). Here, we use the `source` option with the `ping` operation to ensure that the return traffic also uses the GRE tunnel. Based on the interface counters, we see that the sent and received traffic both use the GRE tunnel as expected.

Review Questions

1. What are some common reasons to use IP tunnels?
2. Name some differences between GRE and IP-IP.
3. List the key requirements for GRE and IP-IP tunnels.
4. Why should the route for the remote tunnel endpoint be specific and use a low route preference?

Answers

1.

Although there might be many reasons to use IP tunnels, we highlighted a few common scenarios in which they are used. IP tunnels are used to carry traffic that otherwise is not routable over a public IP network like the Internet. This traffic could include IPX, AppleTalk, or IP traffic that uses RFC 1918 addressing. You can also use IP tunnels as a backup link in case a failure occurs.

2.

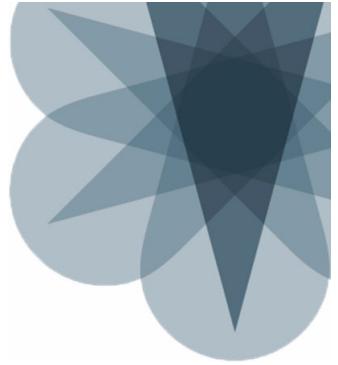
Some differences between GRE and IP-IP include the encapsulation format used by each tunneling protocol, and the types of protocols each protocol can encapsulate. IP-IP adds an additional IP header to the payload packet, whereas GRE adds a GRE header as well as an outer IP header. IP-IP encapsulates only IP packets, whereas GRE can encapsulate a number of Network Layer protocols, such as IPX, AppleTalk, and IP.

3.

Both IP-IP and GRE tunnels require tunnel interfaces and an end-to-end communications path with the required routing information on all participating routers.

4.

The route to the remote tunnel endpoint should be specific and use a low route preference to ensure tunnel stability. If a more preferred route to the remote tunnel endpoint is received over the tunnel, the tunnel interface goes down.



JNCIS-SP Study Guide—Part 1

Chapter 6: High Availability

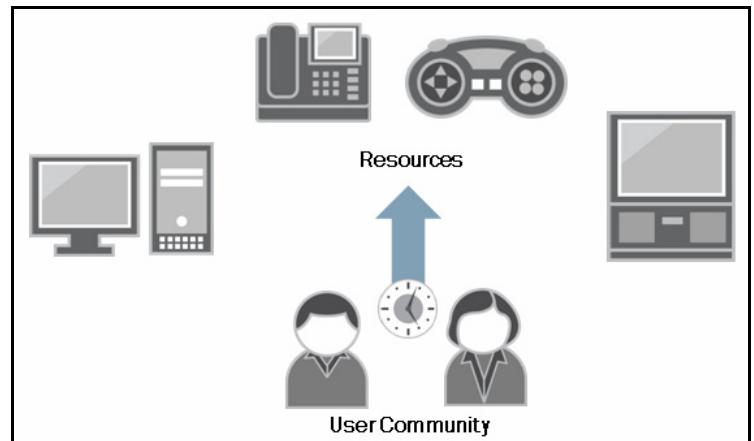
This Chapter Discusses:

- Characteristics of high availability (HA) networks;
- Some high availability features; and
- Configuration and monitoring of some high availability features.

High Availability Defined

Users want their systems, (for example, computers, telephones, video games, or televisions) to work properly and at all times. Availability, in general terms, refers to the ability of a user group to access a desired system or resource. When the user group cannot access a given system or resource, the resource is considered unavailable. Often, the term *downtime* is used to describe a period when a system or resource is unavailable. Therefore, high availability is the ability to ensure a high degree of operational continuity for a given user community to a system or some other resource.

Note that uptime and availability are not the same thing. A system can be up, but unavailable because of other issues. Availability is typically measured as a percentage of uptime over a given duration. The following table provides a mapping for availability percentages and the corresponding amount of time a system is considered to be unavailable:

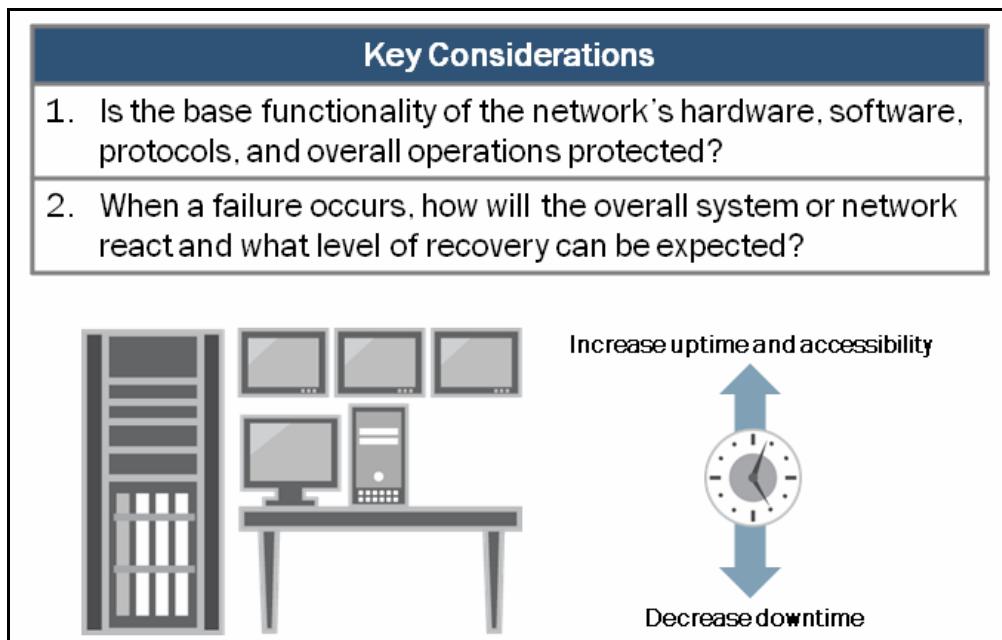


Availability and Downtime Mappings

Availability %	Downtime per year	Downtime per month (30 days)	Downtime per week
90%	36.5 days	72 hours	16.8 hours
99%	3.65 days	7.20 hours	1.68 hours
99.9%	8.76 hours	43.2 minutes	10.1 minutes
99.99%	52.6 minutes	4.32 minutes	1.01 minutes
99.999%	5.26 minutes	25.9 seconds	6.05 seconds
99.9999%	31.5 seconds	2.59 seconds	0.605 seconds

The obvious objective is to achieve the highest level of availability possible. Note that downtime calculations can vary between organizations, which can make them somewhat misleading. Because the manner in which downtime is calculated can vary, you might find that overall user satisfaction is a better method of evaluating success.

High Availability Networks



When designing high availability networks, you should ensure that all network components function properly and are available to their respective user community. As previously mentioned, there is a difference between uptime and availability, but both are equally important and must coexist for full operational continuity. All high availability networks include provisions in the form of features and physical characteristics that allow for maximum uptime and accessibility.

To maximize uptime and accessibility in a network, you should consider the following as you design and implement your network:

1. Is the base functionality of the network's hardware, software, protocols, and overall operations protected?
2. When a failure occurs, how will the overall system or network react, and what level of recovery can be expected?

To properly protect a network, your network design should include some level of redundancy. Although redundancy can provide a large amount of protection for the network, it does come with a cost. Many devices running the Junos operating system include redundant hardware components, such as Routing Engines (REs), control boards (CBs), power supplies, and cooling fans. Refer to the technical publications at <http://www.juniper.net/techpubs/> for details on a specific Junos device.

In addition to redundant hardware, you can also use various software features that accommodate redundancy and rapid failure detection. We cover some of these high availability features in the next section.

Supported High Availability Features

The Junos OS supports a number of software features that can increase availability in a network. Some high availability features compliment other features, whereas in other cases, they are mutually exclusive and cannot be enabled together.

The following is a brief summary of the high availability features shown in the table:

- **Graceful restart (GR):** This feature allows uninterrupted packet forwarding and temporary suppression of all routing protocol updates. GR enables a router to pass through intermediate convergence states that are hidden from the rest of the network.
- **Graceful Routing Engine switchover (GRES):** This feature enables a routing platform with redundant REs to continue forwarding packets even if one RE fails. Graceful RE switchover preserves interface and kernel information and ensures that traffic is not interrupted. Graceful RE switchover does not, however, preserve the control plane.

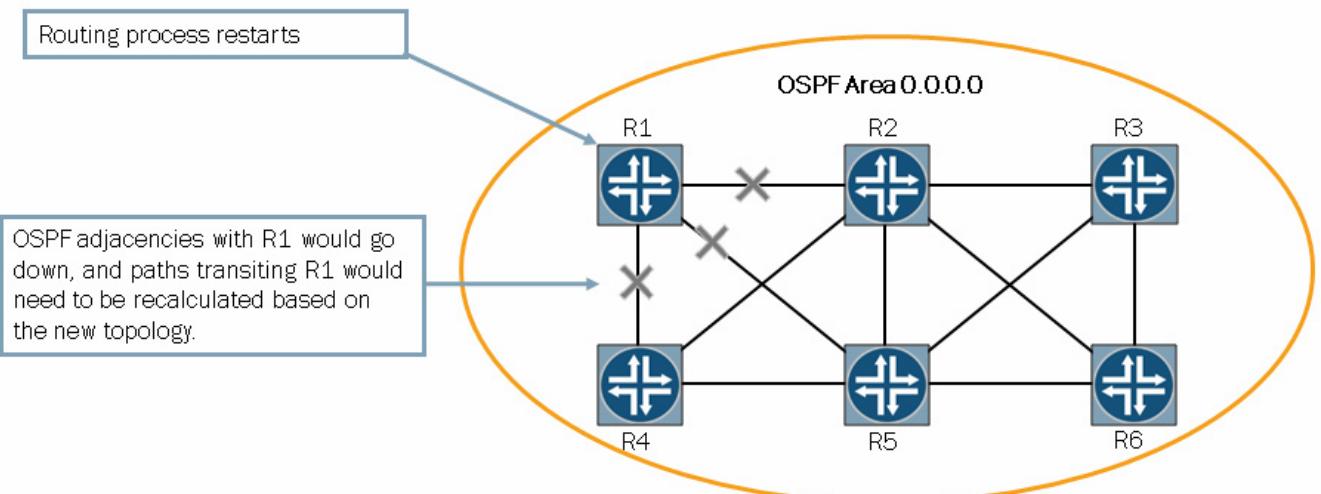
High Availability Features
Graceful restart (GR)
Graceful Routing Engine Switchover (GRES)
Nonstop routing (NSR)
Bidirectional Forwarding Detection (BFD)
Virtual Router Redundancy Protocol (VRRP)
Unified In-Service Software Upgrade (ISSU)

- **Nonstop active routing (NSR):** This feature uses the same infrastructure as graceful RE switchover to preserve interface and kernel information. However, NSR also saves routing protocol information by running the routing protocol process (rpd) on the backup RE. By saving this additional information, NSR is self contained and does not rely on helper routers to assist the routing platform in restoring routing protocol information. NSR is advantageous in networks where neighbor routers do not support GR. As a result of this enhanced functionality, NSR is a natural replacement for GR. NSR and GR are mutually exclusive and cannot be enabled at the same time. Note that graceful RE switchover must be configured for NSR to function properly.
- **Bidirectional Forwarding Detection (BFD):** This feature is a simple hello mechanism that detects failures in a network. BFD sends hello packets at a specified, regular interval. A neighbor failure is detected when the routing device stops receiving a reply after a specified interval. BFD works with a wide variety of network environments and topologies. The failure detection timers for BFD have shorter time limits than default failure detection mechanisms, providing faster detection.
- **Virtual Router Redundancy Protocol (VRRP):** This feature enables hosts on a LAN to make use of redundant routing platforms on that LAN without requiring more than the static configuration of a single default route on the hosts. The VRRP routing platforms share the IP address corresponding to the default route configured on the hosts. At any time, one of the VRRP routing platforms is the master (active) and the others are backups. If the master fails, one of the backup routers becomes the new master router, providing a virtual default routing platform and enabling traffic on the LAN to be routed without relying on a single routing platform.
- **Unified In-Service Software Upgrade (ISSU):** This feature allows you to upgrade between two different Junos OS releases with no disruption on the control plane and with minimal disruption of traffic. Unified ISSU is supported only on dual Routing Engine platforms. In addition, graceful Routing Engine switchover (GRES) and nonstop active routing (NSR) must be enabled.

Note that support for these highlighted features can vary between Junos devices. Refer to the documentation for your specific product for details. We cover the highlighted features in more detail in the subsequent sections. The Junos OS supports several other HA features and supporting technologies not shown in the table. You can refer to the technical publications at <http://www.juniper.net/techpubs/> for more details.

What If ...?

- Given the sample topology, what would happen if R1's routing process (rpd) restarted?

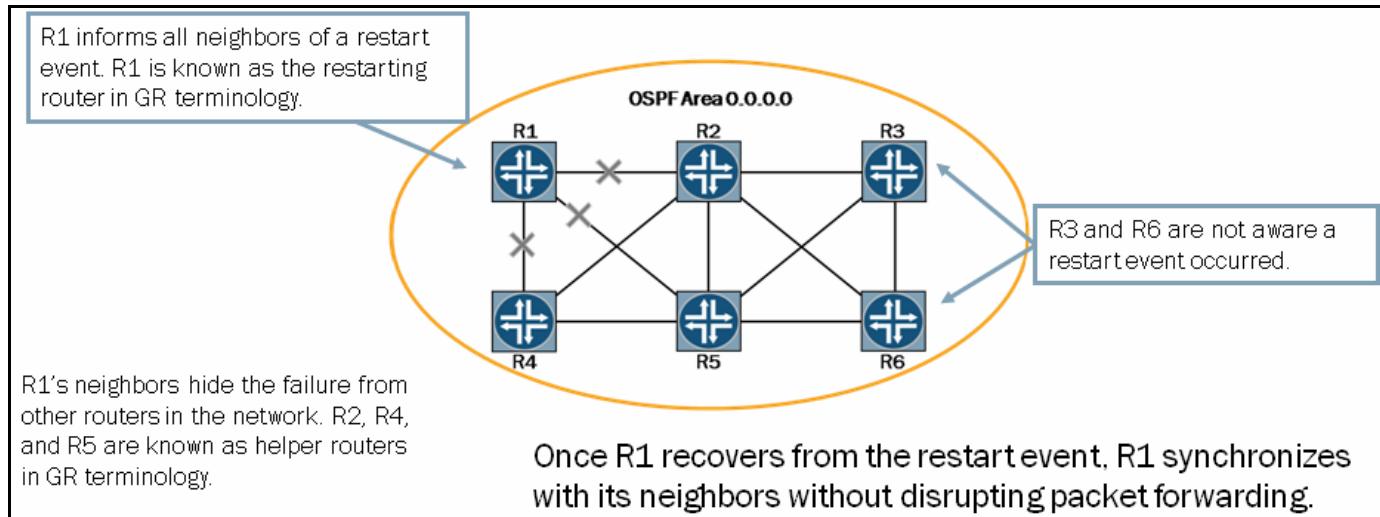


Note: Once the routing process has restarted and OSPF adjacencies are reformed, a new calculation occurs on all routers for the affected routes, causing a second disruption.

There are a number of possible events that can cause network disruptions. One such event is presented in the graphic, where R1's routing process (rpd) restarts. When the routing process restarts, all configured protocols are affected, which means the established network topology and communication paths are also affected. In the scenario presented in the graphic, all of R1's OSPF neighbors (and in fact all OSPF routers in this case) must recalculate any path that traverses R1 because of the topology change.

Typically, when rpd restarts, the effect on the network is temporary. In other words, once rpd restarts and the affected protocols re-establish their respective adjacencies or peering sessions, the topology and data paths return to their original state, thus causing multiple, networkwide disruptions—one when the initial topology change occurs because of the rpd restart event, and one when the topology is restored to its original form. These temporary disruptions can have a significant impact on a user's experience, especially considering today's modern networks, which include voice and video communications.

Introducing Graceful Restart



Graceful restart (GR) addresses the situation described on the previous page. GR allows a router undergoing a restart event, including a restart of the routing protocol process (rpd), to inform its adjacent neighbors and peers of its condition. The restarting router requests a grace period from the neighbor or peer, which can then cooperate with the restarting router. When a restart event occurs and GR is enabled, the restarting router can still forward traffic during the restart period, and convergence in the network is not disrupted. The neighbors or peers of the restarting router, also known as helper routers, hide the restart event from other devices not directly connected to the restarting router. In other words, the restart is not visible to the rest of the network, and the restarting router is not removed from the network topology.

The graceful restart request occurs only if the following conditions are met:

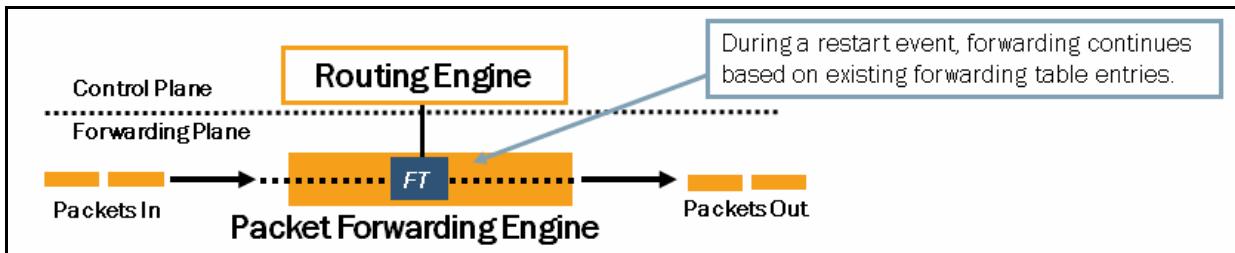
- The network topology is stable;
- The neighbor or peer cooperates;
- The restarting router is not already cooperating with another restart already in progress; and
- The grace period does not expire.

GR Support

Examples of Protocols That Support GR			
OSPF	IS-IS	BGP	RIP
RSVP	LDP	MSDP	PIM

As shown in the graphic, GR is supported by several standards-based protocols. A number of RFCs and drafts exist that document the operational details for GR and each of the protocols for which GR is supported. While these different protocols implement GR slightly differently, the basic concepts and operations are the same from a high availability point of view.

GR Requirements



Routers must have GR enabled to support both GR router modes—the restarting router mode and helper router mode. By default, Junos devices can operate as helper routers but not as restarting routers; restarting router mode functionality must be enabled through configuration. We cover GR configuration on subsequent slides.

In addition to having the GR functionality enabled, the router must support nonstop forwarding operations, which simply means the router must be able to continue forwarding traffic during times of control plane instability. Nonstop forwarding is an inherent attribute of Junos devices because of the architectural design, which cleanly separates the control and forwarding planes.

Configuring GR: Part 1

```
[edit]
user@R1# show routing-options
graceful-restart {
    disable;
}

[edit]
user@R1# show protocols bgp
graceful-restart;
group my-group {
    type internal;
    local-address 192.168.1.1;
    neighbor 192.168.1.2;
    neighbor 192.168.2.2 {
        graceful-restart {
            disable;
        }
    }
}
```

Disables helper mode globally for all protocols that support GR

Enables helper mode for BGP

Note: The most specific application is preferred.

Disables GR for BGP peer

GR helper mode is enabled by default on all Junos devices. You can disable GR helper mode globally for all supported protocols at the [edit routing-options] hierarchy or on a per-protocol, per-group, or per-neighbor basis, depending on the specific protocol. The inset illustrates the syntax required to disable GR helper mode globally, enable GR helper mode for the BGP protocol, and disable GR for a BGP peer. As with many similar configuration scenarios, the most specific definition is used.

Configuring GR: Part 2

```
[edit]
user@R1# show routing-options
graceful-restart;

[edit]
user@R1# show protocols bgp
graceful-restart;
group my-group {
    type internal;
    local-address 192.168.1.1;
    neighbor 192.168.1.2;
    neighbor 192.168.2.2 {
        graceful-restart {
            disable;
        }
    }
}
```

Enables restarting router mode for all protocols that support GR

Disables GR for BGP peer

GR's restarting router mode is not enabled by default. You can enable GR restarting router mode through configuration at the [edit routing-options] hierarchy. The inset provides a sample configuration used to enable GR's restarting router mode globally and for all protocols along with a sample configuration that disables GR for a specific BGP peer.

The configuration options used with GR vary between the supported protocols. The following are the available GR configuration options for OSPF and BGP:

```
[edit protocols]
user@R1# set ospf graceful-restart ?
Possible completions:
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except   Don't inherit configuration data from these groups
 disable                Disable OSPF graceful restart capability
 helper-disable          Disable graceful restart helper capability
 no-strict-lsa-checking Do not abort graceful helper mode upon LSA changes
 notify-duration         Time to send all max-aged grace LSAs (1..3600 seconds)
 restart-duration        Time for all neighbors to become full (1..3600 seconds)
```

```
[edit protocols]
user@R1# set bgp graceful-restart ?
Possible completions:
 <[Enter]>                  Execute this command
 + apply-groups              Groups from which to inherit configuration data
 + apply-groups-except       Don't inherit configuration data from these groups
 disable                     Disable graceful restart
 restart-time               Restart time used when negotiating with a peer (1..600)
 stale-routes-time          Maximum time for which stale routes are kept (1..600)
 |                           Pipe through a command
```

For other protocol-specific GR configuration options, refer to the technical publications.

Monitoring GR

```
user@R1> show bgp neighbor peer-address
...
Options: <Preference LocalAddress GracefulRestart Refresh>
...
NLRI for restart configured on peer: inet-unicast
NLRI advertised by peer: inet-unicast
NLRI for this session: inet-unicast
Peer supports Refresh capability (2)
Restart time configured on the peer: 120
Stale routes from peer are kept for: 300
Restart time requested by this peer: 120
NLRI that peer supports restart for: inet-unicast
NLRI that restart is negotiated for: inet-unicast
...
```

Indicates restarting router mode is enabled

- For OSPF, enable traceoptions with the **graceful-restart** flag option and monitor the associated log file

```
[edit protocols ospf]
user@R1# show traceoptions
file trace-ospf;
flag graceful-restart;
```



```
user@R1> show log trace-ospf
```

You monitor GR differently depending on the protocol with which it is configured. For BGP, you use the **show bgp neighbor** command along with the peer address. If you have GR restarting router mode enabled under the [edit routing-options] hierarchy, the GracefulRestart option appears in the output generated by the **show bgp neighbor** command; otherwise this option is omitted from the options list. The other references of restart shown in the sample output are indicative that this router supports the GR helper router mode, which is enabled by default.

For OSPF, and other protocols such as IS-IS, you monitor GR operations using traceoptions. You must first enable traceoptions using the **graceful-restart** flag option. Then you monitor the associated log file using the **show log log-file-name** command. The following output illustrates a sample GR transaction from the perspective of the restarting router:

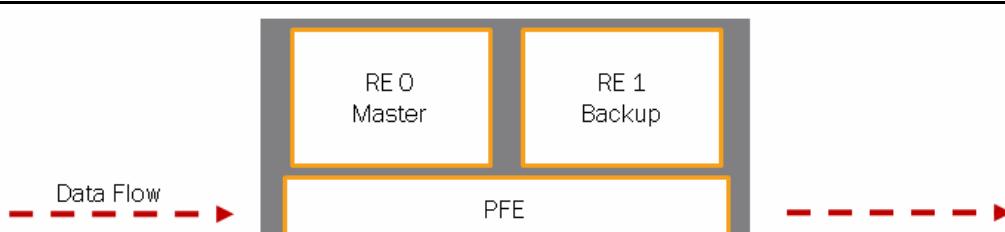
```
user@R1> show log trace-ospf
Jan 15 10:08:53.652803 OSPF Restart: phase now 2
Jan 15 10:08:55.300166 OSPF Restart: sending grace lsas
Jan 15 10:08:55.301066 OSPF Restart: estimated restart duration timer triggered
Jan 15 10:08:55.301139 OSPF Restart: area 0.0.0.0 triggered restart maxwait timer of 40
seconds
Jan 15 10:08:56.312242 OSPF Restart: sending more grace lsas
Jan 15 10:08:57.312678 OSPF Restart: sending more grace lsas
Jan 15 10:09:01.320373 OSPF Restart: graceful restart OK to send hellos
Jan 15 10:09:01.320486 OSPF Restart: phase now 3
Jan 15 10:09:01.336244 RPD OSPF_NBRUP: OSPF neighbor 172.20.77.2 (realm ospf-v2 ge-0/0
1.0 area 0.0.0.0) state changed from Init to 2Way due to 2WayRcvd (event reason:
neighbor detected this router)
Jan 15 10:09:35.308470 OSPF Restart: area 0.0.0.0 restart maxwait timeout
Jan 15 10:09:37.703251 OSPF Restart: area 0.0.0.0 building the router lnk nbr tree
Jan 15 10:09:37.703554 OSPF Restart: lnk_nbr_tree link type 2 id 172.20.77.2 data
172.20.77.1, numbered 1
Jan 15 10:09:37.703800 OSPF Restart: build intf_lnk_nbr tree interface ge-0/0/1.0 area
0.0.0.0
Jan 15 10:09:37.703859 OSPF Restart: lnk_nbr_tree link type 2 id 192.168.2.1 data
172.20.77.1, numbered 1
Jan 15 10:09:37.756306 RPD OSPF_NBRUP: OSPF neighbor 172.20.77.2 (realm ospf-v2 ge-0/0
1.0 area 0.0.0.0) state changed from Loading to Full due to LoadDone (event reason:
OSPF loading completed)
Jan 15 10:09:37.756403 OSPF Restart: remove neighbor id 192.168.2.1 interface ge-0/0/
1.0 area 0.0.0.0 from interface lnk_nbr_tree
```

```

Jan 15 10:09:37.756465 OSPF Restart: removed neighbor 172.20.77.2 id 192.168.2.1
interface ge-0/0/1.0 area 0.0.0.0 from area lnk_nbr_tree
Jan 15 10:09:37.756540 OSPF Restart: lnk_nbr_tree is empty for area 0.0.0.0
Jan 15 10:09:37.756588 OSPF Restart: phase now 4
Jan 15 10:09:37.756731 OSPF Restart: all neighbors acquired. Notifying all that TED
database is populated
Jan 15 10:09:37.756779 OSPF Restart: exit imminent
Jan 15 10:09:47.793197 OSPF Restart: the restart estimated duration is up
Jan 15 10:09:47.793330 OSPF Restart: phase now 5
Jan 15 10:09:47.793569 OSPF Restart: purging grace lsas
Jan 15 10:09:47.997263 OSPF Restart: purging old LSAs
Jan 15 10:09:47.997361 OSPF Restart: phase now 0
Jan 15 10:09:47.997791 OSPF Restart: phase 0 restart complete

```

Graceful RE Switchover Defined



When graceful Routing Engine switchover is enabled, interface and kernel information is preserved so traffic forwarding is not interrupted.

An Operational Comparison

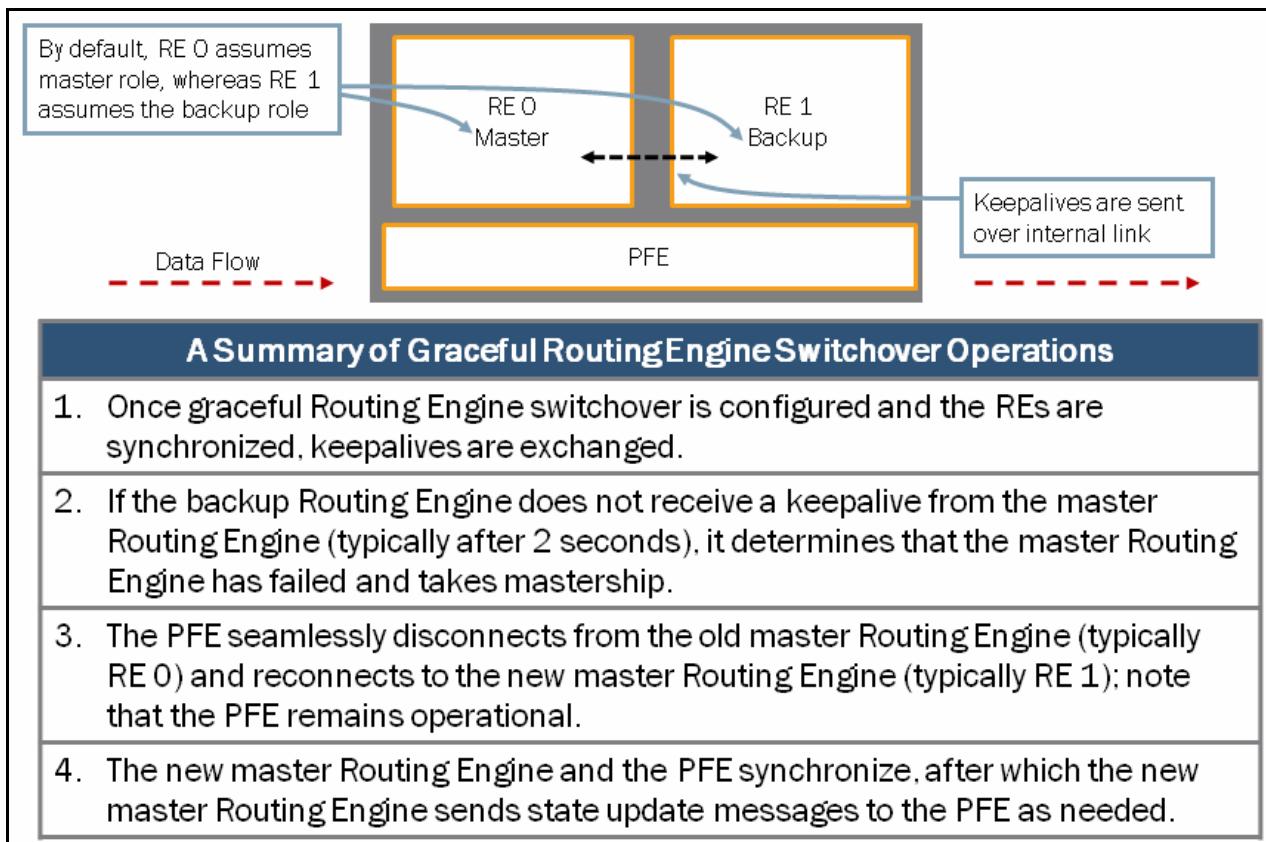
An Operational Comparison	
Without Graceful Routing Engine Switchover	PFE restarts and all hardware and interfaces are discovered by the new master Routing Engine. The new master Routing Engine restarts rpd, and all adjacencies are aware of the physical (interface alarms) and routing changes.
With Graceful Routing Engine Switchover	PFE is not restarted and interface and kernel information is preserved, so the switchover is faster. The new master Routing Engine restarts rpd, and all adjacencies are aware of the routing change.

Many Junos devices offer hardware component redundancy, such as redundant REs. Graceful RE switchover enables a routing platform with redundant REs to continue forwarding packets, even if one RE fails. Graceful RE switchover preserves interface and kernel information and ensures that traffic forwarding is not interrupted during a mastership change. Graceful RE switchover does not, however, preserve the control plane, which means the routing protocol process (rpd) must restart and the information learned through that process must be relearned (unless NSR is also configured). The graphic shows a brief comparison that highlights the benefits of using graceful RE switchover.

Without graceful RE switchover enabled, the PFE restarts and all hardware and interfaces are discovered by the new master RE when a mastership change occurs. The new master RE restarts rpd, so all adjacencies are aware of the topological changes. Because all interfaces go down during this process, the system generates interface alarms. The network also undergoes a topology change because all protocol adjacencies are affected, albeit temporarily.

With graceful RE switchover enabled, the PFE is not restarted and interface and kernel information is preserved. By allowing the PFE to remain up during a mastership switchover and preserving interface and kernel information, graceful RE switchover greatly reduces the time the RE failover process takes. With no other high availability features enabled, the new master RE must restart rpd, so all adjacencies are aware of the routing change. To preserve routing during a switchover, graceful RE switchover must be combined with either GR or NSR. We discuss NSR in the next section of this chapter.

Graceful RE Switchover Operations



By default, when two REs are installed in a router, the RE installed in slot 0 (known as RE0) assumes the master RE role, whereas the RE installed in slot 1 (known as RE1) assumes the backup RE role. The following sample output illustrates this concept:

```
{master}
user@R1-re0> show chassis routing-engine
Routing Engine status:
Slot 0:
  Current state           Master
  Election priority       Master (default)
...
Routing Engine status:
Slot 1:
  Current state           Backup
  Election priority       Backup (default)
...
```

To allow some distinction between both REs and ensure that both REs are accessible regardless of their current role, you can define and apply RE groups, as shown in the following sample capture:

```
{master} [edit groups]
user@R1-re0# show
re1 {
  system {
    host-name R1-re1;
    backup-router 172.18.66.1;
  }
  interfaces {
    fxp0 {
      unit 0 {
        family inet {
          address 172.18.66.51/24;
```

```

        }
    }
}
re0 {
    system {
        host-name R1-re0;
        backup-router 172.18.66.1;
    }
    interfaces {
        fxp0 {
            unit 0 {
                family inet {
                    address 172.18.66.50/24;
                }
            }
        }
    }
}

```

You apply the **re0** and **re1** groups at the root hierarchy level:

```
{master} [edit]
user@R1-re0# set apply-groups [re0 re1]
```

The previous sample configuration ensures each RE can be distinguished by a unique name and gives each RE its own management IP address for direct out-of-band (OoB) management access. By default, the backup RE does not have rpd running, which means it does not have any routes installed. You can install a route entry (in this case a default route) on the backup RE using the **backup-router** statement. The specified address represents the gateway address for this route entry. Note that the **backup-router** statement can alternatively be included under the [edit system] hierarchy level as a single entry and its definition is not required but highly recommended.

Once graceful RE switchover is enabled, you should synchronize the configurations using the **commit synchronize** command. If you do not synchronize configurations, the configuration applied on the backup RE will be used when a failover occurs. If graceful RE switchover is enabled and you do not issue **commit synchronize**, you will see the following commit warning:

```
{master} [edit chassis]
user@R1-re0# commit
warning: graceful-switchover is enabled, commit synchronize should be used
commit complete
```

You can, alternatively, add the synchronize functionality through the configuration. With this functionality enabled, issue **commit** to synchronize configurations, as follows:

```
{master} [edit system]  
user@R1-re0# set commit synchronize
```

```
{master} [edit system]
user@R1-re0# commit
re0:
configuration check succeeds
re1:
commit complete
re0:
commit complete
```

Note that in some virtualization implementations, the commit synchronize operation is automatic. Check the documentation for your specific platform for details.

Once the REs are synchronized, they exchange keepalives. If the backup RE does not receive a keepalive from the master RE after a specified timeout (typically 2 seconds), it determines that the master RE has failed and takes mastership. When a mastership change occurs, the PFE seamlessly disconnects from the old master RE and reconnects to the new master RE. The

PFE does not reboot and continues forwarding traffic based on the existing forwarding table entries. The new master RE then synchronizes its state with the PFE. If the new master RE detects that the PFE state is not up to date, it resends state update messages.

When graceful RE switchover is enabled, the router alters the information flow from the RE to the PFE. The router now duplicates changes made on the master RE to the backup RE prior to signalling those changes to the PFE. When graceful RE switchover is not enabled, the master RE notifies only the PFE.

If graceful RE switchover is not enabled, we recommend that you implement RE failover protection. You can enable failover protection, adjust the keepalive interval and alter the default mastership assignments under the [edit chassis redundancy] hierarchy. When RE failover protection is enabled, the default keepalive interval is 300 seconds.

```
[edit chassis redundancy]
user@R1# set ?
Possible completions:
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except   Don't inherit configuration data from these groups
> failover              Failover to other Routing Engine
    keepalive-time       Time before Routing Engine failover (2..10000 seconds)
> routing-engine        Redundancy options for Routing Engines
```

If needed, you can perform a manual RE mastership switchover using the **request chassis routing-engine master** commands, as follows:

```
user@R1> request chassis routing-engine master ?
Possible completions:
  acquire                Attempt to become master Routing Engine
  release                Request that other Routing Engine become master
  switch                 Toggle mastership between Routing Engines
```

Redundancy configuration options are platform specific. Check the documentation for your specific product for support details.

Configuring Graceful RE Switchover

```
[edit chassis]
user@R1-re0# set redundancy graceful-switchover

[edit chassis]
user@R1-re0# show
redundancy {
    graceful-switchover;
}

[edit chassis]
user@R1-re0# commit
commit complete
```

Once graceful Routing Engine switchover is activated, a new banner is added to reflect the RE's role

{master} [edit chassis]
user@R1-re0#

{backup} [edit chassis]
user@R1-re1>

You enable graceful RE switchover under the [edit chassis] hierarchy using the **set redundancy graceful-switchover** command. Once you activate the configuration change, the banner should change, indicating the RE's current role (either master or backup) as shown in the inset. In some virtualization implementations, such as the virtual chassis (EX Series) or chassis cluster (SRX Series), the role indicator (either master or backup) is automatically displayed, regardless of whether graceful RE switchover is enabled.

Monitoring Graceful RE Switchover

```

{backup}
user@R1-re1> show system switchover
Graceful switchover: On
Configuration database: Ready
Kernel database: Ready
Peer state: Steady State

```

Graceful Routing Engine switchover is enabled and the databases are synchronized.


```

{master} ←
user@R1-re0> show system s?
Possible completions:
  services          Show service applications information
  snapshot          Show snapshot information
  software          Show loaded JUNOS extensions
  statistics        Show statistics for protocol
  storage           Show local storage data
{master}
user@R1-re0> show system s

```

Note: You cannot verify graceful Routing Engine switchover state details on the master Routing Engine. The required command is only available on the backup Routing Engine.

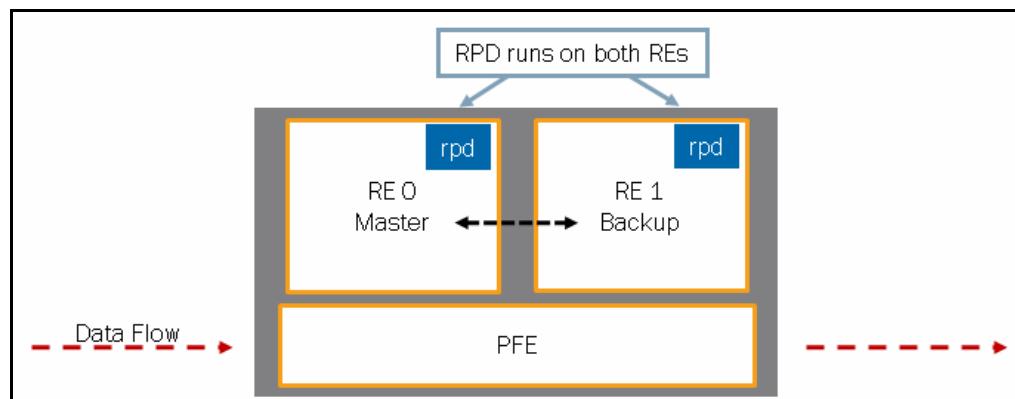
The inset illustrates the **show system switchover** command, which is used to verify whether graceful RE switchover is enabled and that the databases are synchronized. Note that you can issue this command on the backup RE only.

For graceful RE switchover to function properly, the master RE replicates its state to the backup RE and PFE. The following are the three specific states that must be replicated:

- Configuration database:** The configuration database, or repository of configuration files, is replicated through the **commit synchronize** process. Several system processes require the configuration database to perform their designated functions; for example, the device control daemon (dcd) checks this database when it brings interfaces online, the chassis process (chassisid) uses this database to manage hardware components, and rpd uses this database to control routing protocols.
- Kernel and related entries:** Enabling graceful RE switchover starts a custom Junos process known as ksyncd. The ksyncd process is responsible for kernel state replication tasks between various hardware components.
- PFE state:** The Junos OS uses chassisid to perform PFE state replication. When a mastership change occurs, chassisid performs a soft restart to query the system's hardware inventory. When the hardware components respond to the query, the system re-attaches them to the backup RE and brings them online without any disruption.

Nonstop Active Routing

Nonstop Active Routing (NSR) enables a routing platform with redundant REs to switch from a primary RE to a backup RE without alerting peer nodes. NSR uses the same infrastructure as graceful RE switchover to preserve interface and kernel information. In addition to maintaining interface and kernel information, NSR also saves routing protocol information by running the routing protocol



process (rpd) on the backup RE. By saving this additional information, NSR is self-contained and does not rely on helper routers to assist the routing platform in restoring routing protocol information. NSR is advantageous in networks where neighbor routers do not support GR protocol extensions, as well as failure scenarios where GR cannot negotiate a grace period. As a result of this enhanced functionality, NSR is a natural replacement for GR. If you enable both NSR and GR, the commit operation will fail.

NSR requires that the participating REs run the same version of the Junos OS. Although NSR supports most protocols, it does not support all protocols. For all protocols supported by NSR, the state information is preserved during a switchover event. If you configure a protocol that is not supported by NSR, the protocol operates as usual. When a switchover occurs, the state information for the unsupported protocol is not preserved and must be refreshed using the normal recovery mechanisms

inherent in the protocol. Also note that NSR is not supported on all Junos devices. Refer to the technical publications for platform and protocol support details.

Configuring NSR

```
{master} [edit routing-options]
user@R1-re0# set nonstop-routing

{master} [edit routing-options]
user@R1-re0# show
nonstop-routing;

{master} [edit chassis]
user@R1-re0# show
redundancy {
    graceful-switchover;
}
```

The inset provides a sample configuration used to enable NSR. Note that you must also enable graceful RE switchover for NSR to function. We covered graceful RE switchover in detail in the previous section of this chapter. In addition to enabling NSR and graceful RE switchover, you should also ensure that the commit operation synchronizes the configuration file from the master RE to the backup RE. You can issue the **commit synchronize** command each time you perform a commit operation or enable the synchronize functionality in the configuration, as follows:

```
{master} [edit]
user@R1-re0# set system commit ?
Possible completions:
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except   Don't inherit configuration data from these groups
synchronize              Synchronize commit on both Routing Engines by default
```

Once NSR is enabled and the configurations on the master and backup REs are synchronized, the routing protocol process on the backup RE actively gathers information sent to and from the routing protocol process on the master RE. This process allows the backup RE to keep its state up-to-date with the master RE does.

Monitoring NSR

As shown in the graphic, you use the **show task replication** command to verify NSR synchronization. You should see all configured protocols that support NSR synchronization listed in the output, along with their complete status.

Alternatively, you can log in to the backup RE and issue the same operational **show** commands you would issue on the master RE to determine protocol and routing information, as follows:

```
{master}
user@R1-re0> show task replication
      Stateful Replication: Enabled
      RE mode: Master

      Protocol           Synchronization Status
      OSPF               Complete
      BGP                Complete
```

```
{master}
user@R1-re0> request routing-engine login other-routing-engine
€
--- JUNOS 10.1R1.8 built 2010-02-12 18:31:54 UTC
{backup}
user@R1-re1> show ospf neighbor
Address      Interface      State      ID      Pri  Dead
10.1.1.2    fe-0/0/1.0    Full      192.168.100.2  128   0
10.1.2.2    fe-0/0/2.0    Full      192.168.100.3  128   0

{backup}
user@R1-re1> show bgp summary
Groups: 1 Peers: 2 Down peers: 0
Table      Tot Paths  Act Paths Suppressed      History Damp State      Pending
inet.0        10         10          0            0          0          0          0
Peer          AS       InPkt     OutPkt      OutQ   Flaps Last Up/Dwn
State| #Active/Received/Accepted/Damped...
192.168.100.2  64700      55         54          0          0        24:29 5/5/5/0
0/0/0/0
```

192.168.100.3	64700	54	52	0	0	23:53 5/5/5/0
0/0/0/0						

```
{backup}
user@R1-re1> show route protocol ospf

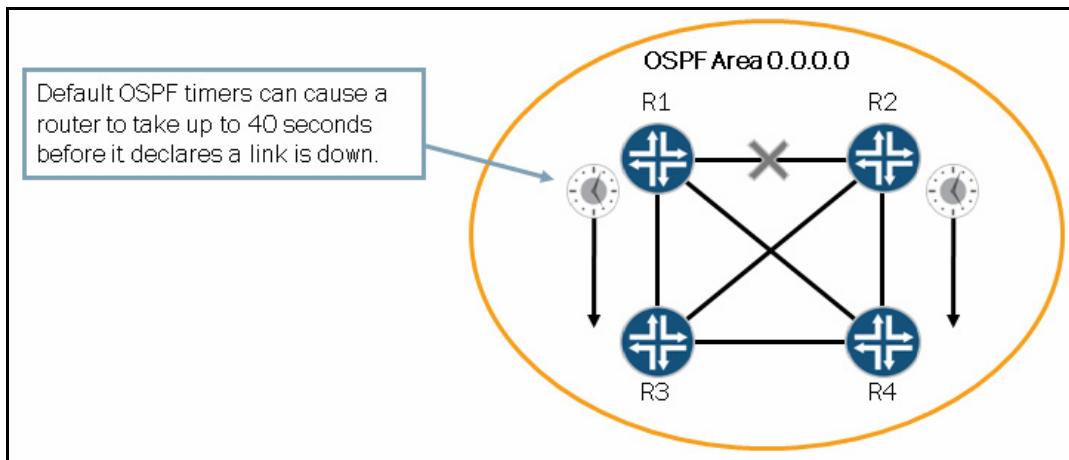
inet.0: 21 destinations, 34 routes (19 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.100.2/32      *[OSPF/10] 00:39:47, metric 1
                      > to 10.1.1.2 via fe-0/0/1.0
192.168.100.3/32      *[OSPF/10] 00:39:47, metric 1
                      > to 10.1.2.2 via fe-0/0/2.0
224.0.0.5/32          *[OSPF/10] 00:44:22, metric 1
                      MultiRecv
```

Note that you can enable traceoptions with the **nsr-synchronization** flag option and monitor the associated log file to view NSR synchronization details for a given protocol.

```
{master} [edit protocols ospf]
user@R1-re0# show
traceoptions {
    file nsr-trace;
    flag nsr-synchronization detail;
}
[Trimmed]
```

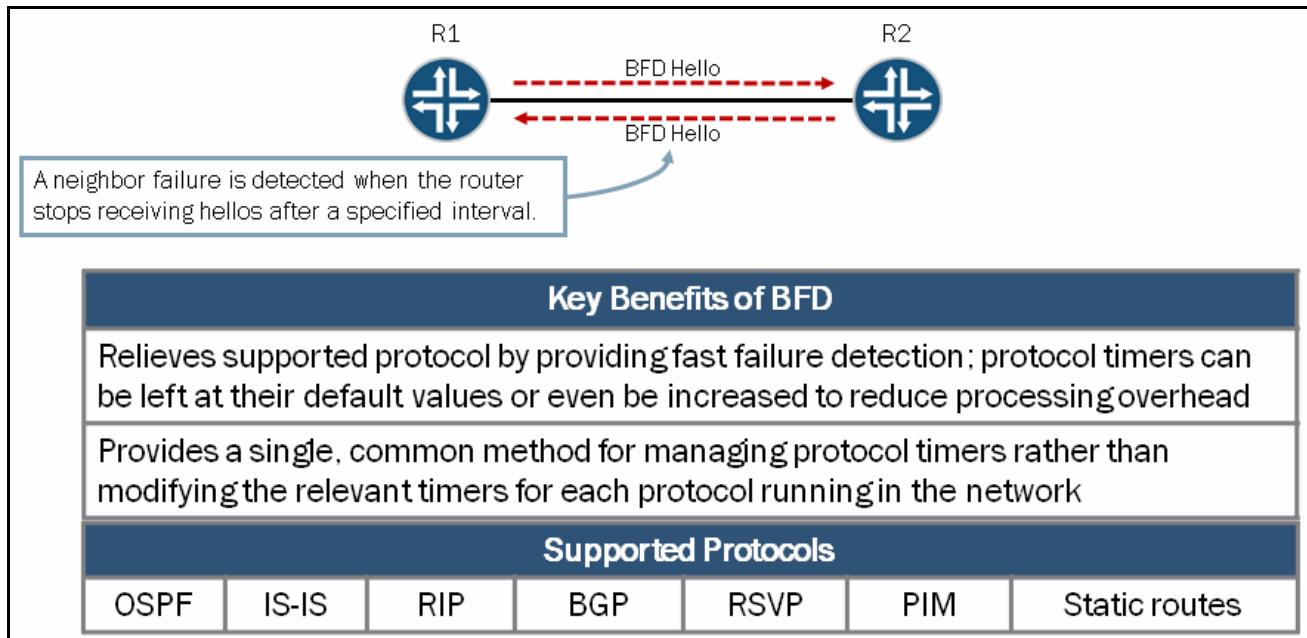
Detecting Network Failures



All modern protocols, such as OSPF and BGP, include some mechanism to detect network failures. One problem with protocol failure detection mechanisms is that they can be slow (especially with the default timers). One example, illustrated in the graphic, is OSPF, which, with its default timers, can take up to 40 seconds before a neighbor is declared dead.

You can adjust the default timers to lower the time it takes a protocol to detect failures and declare a neighbor or peer dead. In fact, you can adjust the dead timers for some interior gateway protocols (IGPs), such as OSPF and IS-IS, to detect failures in about 1 second. There is, however, a cost associated with lowering protocol timers. Lowering a protocol's timers often equates to an increase of hellos or keepalives and, consequently, more processing overhead for the related protocol and routing process. Increasing the load on a protocol or the routing process can potentially cause undesirable results and adversely affect a router's overall performance, especially in large networks.

Bidirectional Forwarding Detection



The Bidirectional Forwarding Detection protocol (BFD) is a simple protocol designed to rapidly detect link failures. Once two devices negotiate and establish a BFD session, BFD continuously sends hellos to monitor the associated link. If BFD stops receiving hellos from its neighbor, it takes down the session and notifies the system that a communication problem exists. BFD can detect link failures in less than a second, which means hellos are exchanged and processed frequently and efficiently.

BFD can provide a number of key benefits within a high availability network. BFD relieves protocols from being required to provide fast-failure detection; in fact, routing protocol timers for hellos or keepalives can be left at their default values or even be increased to reduce the associated processing. BFD also provides a single, common method for managing protocol timers. Rather than modifying the relevant timers for each protocol running in the network, you can leave the protocol timers at their default settings and simply implement BFD to use consistent timer values for all protocols. Another key benefit provided by BFD is that it provides a failure detection mechanism for static routes, which, unlike modern routing protocols, do not have such a mechanism otherwise.

The graphic lists the protocols that support BFD. When a failure occurs, BFD notifies the protocol it is configured to support, at which time the protocol can take the needed action; in most cases, it routes around the detected failure. BFD is configured directly under the protocol or static route you want it to support. We examine the configuration details on a subsequent page.

Configuring BFD

```
[edit protocols]
user@R1# show ospf
area 0.0.0.0 {
    interface ge-0/0/1.0 {
        bfd-liveness-detection {
            minimum-interval 300;
        }
    }
}

[edit protocols]
user@R1# show bgp
group my-ebgp-group {
    bfd-liveness-detection {
        minimum-interval 300;
    }
    neighbor 172.18.1.1 {
        peer-as 65510;
    }
}
```

Minimum configuration requirements

Enables minimum transmit and receive interval for BFD hellos. Note the transmit and receive intervals can be set separately and can use different values.

Enables BFD for EBGP group

We recommend the transmit and receive intervals never be set to a value lower than 300 milliseconds.

The inset highlights some BFD configuration examples and a recommendation for the transmit and receive intervals. You can set the transmit and receive intervals separately or define a `minimum-interval` for both directions, as shown in the inset. Although you can set the minimum interval to a value lower than 300 ms, we do not recommend a value below that interval. With the minimum interval set to 300 ms, BFD still achieves a subsecond failure detection. This subsecond failure detection assumes the default multiplier value of three, which simply means if three consecutive hellos are missed, the link is considered failed ($3 * 300 \text{ ms} = 900 \text{ ms}$ failure detection). On some Junos devices, BFD uses periodic packet management (PPM) on the PFE. PPM off loads the processing overhead typically placed on the RE by delegating some of the processing responsibilities to the PFE. PPM is enabled by default on the Junos devices on which it is supported. For PPM support details, check the product documentation for your specific platform.

Depending on the media type, BFD might not provide a significant benefit. For example, some media types like SONET and ATM already provide rapid detection for link failure, so using BFD to monitor those link types will not add a significant amount of value. BFD does, however, add significant value for Ethernet interfaces, especially when Layer 2 switches are positioned between two devices. Alternatively, you might consider Ethernet OAM, which provides rapid link failure detection for Ethernet connections. Ethernet OAM, unlike BFD, operates at Layer 2. Detailed coverage of Ethernet OAM is outside the scope of this study guide.

By default, BFD sessions are adaptive, which means you can adjust the intervals for the session. You define a minimum interval value for the transmit and receive directions on the router. If the neighbor router's interval for either direction is higher, that BFD session uses the higher value.

The hierarchy at which you enable BFD and the actual configuration options vary between the supported protocols. The configuration options for OSPF and BGP include the following:

```
[edit protocols]
user@R1# set ospf area 0 interface ge-0/0/1.0 bfd-liveness-detection ?
Possible completions:
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except   Don't inherit configuration data from these groups
> authentication         Authentication options
> detection-time        Detection-time options
full-neighbors-only     Setup BFD sessions only to Full neighbors
minimum-interval         Minimum transmit and receive interval (milliseconds)
minimum-receive-interval Minimum receive interval (1..255000 milliseconds)
multiplier               Detection time multiplier (1..255)
no-adaptation            Disable adaptation
> transmit-interval     Transmit-interval options
version                  BFD protocol version number
```

```
[edit protocols]
user@R1# set bgp bfd-liveness-detection ?
Possible completions:
+ apply-groups          Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
> authentication        Authentication options
> detection-time       Detection-time options
  holddown-interval    Time to hold the session-UP notification to the client
  minimum-interval     Minimum transmit and receive interval (milliseconds)
  minimum-receive-interval  Minimum receive interval (1..255000 milliseconds)
  multiplier            Detection time multiplier (1..255)
  no-adaptation        Disable adaptation
> transmit-interval   Transmit-interval options
  version              BFD protocol version number
```

You can define BFD options for BGP at the protocol, group, or neighbor hierarchy levels. As with similar configuration scenarios, the most specific definition is used (neighbor then group, group then protocol). For BFD configuration options for other protocols, refer to the technical publications for the specific protocol.

Monitoring BFD

```
user@R1> show bfd session ?
Possible completions:
<[Enter]>          Execute this command
address             Show BFD session with specific neighbor address
brief               Display brief output (default)
detail              Display detailed output
discriminator       Show BFD session with specific local discriminator
extensive           Display extensive output
prefix              Show all BFD sessions for this LDP FEC
summary             Display summary output
|                  Pipe through a command

user@R1> show bfd session
Address           State      Interface      Detect Time  Transmit Interval Multiplier
172.18.1.1        Up         ge-0/0/3.0  0.900  0.300   3
172.20.77.2       Up         ge-0/0/1.0  1.200  0.400   3

Indicates that the BFD neighbor has its transmit interval set to 400 ms; proof of the default adaptive mode
Default multiplier value; can be adjusted in configuration
```

The inset shows the key command and options for monitoring BFD. In the sample output, the default multiplier value is 3, which means three consecutive BFD hellos must be missed prior to declaring the BFD session dead. You can also see that transmit interval of 400 ms is used for the session with the 172.20.77.2 neighbor, which is a higher value than what was configured. (Refer to the previous page for details.) The higher transmit interval value is indicative of this BFD session operating in adaptive mode, which is the default behavior. You can disable BFD adaptive mode using the **no-adaptation** configuration option.

In addition to the **show bfd session** commands, you can also issue the **show bgp neighbor** command for EBGP neighbors to verify BFD status for a given peer, as follows:

```
user@R1> show bgp neighbor 172.18.1.1
Peer: 172.18.1.1+179 AS 65510 Local: 172.18.1.2+49363 AS 64700
Type: External  State: Established  Flags: <Sync>
Last State: OpenConfirm  Last Event: RecvKeepAlive
Last Error: None
Export: [ adv-aggregates ]
Options: <Preference AdvertiseInactive GracefulRestart PeerAS Refresh>
Options: <BfdEnabled>
Holdtime: 90 Preference: 170
Number of flaps: 0
Peer ID: 10.10.10.10      Local ID: 192.168.1.1      Active Holdtime: 90
```

Keepalive Interval: 30

Peer index: 0

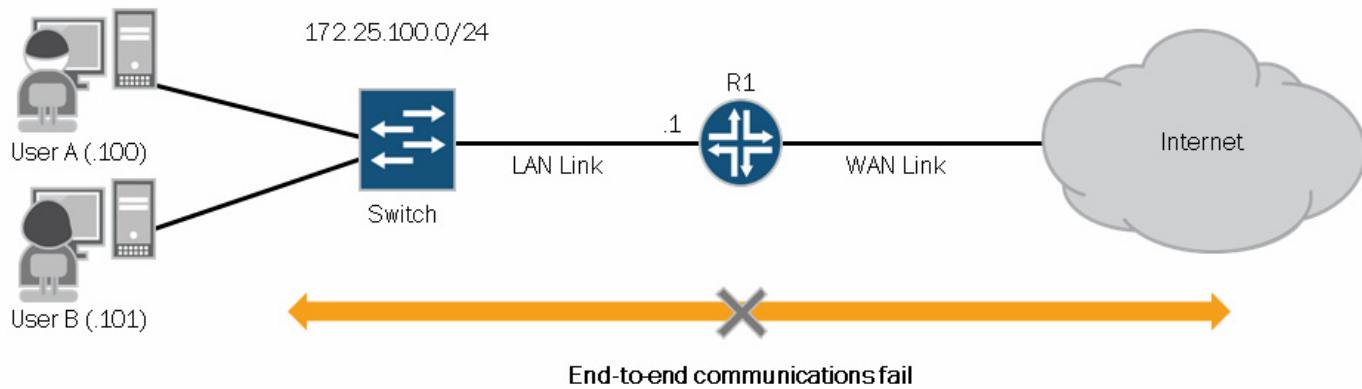
BFD: enabled, up

[Trimmed]

If you do not see the BFD session in the up state, check the configurations on both devices to ensure they are compatible. You might also need to check any configured firewall filters on the Junos devices configured for BFD to ensure that the defined firewall filter permits BFD communications.

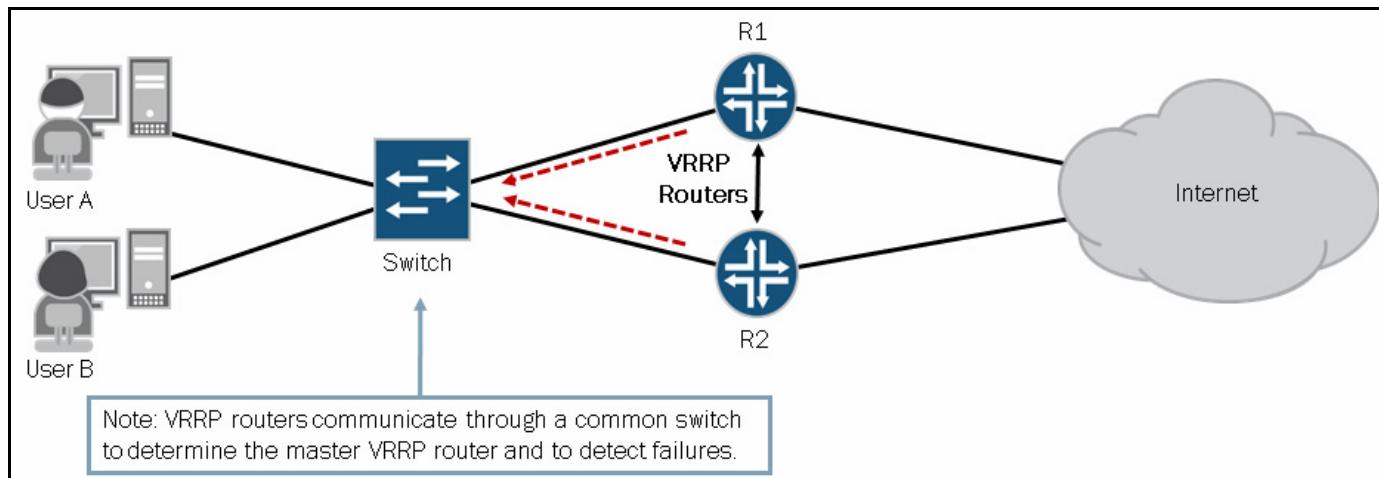
What If ...?

- What if R1, which is currently functioning as the gateway for the 172.25.100.0/24 subnet, failed or one of its two directly connected links became unusable?



The graphic presents a sample scenario and tests your knowledge of the overall impact the potential failures might have in this sample network environment. Because R1 serves as the only gateway beyond the 172.25.100.0/24 subnet, if it failed or either of its directly connected links became unusable, the users on the referenced subnet would not be able to communicate with any remote subnet. We present a recommended solution for this scenario throughout the subsequent pages in this section.

VRRP Defined



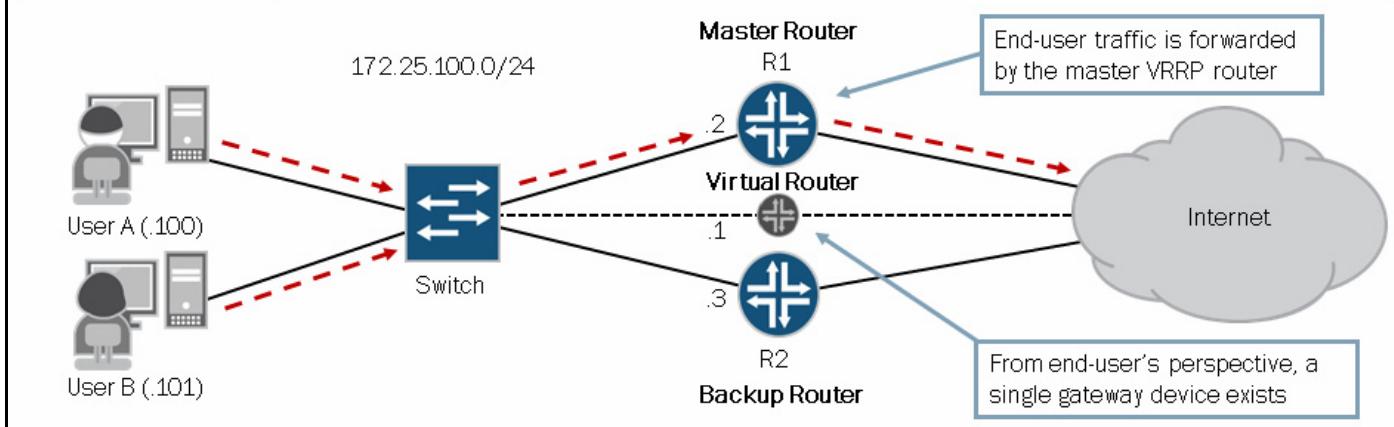
The Virtual Router Redundancy Protocol (VRRP) is a standards-based election protocol that can facilitate redundancy in a LAN environment and eliminate the single point of failure highlighted in the previous graphic. VRRP elects one of the participating

routers (known as VRRP routers) to function as gateway device, while all other VRRP routers serve in a backup capacity. All communications between VRRP routers occur through a common switch. We cover more details regarding the election process and general VRRP communications on subsequent slides.

Note that VRRP is very similar in functionality to Cisco Systems' Hot Standby Router Protocol (HSRP). VRRP is most commonly found in Ethernet environments but can also be used in LAN environments that use Token Ring or Fiber Distributed Data Interface (FDDI). VRRP is an industry standard and is defined in RFC 2338.

Terms and Concepts

Term	Description
VRRP Router	Any router participating in VRRP, including the master and all backup routers
Master Router	VRRP router performing packet forwarding and responding to ARP requests
Backup Routers	VRRP router available to assume the role of the master router upon failure
Virtual Router	Virtual entity that functions as default router on LAN: consists of virtual router ID and IP address used as gateway address known as VIP address

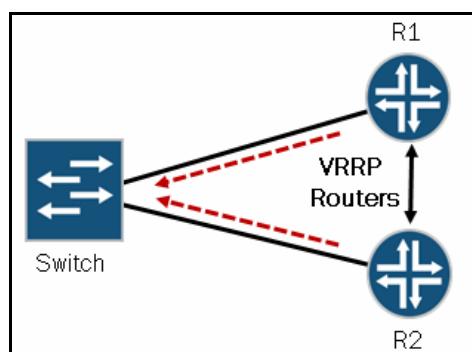


Any router participating in VRRP including the master and backup routers, is known as a VRRP router. The master router is the VRRP router responsible for forwarding packets on a given LAN segment. The master router also performs the Address Resolution Protocol (ARP) functions for the virtual router that it represents. A backup router is a VRRP router that is available to assume the role of the master router if a failure occurs. Multiple backup routers can exist for a given VRRP group.

From the user's perspective, a single gateway device exists when VRRP is deployed. This single gateway device, in actuality, is a virtual device known as the virtual router. The virtual router is a logical entity that functions as the default router for a LAN. The virtual router consists of a virtual router identifier (VRID) and virtual IP (VIP) address. The VRID uniquely identifies one virtual router from another. The VIP address is managed by the virtual router and is attached to the VRRP router functioning as the master for that network at any point in time.

VRRP Communications

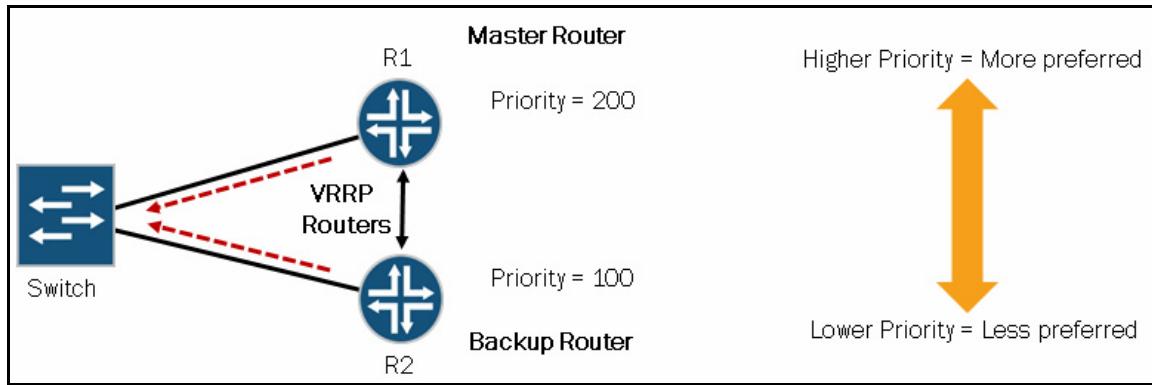
VRRP version 2 uses a common advertisement packet to facilitate communication between VRRP routers. VRRP uses this common advertisement packet primarily to relay the priority and state details of the master router for a given virtual router. The VRRP advertisement packet is encapsulated in an IP packet and sent to the Internet Assigned Numbers Authority (IANA) designated multicast address for VRRP, which is 224.0.0.18. The VRRP advertisement packet uses a time-to-live (TTL) value of 255 and cannot be forwarded beyond the local subnet on which it is sent. This value cannot be adjusted, and any packet that has a TTL value other than 255 is discarded. The default interval for VRRP advertisements is 1 second. You can modify this interval to a value in the range of 1–255 seconds. If subsecond VRRP advertisements are required, configure the **fast-interval** option with a range of 100–999 milliseconds. All participating VRRP routers must support this option.



Certain fields within the VRRP advertisements must match on all VRRP routers for a given group or virtual router. Some examples are the VRID value and authentication parameters. If the values for these fields, which require a match, are not the same, the packets are discarded and VRRP will not work properly.

When a VRRP router sends a VRRP packet, the router uses the virtual MAC address as the source MAC address. If a host on a LAN segment sends an ARP request for the VIP address, the master router responds with the virtual MAC address associated with the virtual router or group. The virtual MAC address is deterministic and uses the VRID or group number as its unique identifier. The virtual MAC address uses the following format: 00-00-5E-00-01-VRID.

Determining the Master Router



VRRP uses priority to elect the master router. The priority value range is 1–255, and higher priority values are preferred. The Junos OS assigns a default priority value of 100 to each VRRP router, unless the VRRP router owns the VIP address. If a VRRP router owns the VIP address, the priority value must be set to 255. In this case, this router always assumes the role of master if it is functioning properly. If this router fails, the VRRP router or backup router with the next highest priority at that time then assumes the role of master. When the router that owns the VIP address returns to normal operation, it preempts the existing master and assumes the role of master once again.

In environments where the VRRP routers do not own the VIP address, you can administratively disable preemption. We cover VRRP configuration details later in this section.

VRRP States

VRRP State	Description
Initialize	Router negotiates VRRP roles through startup events; no forwarding can be performed while in this state
Master	Router assumes traffic forwarding responsibilities for the LAN and responds to ARP requests
Backup	Router monitors master VRRP router and is ready to assume forwarding responsibilities if failure occurs
Transition	Router switches between master and backup states; no forwarding can be performed while in this state

The diagram shows a network setup with a 'Switch' on the left connected to two routers, R1 and R2, at the top. Router R1 is labeled 'Master' and Router R2 is labeled 'Backup'. A central question mark node represents the election logic. Arrows point from both routers to the question mark node. Above the routers, a box contains the text 'VRRP Details Compared'. To the right of the question mark node, a table lists 'R1's Priority = 200' and 'R2's Priority = 100'. Arrows point from the table to the 'Master' and 'Backup' labels respectively.

Prior to electing the master and backup routers for a VRRP group, all routers begin in the initialize state. In the initialize state, a startup event occurs, which essentially announces each VRRP router's capability and includes each router's priority setting as well as other required parameters used in the election process. During the initialize state, no forwarding occurs because there is no master router to represent the virtual router.

Based on the VRRP exchanges during the initialize state, VRRP elects the master router based on the priority values; all other routers assume the backup state. The master router assumes forwarding responsibilities for the LAN and responds to ARP requests sent to the VIP address. The master router sends periodic announcements to all other VRRP routers within the VRRP group. These announcements indicate the master router's state and priority. If the announcements are not received for a deterministic period of time, the backup router with the next highest priority assumes mastership. If a backup router that has a higher priority value than the current master router becomes available, it assumes the role of master unless preemption is administratively disabled.

Routers in the backup state monitor the health or state of the master router and stand ready to assume the mastership if necessary.

In the event of a mastership change, the backup router might, for a very brief moment, be in the *transition* state. This state is simply a transitional step, in which a router changes from the backup state to the master state. While in this state, no forwarding occurs for the LAN.

Sample VRRP Configuration

```
[edit interfaces ge-0/0/4]
user@R1# show
unit 0 {
    family inet {
        address 172.25.100.2/24 {
            vrrp-group 10 {
                virtual-address 172.25.100.1;
                priority 200;
            }
        }
    }
}
```

VRRP group number and VIP address must match on all routers in same redundant group

```
[edit interfaces ge-0/0/4]
user@R2# show
unit 0 {
    family inet {
        address 172.25.100.3/24 {
            vrrp-group 10 {
                virtual-address 172.25.100.1;
                priority 100;
            }
        }
    }
}
```

Based on the configurations shown, which router will assume the role of the master VRRP router? Why?

The inset provides sample VRRP configurations for two routers participating in the same VRRP group. Based on the configurations shown and the details covered on the previous pages, you know that R1 should assume the role of the master VRRP router because of its higher priority value.

To implement load balancing in your network, configure multiple VRRP groups for the same subnet with a different VIP for each group. You must ensure, using priority, that one router is the master VRRP router for one group while the second router is the master VRRP router for the other group. This configuration introduces two gateways for the same subnet, where each gateway device (the master VRRP router for each group) has a backup gateway device (the backup VRRP router for each group). Note that you must configure the end hosts to use one of the available gateways—the VIP for one of the two VRRP groups. Note that the load balancing distribution will vary between environments and relies on the number of hosts assigned to each gateway and the overall traffic patterns associated with those hosts.

Note that VRRP is not supported on all Junos devices. Check the product-specific documentation for your product for support information.

Additional Configuration Options

Configuration Option	Description
track	Monitors state of specified interface (typically a WAN interface) or route and reduces designated priority value for VRRP group if tracked interface or route is no longer available
accept-data	Allows master router to respond to ICMP requests sent to VIP address—by default, master router does not respond
authentication-type authentication-key	Authenticates VRRP messages between VRRP routers (type and key must match on all VRRP routers in the same group)
no-preempt	Disables preemption to avoid unwanted mastership changes: Note: Preemption is enabled by default, which means the router with the highest priority always assumes the master role

This graphic highlights some additional VRRP configuration options, along with a brief summary for each option. Refer to the following additional details for these options:

- Use the **track** configuration option to monitor interfaces or routes that are essential when forwarding traffic received through the VRRP interface. When a tracked interface or route becomes unavailable, the current priority value for the related VRRP group is reduced to a new priority value based on a user-defined setting. The reduction of priority for the VRRP group on the VRRP router involved can then trigger a mastership change. The **track** configuration option is an excellent way to maintain external reachability during a failure scenario.
- By default, the master router representing the virtual router does not respond to Internet Control Message Protocol (ICMP) requests sent to the VIP address unless the master router owns that IP address. If the behavior is to have the master router respond to ICMP requests sent to the VIP address—even if that router does not own that specific IP address—use the **accept-data** configuration option. Note that using the **accept-data** configuration option to facilitate ICMP responses violates RFC 2338. The RFC strictly prohibits any ICMP response from a VRRP router unless that router owns the VIP address. The use of this option can, however, dismiss unnecessary problem reports indicating that the gateway router is not responding to ICMP requests.
- Environments that have potential security risks should incorporate some level of authentication. Use the **authentication-type** and **authentication-key** configuration options to authenticate VRRP messages. The authentication options for VRRP are none, simple, and Message Digest 5 (MD5). For LANs with security concerns, we recommend MD5 authentication because it is the only authentication option that encrypts the VRRP exchanges. These configuration options must match on all VRRP routers within the same VRRP group.
- By default, the VRRP router with the highest priority setting becomes the master router for a given virtual router. If a new router becomes available that has a higher priority, it preempts mastership from the existing master. In situations where the VIP address is not owned by any of the participating VRRP routers within a specific VRRP group, you can administratively disable preemption using the **no-preempt** configuration option.

Additional VRRP configuration options exist. The following output provides a full list of the VRRP configuration options:

```
[edit interfaces ge-0/0/4 unit 0 family inet address 172.25.100.2/24]
user@R1# set vrrp-group 10 ?
Possible completions:
```

```

accept-data           Accept packets destined for virtual IP address
advertise-interval   Advertisement interval (1..255 seconds)
+ apply-groups        Groups from which to inherit configuration data
+ apply-groups-except Don't inherit configuration data from these groups
authentication-key    Authentication key
authentication-type   Authentication type
fast-interval         Fast advertisement interval (100..999 milliseconds)
no-accept-data       Don't accept packets destined for virtual IP address
no-preempt            Don't allow preemption
> preempt             Allow preemption
priority              Virtual router election priority (0..255)
> track               Interfaces to track for VRRP group
+ virtual-address     One or more virtual IPv4 addresses
virtual-link-local-address Virtual link-local addresses
> vrrp-inherit-from   VRRP group to follow for this VRRP group

```

You can use the **vrrp-inherit-from** configuration option when multiple VRRP groups are defined on the same physical interface. This option allows one or more inheriting groups to acquire certain configuration parameters from the active group. The inheriting groups acquire the **advertise-interval**, **authentication-key**, **authentication-type**, **fast-interval**, **no-preempt**, **preempt**, **track interface**, and **track route** configuration options from the active group.

In addition to simplifying the configuration tasks when multiple VRRP groups are defined on a single interface, the **vrrp-inherit-from** configuration option also reduces the amount of VRRP traffic that passes through that interface. Only the active group exchanges VRRP messages. Inheriting groups do not exchange VRRP messages, but rather inherit the VRRP state associated with the active group.

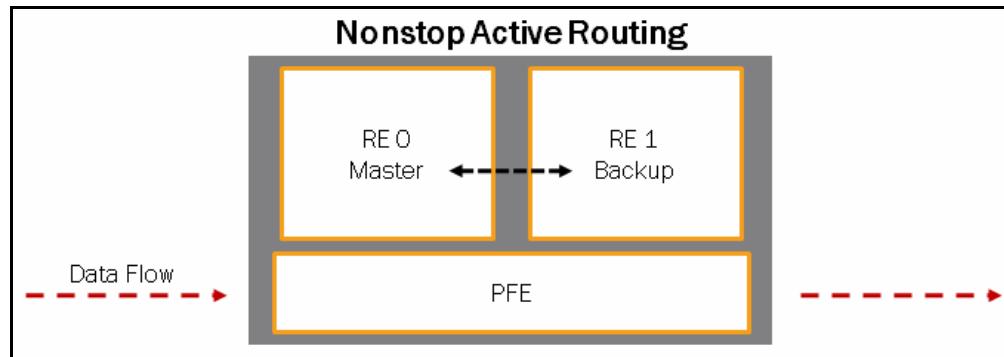
Monitoring VRRP Operations

<pre> user@R1> show vrrp ? Possible completions: <[Enter]> brief detail extensive interface summary track </pre>	<p>Execute this command</p> <p>Include options to view additional details or filter generated output</p>
<pre> user@R1> show vrrp summary Interface State Group VR state VR Mode Type Address ge-0/0/4.0 up 10 master Active lcl 172.25.100.2 vip 172.25.100.1 </pre>	
<pre> user@R2> show vrrp summary Interface State Group VR state VR Mode Type Address ge-0/0/4.0 up 10 backup Active lcl 172.25.100.3 vip 172.25.100.1 </pre>	

You can use the **show vrrp** commands shown in the inset to gather varying levels of detail. The inset shows that R1 is master and R2 is backup for VRRP group 10.

If you see that both VRRP routers show the master state, check the configuration and physical connectivity through the switch. You might also need to check any configured firewall filters on the VRRP routers to ensure those firewall filter permit VRRP communications.

Unified ISSU



A unified in-service software upgrade (ISSU) enables you to upgrade between two different Junos OS releases with no disruption on the control plane and with minimal disruption of traffic. Unified ISSU is supported only on dual Routing Engine platforms. In addition, graceful Routing Engine switchover (GRES) and nonstop active routing (NSR) must be enabled. Note that NSR is not supported on all the Junos devices. Refer to the technical publications for platform and protocol support details.

The master Routing Engine and backup Routing Engine must be running the same software release before you can perform a unified ISSU. You cannot take any PICs online or offline during a unified ISSU.

Perform a Unified ISSU

1. Enable GRES and NSR, and verify that the Routing Engines and protocols are synchronized
2. Verify that the master and backup engines are running the same software version.
3. Download the new software package from the Juniper Networks Support website and then copy the package to the router
4. Issue the **request system software in-service-upgrade** command on the master Routing Engine

The inset lists the high-level process for performing a unified ISSU.

To verify the status of Flexible PIC Concentrators (FPCs) and their corresponding PICs after the most recent unified ISSU, issue the **show chassis in-service-upgrade** command on the master Routing Engine:

```
user@host> show chassis in-service-upgrade
  Item          Status           Reason
  FPC 0        Online
  FPC 1        Online
  FPC 2        Online
    PIC 0      Online
    PIC 1      Online
  FPC 3        Offline          Offlined by CLI command
  FPC 4        Online
    PIC 1      Online
  FPC 5        Online
    PIC 0      Online
  FPC 6        Online
    PIC 3      Online
  FPC 7        Online
```

Troubleshooting Unified ISSU

1. Open a new session on the master Routing Engine and issue the **request system software abort in-service-upgrade** command.
2. Check the existing router session to verify that the upgrade has been aborted.

The inset lists a troubleshooting procedure for unified ISSU.

Review Questions

1. Describe the basic operations of GR.
2. What is a key benefit to using graceful Routing Engine switchover?
3. List the benefits of using BFD.
4. What is the purpose of the virtual router in VRRP?

Answers

1.

GR allows a restarting router to inform its adjacent neighbors of a restart event. The restarting router requests a grace period from the neighbor. During the restart event, the restarting router continues forwarding traffic, and convergence in the network is not disrupted. The helper routers hide the restart event from other devices not directly connected to the restarting router. In other words, the restart is not visible to the rest of the network, and the restarting router is not removed from the network topology.

2.

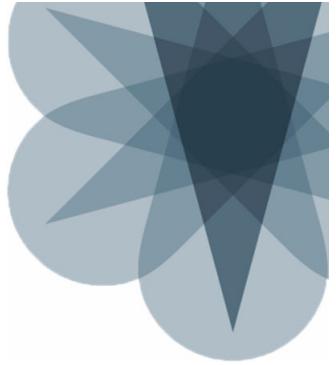
Graceful RE switchover preserves interface and kernel information and ensures that traffic forwarding is not interrupted during a mastership change.

3.

BFD relieves protocols from the need to provide fast failure detection, provides a single, common method for managing protocol timers, and can provide a failure detection mechanism for static routes.

4.

The virtual router is a logical entity that functions as the default router for a LAN. The virtual router consists of a virtual router identifier (VRID) and a virtual IP (VIP) address, and is always associated with the master VRRP router, which is responsible for forwarding traffic.



JNCIS-SP Study Guide—Part 1

Appendix A: IPv6

This Appendix Discusses:

- Differences between IP version 4 (IPv4) and IP version 6 (IPv6);
- IPv6 address types and addressing format;
- IPv6 interface configuration;
- Routing configuration for IPv6 environments; and
- Tunneling IPv6 traffic over an IPv4 network.

What Is IPv6?

IPv6 is a next-generation protocol designed to eventually replace IPv4, which is used primarily in networks today. Defined by the Internet Engineering Task Force (IETF), IPv6 is seen as a redesign of the IPv4 framework, primarily due to the depletion of available IP addresses. In fact, as early as the late 1980s, an Internet standard was defined in 1998 as RFC 2460.

There was an experimental protocol named Internet Stream Protocol (ST) that was considered by some to be Internet Protocol version 5, but ST was never officially known as IPv5 and never passed experimental stages.

IPv4 Versus IPv6

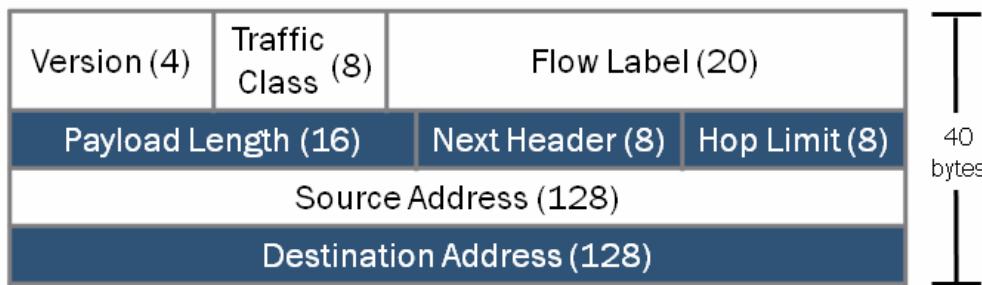
IPv4	IPv6
32-bit (4-byte) address supports 4,294,967,296 addresses	128-bit (16-byte) address supports 2^{128} (about 3.4×10^{38}) addresses
NAT can be used to extend address space limitations	Does not support NAT by design
Administrators must use DHCP or static configuration to assign IP addresses to hosts	Hosts use stateless address autoconfiguration to assign an IP address to themselves
IPsec support is optional	IPsec support is necessary
Options are integrated into the base header	Improved support for options using extension headers and overall simplification of the header format

Although address exhaustion was a major reason for the development of IPv6, several other issues exist that IPv6 attempts to resolve. The table describes some of the differences between IPv4 and IPv6. We discuss many of these differences in detail on the following pages.

Some of the additional benefits of IPv6 include the following:

- More efficient routing;
- Quality of service (QoS);
- Elimination of the NAT requirement;
- Network Layer security with end-to-end IPsec;
- Ease of management using stateless address autoconfiguration; and
- Improved header format to reduce header overhead.

IPv6 Structure

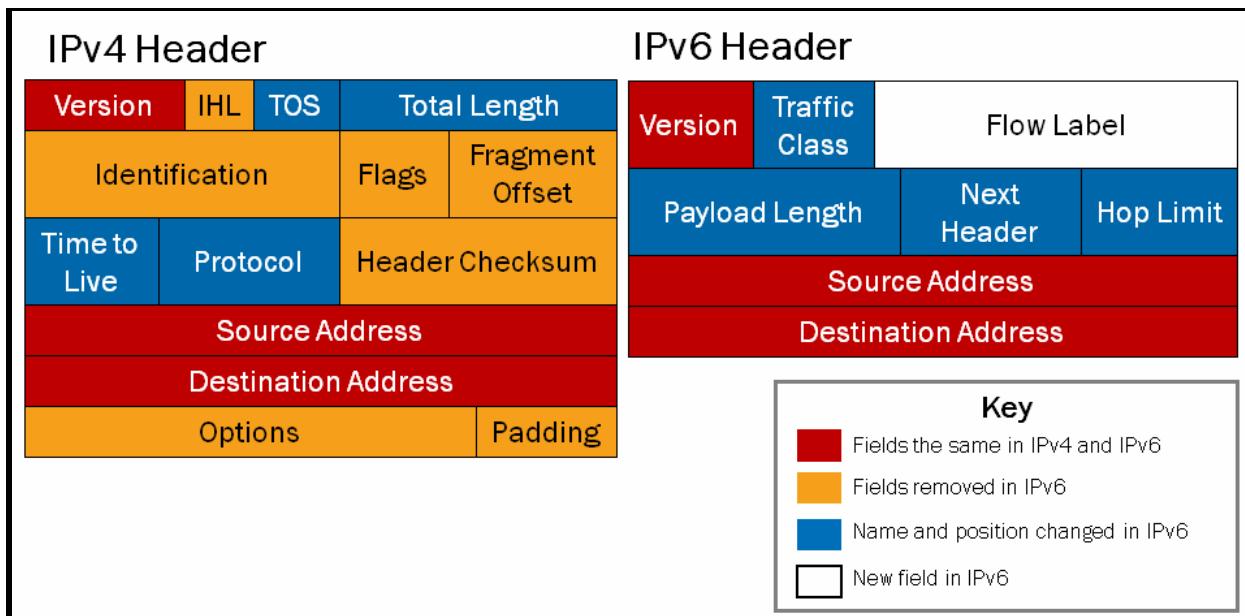


IPv6 headers include improvements over IPv4 headers that provide more efficient processing. Where IPv4 has a 40-byte field set aside for special options, IPv6 supports extension headers that can be appended to the IPv6 header to support additional options.

The IPv6 header has a fixed length of 40 bytes and includes the following fields:

- *Version*: 4-bit field containing the number 6, indicating IPv6;
- *Traffic class*: 8-bit field that determines the traffic priority;
- *Flow label*: 20-bit field used for QoS management;
- *Payload length*: 16-bit field indicates the size of the payload in octets;
- *Next header*: 8-bit field indicating the next encapsulated protocol;
- *Hop limit*: 8-bit field replaces the time-to-live (TTL) field in IPv4;
- *Source address*: 128 bits; and
- *Destination address*: 128 bits.

IPv4 Header Versus IPv6 Header



IPv4 headers and IPv6 headers share several characteristics. For example, the version, source, and destination fields from IPv4 were carried over to IPv6.

One of the biggest differences between an IPv4 packet and an IPv6 packet is that if IPv6 needs to fragment a packet, it uses extension headers to do so.

Several fields from the IPv4 header were dropped in favor of a more streamlined, efficient header. Because the IPv6 header is always a fixed size, the header length is no longer necessary. If a packet requires fragmentation, extension headers will take over the process using path MTU discovery. We discuss extension headers in more detail later in this appendix.

Some fields in the IPv6 header moved from their previous locations in IPv4. The type of service (ToS) field was replaced by the traffic class (TC) field. The TTL and protocol fields also were renamed and moved in IPv6.

IPv6 Extension Headers

As mentioned previously, the IPv6 header was improved with faster packet processing in mind. Because the IPv6 header is a fixed size of 40 bytes, the options that were once included inside the IPv4 header are now appended to the IPv6 packet using extension headers.

IPv6 Defines Six Extension Headers

Six extension headers can be used in IPv6, and can be described as follows:

- *Hop-by-hop options*: Signifies that the options need to be examined by each node along the path of a packet;
- *Routing*: Provides a list of intermediate nodes that should be visited on the path to the packet's destination;
- *Fragment*: Signals when a packet has been fragmented by the source;
- *Destination options*: Options examined only by the destination node, and capable of appearing twice in a packet;
- *Authentication header*: Used with IPsec to verify authenticity of a packet; and
- *Encrypted security payload*: Used with IPsec and carries encrypted data for secure communication.

IPv6 Addressing

The escalating demand for IP addresses was the driving factor for IPv6. IPv4 uses a 32-bit address supporting approximately 4.29 billion addresses. Because of the demand of address space, as well as early inefficient address implementation, IPv4 address allocation is expected to reach exhaustion in the near future. In April of 2009, the American Registry for Internet Numbers (ARIN) sent a letter predicting complete IPv4 address exhaustion as early as 2011, and warned all organizations to begin planning for IPv6 adoption. Other Regional Internet Registries (RIRs) have made similar predictions.

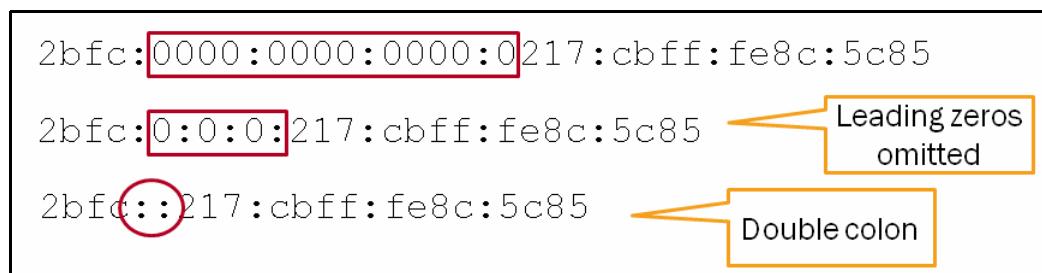
In comparison, IPv6 uses a 128-bit address supporting approximately 2^{128} addresses. If 2^{128} doesn't sound like a big number, think of the number of available addresses in another way. Enough IPv6 addresses exist to supply about 2^{95} addresses for each person on Earth or about 2^{52} addresses for each observable star in the known universe!

IPv6 Address Types

Three types of IPv6 address exist with which you should be familiar:

- **Unicast:** The unicast address is a unique address that identifies an interface or node. A packet with a unicast address travels only to the interface identified.
- **Multicast:** A multicast address is an identifier for a group of IPv6 interfaces that might belong to different nodes. An IPv6 packet with the multicast address as the destination travels to all interfaces in the group.
- **Anycast:** An anycast address is an identifier for a group of IPv6 interfaces that might belong to different nodes. However, unlike multicast addresses, an IPv6 packet with the anycast address as the destination travels to only one of the nodes identified in the group—typically the nearest node.

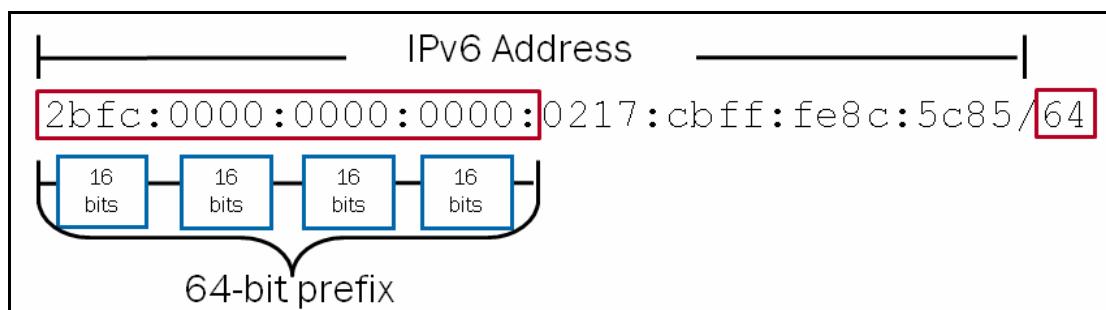
Address Notation



At first glance, an IPv6 address can look alarmingly complicated. IPv6 presents a 128-bit address in eight 16-bit hexadecimal blocks separated by colons. However, IPv6 does allow for abbreviation.

Consider the example in the inset. You might think that you are looking at three different IP addresses, but in fact, you are looking at the same address in all three examples. Four consecutive zeros in an address can be identified as a single zero. You can omit leading zeros from the notation. A double colon can replace consecutive zeros, leading zeros, or trailing zeros; however, you can not use a double colon twice in an address notation. Doing so will result in the misinterpretation of the IP address.

Prefix Notation



IPv6 has some similarities to IPv4 in text presentation. The prefix determines the network address or subnet. The example in the graphic shows a 64-bit prefix. The prefix in the graphic would be noted as follows:

2bfc:0000:0000:0000:0/64

2bfc:0:0:0:0/64

2bfc::0/64

Just like with IPv4, certain prefixes are reserved and should be used for specific types of traffic. RFC 4291 defines the latest rules regarding prefix notation. The following are a few examples:

- ::/128: The prefix notation is unspecified;
- ::1/128: This prefix notation should be used for the loopback;
- FF00::/8: This prefix notation should be used for multicast; and
- FE80::/10: This prefix notation should be used for the local link.

Address Allocation

Subscriber	Prefix
Home network subscribers, connecting through on-demand or always-on connections	48-bit prefix
Small and large enterprises	48-bit prefix
Very large subscribers	47-bit, or multiple 48-bit prefixes
Mobile networks, such as vehicles or mobile phones with an additional network interface	64-bit prefix, which allows multiple connections through a single prefix
A single PC, with no additional need to subnet, dialing-up from a hotel room	128-bit address can be assigned as part of a 64-bit prefix

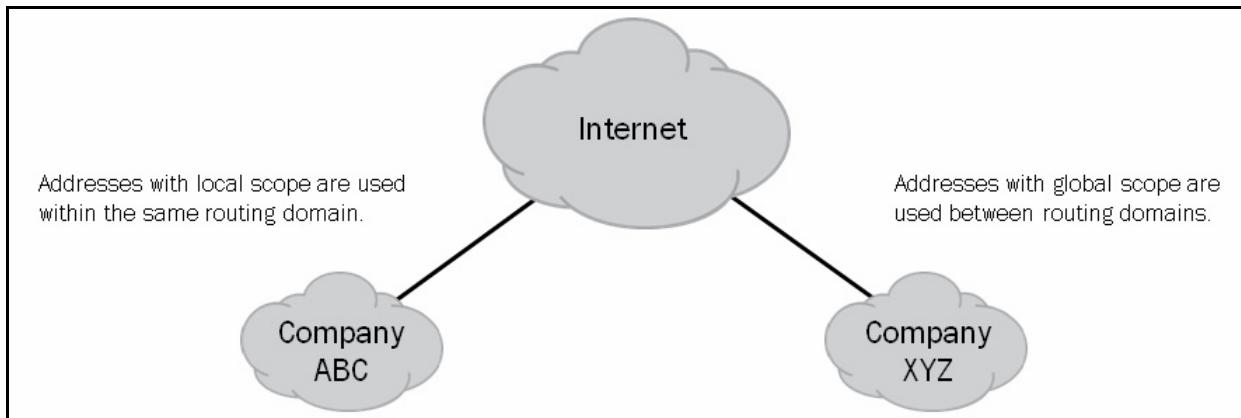
Similar to IPv4, IPv6 nodes must receive IP address assignments from their ISP. ISPs find information about their regional registries to verify if an IP address is in use. Address allocation is a work in progress, but RFC 3177 specifies a few rules that you should follow, which are shown in the table.

Special Addresses

The ::/16 prefix is reserved for special addressing. The unspecified address is similar to the 0.0.0.0 address in IPv4, and you should never assign it to an interface.

The loopback address is a string of 127 binary zeros followed by a binary one. It is denoted as "::1". A host uses the loopback address to send an IPv6 packet to itself. The IPv6 loopback address is functionally similar to the IPv4 loopback address. Like the unspecified address, the loopback address is not directly assigned to an interface.

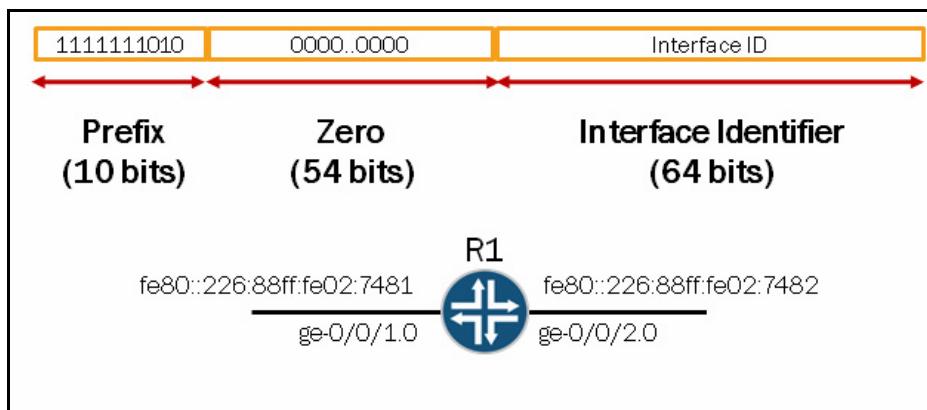
Address Scope



IPv6 addresses have scope, which identifies the application suitable for the address. Unicast and multicast addresses support scoping. Unicast addresses support two types of scope: global scope and local scope. There are two types of local scope: link-local addresses and site-local addresses. Link-local unicast addresses are used within a single network link. The first 10 bits of the prefix identify the address as a link-local address. Link-local addresses cannot be used outside a network link. Site-local unicast addresses are used within a site or intranet. A site consists of multiple network links, and site-local addresses identify nodes inside the intranet. Site-local addresses cannot be used outside the site.

Multicast addresses support 16 different types of scope, including node, link, site, organization, and global scope. A 4-bit field in the prefix identifies the scope.

Link-Local Unicast Addresses

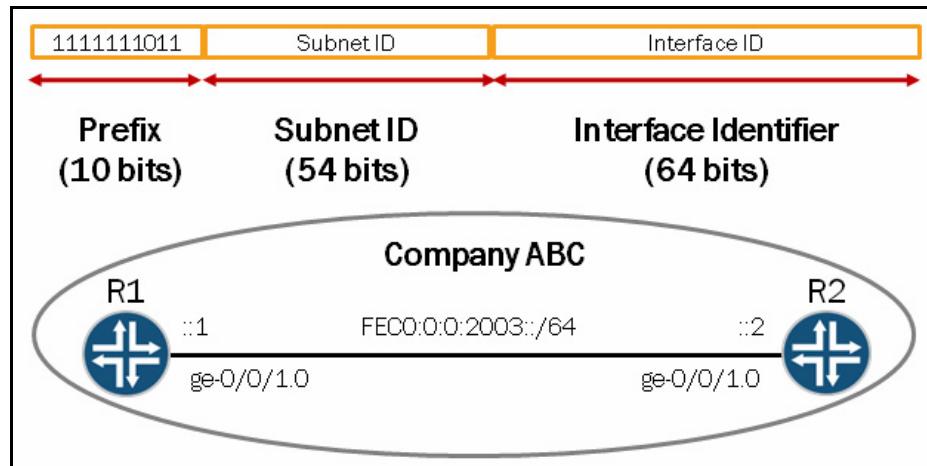


The link-local unicast address is identified with the binary prefix 1111111010 followed by a string of 54 binary zeros, the host-generated Interface ID, and a 64-bit mask. Any host can automatically generate this address and does so on all its IPv6 interfaces.

Because the first 64 bits are the same for every link-local address, this address is guaranteed to be unique only on an individual link, and could be duplicated by hosts on other subnets. Link-local addresses are never routable.

Interfaces configured for IPv6 automatically generate a link-local address. This address is often used for neighbor discovery, autoconfiguration, and routing protocol traffic. Link-local addresses were designed to provide automatic address configuration for small networks with a single subnet.

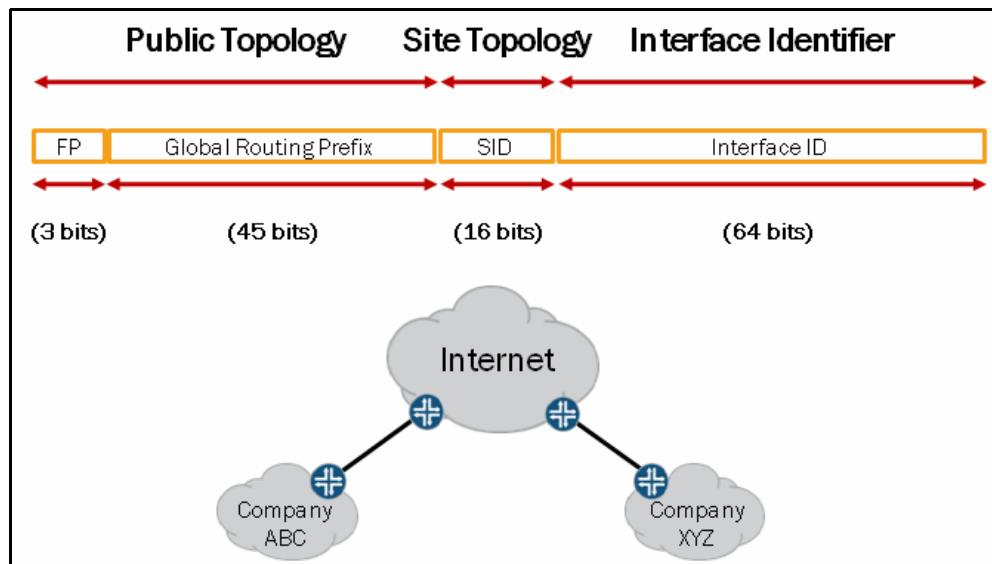
Site-Local Unicast Addresses



Site-local unicast addresses are identified with the binary prefix 1111111011 followed by 54 bits for the subnet ID, and the host-generated interface ID. Site-local unicast addresses are designed as a direct replacement for the RFC 1918 private addresses in IPv4, and should not be routable on the Internet.

The prefix for site-local unicast addresses is the same in all organizations. Site-local addresses are duplicated worldwide. The IPv6 designers expect that most, if not all, organizations will assign site-local addresses to their interfaces.

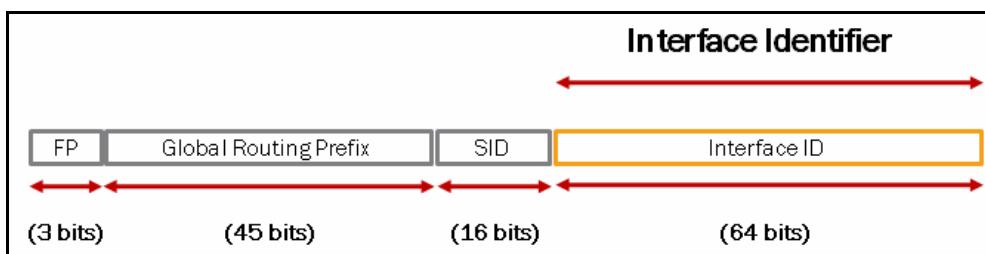
Global Unicast Addresses



Global unicast addresses are globally unique and are used to connect to and route through the Internet. These addresses are identified by the following six different fields:

- *Format prefix (FP)*: The binary prefix 001 is present in the first 3 bits, as this identifies the aggregatable global unicast address space. Currently, this range is the only range currently assigned by IANA for address allocation.
- *Global routing prefix*: The next 45 bits are reserved hierarchical address allocation. These bits are used to provide the highest level of address summarization typically applied at Tier 1 ISPs, regional address allocation entities, or possibly large peering points. It is extremely important that these bits be allocated wisely, because they will have a large impact on how the Internet is routed.
- *Subnet identifier (SID)*: The next 16 bits are reserved for local assignment to a link. The local administrator uses the SID to provide subnetting capability within a site.
- *Interface identifier (interface ID)*: The last 64 bits are reserved for the interface ID, allowing for easy autoconfiguration of a host on a network. We discuss the interface ID in more detail on a subsequent page.

Interface ID



The interface ID is a 64-bit field that uniquely identifies a host on a subnet. On Ethernet interfaces, it is a permutation of the MAC address associated with the interface to which the address is assigned. The interface ID is expressed in the IEEE EUI-64 format.

Because the host can determine the subnet to which it is connected by router advertisements, and can generate its own interface ID, IPv6 addresses are easy to configure automatically on any host.

Because Ethernet MAC addresses are only 48 bits long, the address needs additional bits to widen it to the specified 64-bit length. To obtain an Ethernet interface ID, concatenate the first 24 bits of the MAC address (company ID) with binary value 1111111111111110 (0xFFFF) and the remaining 24 bits of the MAC address (manufacturer extension ID).

EUI-64 addresses set the universal/local bit to a value of 1 to signify that this address is globally unique. The universal/local bit is the next-to-lowest significant bit of the first octet of the address. For example, the Ethernet MAC address 00:10:A4:A6:69:D0 would generate an interface ID of 00:10:A4FF:FEA6:69:D0.

Stateless Autoconfiguration: Part 1

One of the great tasks of network management is assigning the many devices in your network an IP address. In IPv4, address assignment is managed using two different methods. The first method is to statically assign an IP address to each device. This method is efficient only if the network is small and you do not plan to change IP addresses frequently. The second method is to configure IP addresses automatically using the Dynamic Host Configuration Protocol (DHCP). With DHCP, a device that requires an IP address contacts the DHCP server and automatically receives an IP address. Although DHCP is considered easier to manage than statically assigning an IP address to a node, you must configure and maintain the DHCP servers. Static IP address assignment and DHCP are considered stateful configuration methods.

One of the enhancements to IPv6 is the implementation of stateless autoconfiguration. Stateless autoconfiguration allows IPv6 nodes to automatically assign an IPv6 address to a neighbor, effectively eliminating the need for static address assignment and DHCP. An administrator can still use stateful configuration methods, if necessary. We cover DHCPv6 later in this appendix.

Stateless autoconfiguration consists of several elements:

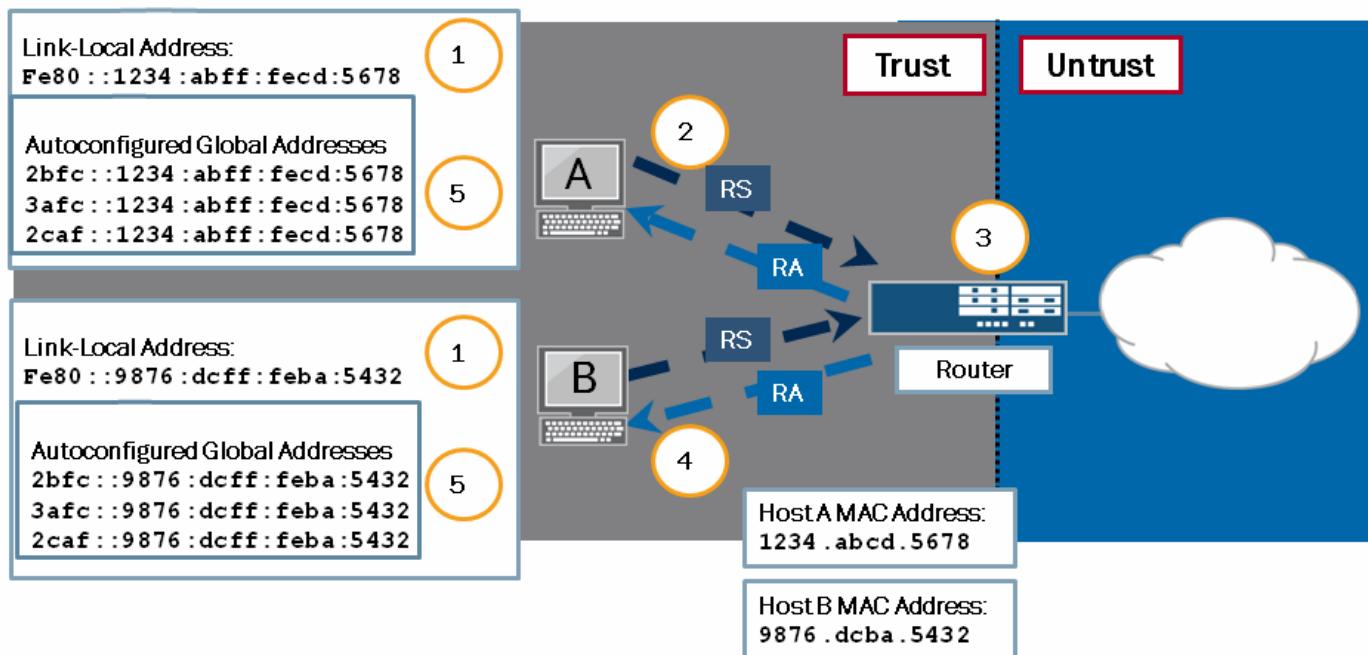
- **Extended unique identifier (EUI):** This 64-bit hexadecimal value is used to identify an interface. If you do not specify an EUI value explicitly, the security device autogenerated it from the MAC address of the IPv6 interface.
- **Router advertisement message:** An IPv6 router sends this message to on-link hosts periodically or in response to an RS request from another host. Information in an RA message can include IPv6 prefixes from the router, MTU messages, specific routes to the router, whether to perform IPv6 autoconfiguration, and a time period for how long an address should remain valid.
- **Router solicitation message:** Hosts send this message to discover the presence and properties of on-link routers. When an IPv6 router receives an RS request from a host, it responds by transmitting an RA message back to the host. An RA announces the existence of the router and provides the host with the information it needs to perform autoconfiguration tasks.
- **Prefix list:** This table contains IPv6 prefixes. When entries are present in the list, the router includes the entries in the RAs the router sends to on-link hosts. Each time a host receives an RA, the host can use the prefixes to perform address autoconfiguration.

Neighbor Discovery

Neighbor discovery (ND) is the process of tracking the reachability status for neighbors in a local link. Defined in RFC 2461, several enhancements were made to neighbor discovery since IPv4. ND combines and improves upon Address Resolution Protocol (ARP), and Internet Control Message Protocol (ICMP) router discovery and redirect to detect whether neighbors are reachable.

A device views a neighbor as reachable when the device receives recent confirmation that the neighbor received and processed IP traffic or NS requests. Otherwise, the device considers the neighbor unreachable. An IPv6 router with ND enabled can send ND information to downstream nodes. ND is optional on IPv6 devices.

Stateless Autoconfiguration: Part 2



Consider the example in the graphic. The following steps must take place for address autoconfiguration to occur:

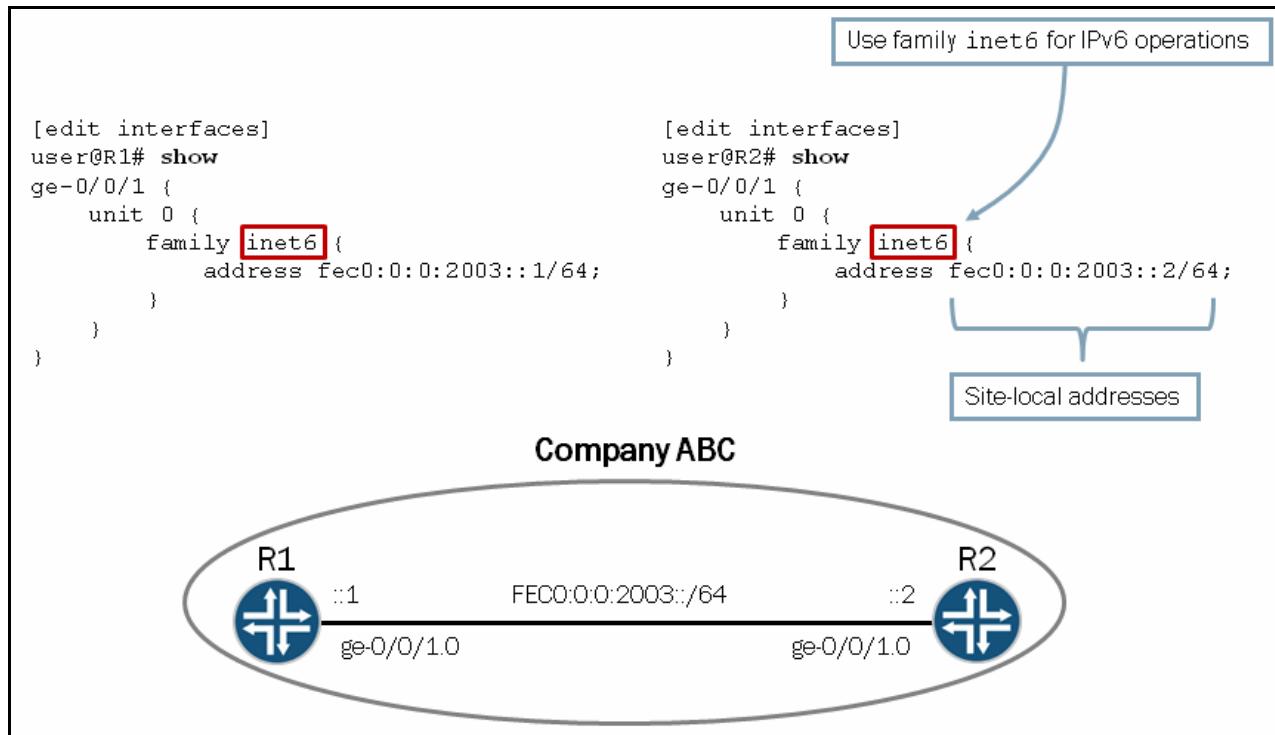
1. On startup, IPv6 Hosts A and B generate link-local addresses from their MAC addresses.
2. Each host broadcasts RS messages. Each message uses the host link-local address as the source address for the RS packets.
3. The IPv6 router receives the RS messages.
4. The IPv6 router transmits confirming RA messages to the hosts. These messages contain a prefix list.
5. The hosts use the prefixes to perform autoconfiguration.

Stateful Autoconfiguration

As mentioned previously, if stateless autoconfiguration is not desired, an administrator can manually assign a static address to a node. DHCP is also available. In IPv6, DHCP is known as stateful DHCPv6, and RFC 3315 defines it.

Several reasons exist to configure DHCPv6; for example, you might want to implement a specific IPv6 addressing scheme across your network. You might desire dynamic assignment of IP addresses to DNS servers. In some cases, you might require dynamic updates to your DNS servers. For security reasons, you might want to exclude the MAC address as part of the IPv6 address.

Interface Configuration Example



The graphic illustrates two routers with a site-local IPv6 address assigned to their ge-0/0/1.0 interface.

Interface Verification Example

user@R1> show interfaces terse ge-0/0/1					
Interface	Admin	Link	Proto	Local	Remote
ge-0/0/1	up	up			
ge-0/0/1.0	up	up	inet6	fe80::226:88ff:fe02:7481/64 fec0:0:0:2003::1/64	

user@R2> show interfaces terse ge-0/0/1					
Interface	Admin	Link	Proto	Local	Remote
ge-0/0/1	up	up			
ge-0/0/1.0	up	up	inet6	fe80::226:88ff:fe02:6b81/64 fec0:0:0:2003::2/64	

You use the **show interface terse** command to verify interface status. The inset shows the site-local address defined in the previous graphic along with the link-local addresses that the configured interfaces generated and assigned.

Displaying IPv6 Routing Information

```
user@R1> show route table inet6

inet6.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

fe80::/64      *[Direct/0] 00:59:12
                > via ge-0/0/1.0
fe80::226:88ff:fe02:7481/128
                * [Local/0] 00:59:12
                Local via ge-0/0/1.0
fec0:0:0:2003::/64 * [Direct/0] 00:59:12
                > via ge-0/0/1.0
fec0:0:0:2003::1/128
                * [Local/0] 00:59:12
                Local via ge-0/0/1.0
```

You use the **show route table inet6** command to verify the routing table contents. In the example output, we have added the interface routes for both the site-local addresses and link-local addresses.

Determining the Data Link Layer Address Using Ping

```
user@R1> show ipv6 neighbors

user@R1> ping fec0:0:0:2003::2
PING6(56=40+8+8 bytes) fec0:0:0:2003::1 --> fec0:0:0:2003::2
16 bytes from fec0:0:0:2003::2, icmp_seq=0 hlim=64 time=19.912 ms
16 bytes from fec0:0:0:2003::2, icmp_seq=1 hlim=64 time=18.091 ms
16 bytes from fec0:0:0:2003::2, icmp_seq=2 hlim=64 time=1.828 ms
16 bytes from fec0:0:0:2003::2, icmp_seq=3 hlim=64 time=2.324 ms
^C
--- fec0:0:0:2003::2 ping6 statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/std-dev = 1.828/10.539/19.912/8.489 ms

user@R1> show ipv6 neighbors
IPv6 Address          Linklayer Address    State        Exp Rtr Secure Interface
fe80::226:88ff:fe02:6b81 00:26:88:02:6b:81  stale       1187 yes no      ge-0/0/1.0
fec0:0:0:2003::2        00:26:88:02:6b:81  stale       747 yes no      ge-0/0/1.0
```

Initially, all Data Link Layer addresses of any IPv6 neighbor are unknown. As shown in the inset, you can use the ping utility not only to verify reachability but also to learn the Data Link Layer addresses of IPv6 neighbors. After the ping operation succeeds, R1's neighbor (R2) is listed along with its IPv6 and Data Link Layer address information.

IPv6 Multicast Address

A multicast address is an identifier for a group of nodes. Unlike with anycast addresses, when a packet travels to a multicast address, all members of the multicast group process the packet. IPv4 defined multicast, but IPv6 has redefined and improved multicast.

Broadcast packets are not routable, and every node on a subnet must process them. Multicast packets, on the other hand, are processed only by nodes of the multicast group. Multicast packets can also be forwarded over routers. In IPv6, ICMPv6 is used for multicast group management to optimize multicast traffic. This process is referred to as Multicast Listener Discovery (MLD).

Multicast addresses are identified by the high order byte FF, or 1111 1111 in binary notation. The following three types of multicast addresses exist, as defined in RFC 4291:

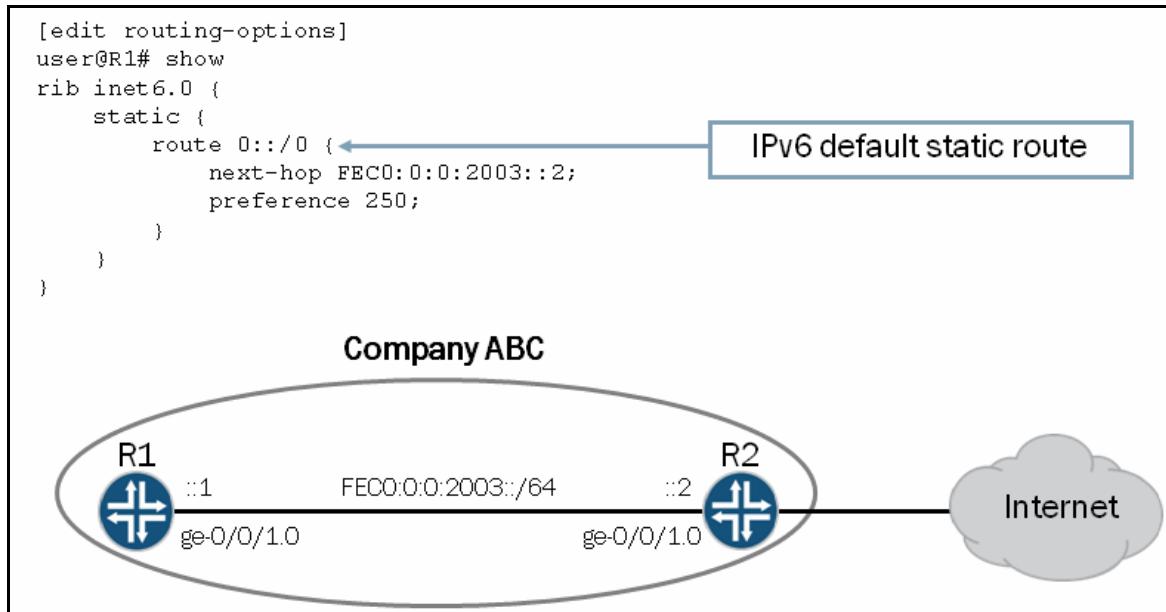
- Solicited-node multicast addresses are for Neighbor Solicitation (NS) messages;
- All-nodes multicast addresses are for Router Advertisement (RA) messages; and
- All-routers multicast addresses are for Router Solicitation (RS) messages.

IPv6 Anycast Address

The anycast address is a new type of address in IPv6, and RFC 2526 defines it. You can assign an anycast address to multiple interfaces in a group. Multiple nodes might be responding with the same anycast address, but the packet will travel only to one of the nodes. An anycast packet travels to the nearest node, based on the notion of distance determined by the routing protocols in use.

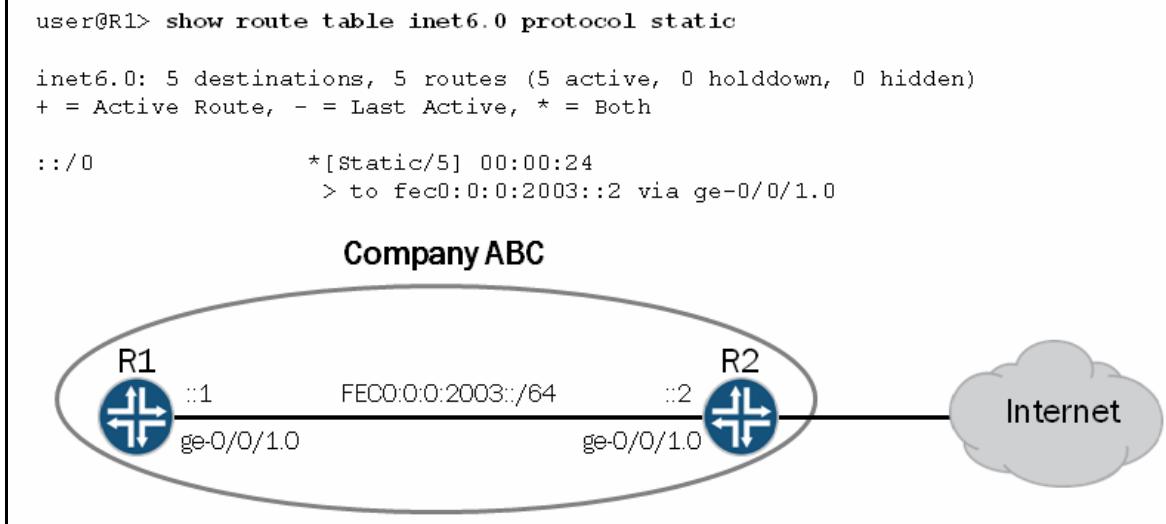
Anycast addresses offer the potential for a number of important services. For example, you can use an anycast address to allow nodes to access one of a collection of servers providing a well-known service, without manual configuration in each node of the list of servers. You can also use an anycast address in a source route to force routing through a specific Internet service provider (ISP), without limiting routing to a single specific router providing access to that ISP.

Static Route Configuration Example



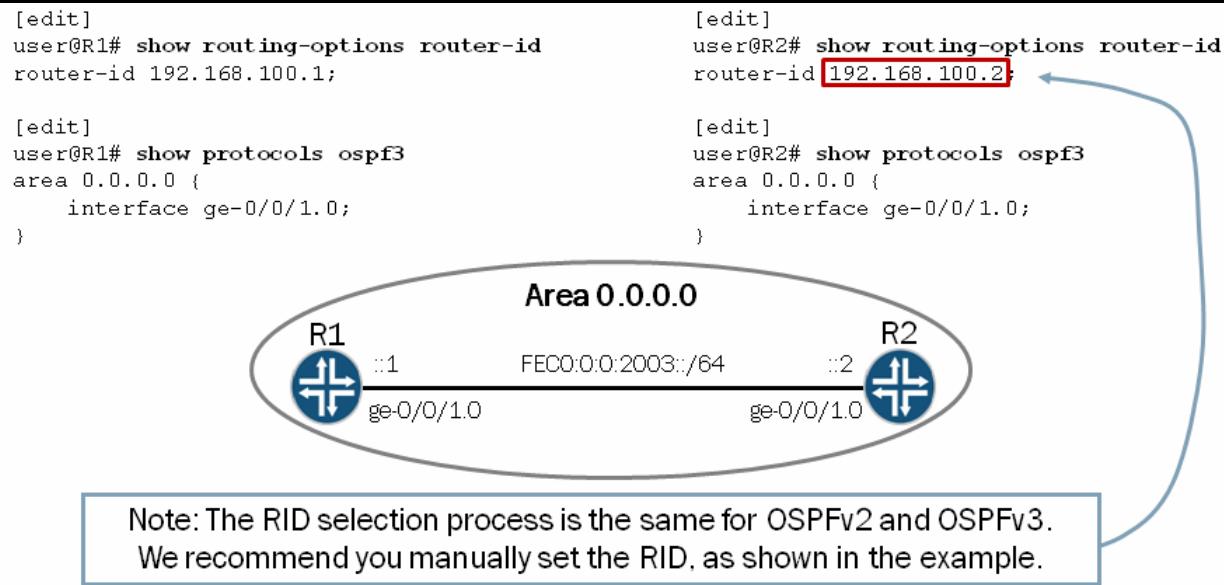
The graphic provides a sample default static IPv6 route configuration.

Displaying Static Routes



The graphic illustrates the operational mode command used to view static routes in the inet6.0 routing table.

OSPFv3 Configuration Example



This graphic shows a sample OSPFv3 configuration for two neighboring routers. Note that OSPFv3, like OSPFv2, uses a 32-bit router ID (RID) which is defined under the [edit routing-options] hierarchy. Although you can dynamically select the RID from IPv4 addresses defined on the router, we recommend you always configure a RID.

Monitoring OSPFv3 Operations

OSPFv2	OSPFv3
show ospf neighbor	show ospfv3 neighbor
show ospf interface	show ospfv3 interface
show ospf database	show ospfv3 database
show ospf route	show ospfv3 route

```
user@R1> show ospfv3 neighbor
ID                  Interface            State      Pri   Dead
192.168.100.2      ge-0/0/1.0        Full      128    36
Neighbor-address fe80::226:88ff:fe02:6b81
```

Most operational **show** commands used to monitor OSPFv3 are nearly identical to the commands used to monitor OSPFv2; the key difference is you must replace the **ospf** portion of those commands with **ospfv3**. The table illustrates a number of common operational commands for both OSPFv2 and OSPFv3 for comparison.

IS-IS Configuration Example

Interface Configuration <pre>[edit interfaces] user@R1# show ge-0/0/1 { unit 0 { family iso; family inet6 { address fec0:0:0:2003::1/64; } } } lo0 { unit 0 { family iso { address 49.0002.1111.1111.1111.00; } family inet6 { address fec0:0:0:1001::1/128; } } }</pre>	Protocol Configuration <pre>[edit protocols] user@R1# show isis { interface ge-0/0/1.0; interface lo0.0; }</pre>
--	--

The graphic illustrates a sample IS-IS configuration using IPv6 addressing instead of IPv4 addressing. Note that other than the IP addresses, the IS-IS configuration is the same for IPv6 and IPv4 environments.

Monitoring IS-IS Operations

<pre>user@R1> show isis interface IS-IS interface database: Interface L CirID Level 1 DR Level 2 DR L1/L2 Metric ge-0/0/1.0 3 0x2 R1.02 R1.02 10/10 lo0.0 0 0x1 Passive Passive 0/0 user@R1> show isis adjacency Interface System L State Hold (secs) SNPA ge-0/0/1.0 R2 1 Up 19 0:26:88:2:6b:81 ge-0/0/1.0 R2 2 Up 20 0:26:88:2:6b:81 user@R1> show isis adjacency detail R2 Interface: ge-0/0/1.0, Level: 1, State: Up, Expires in 21 secs Priority: 64, Up/Down transitions: 1, Last transition: 00:18:10 ago Circuit type: 3, Speaks: IP, IPv6, MAC address: 0:26:88:2:6b:81 Topologies: Unicast Restart capable: Yes, Adjacency advertisement: Advertise LAN id: R1.02, IP addresses: 192.168.100.2 IPv6 addresses: fe80::226:88ff:fe02:6b81 ...</pre>
--

As shown in the inset, you use the same operational **show** commands for IS-IS in IPv4 and IPv6 environments.

BGP Configuration Example

```
[edit routing-options]
user@R1# show
router-id 192.168.100.1;
autonomous-system 64700;

[edit protocols bgp]
user@R1# show
group int-64700 {
    type internal;
    local-address fec0:0:0:1001::1;
    neighbor fec0:0:0:1002::1;
}
group ext-65100 {
    type external;
    peer-as 65100;
    neighbor fec0:0:0:2005::2;
}
```

The inset provides a sample BGP configuration. BGP configuration is almost identical for IPv6 as it is for IPv4. The major difference is that you specify an IPv6 address for the local and peer addresses rather than IPv4 addresses.

Monitoring BGP Operations

```
user@R1> show bgp summary
Groups: 2 Peers: 2 Down peers: 0
Table          Tot Paths   Act Paths Suppressed      History Damp State     Pending
inet6.0          0           0           0           0           0           0           0
Peer             AS          InPkt       OutPkt      OutQ   Flaps Last Up/Dwn
State|#Active/Received/Accepted/Damped...
fec0:0:0:1002::1 64700        11          12          0          0    4:00 Establ
    inet6.0: 0/0/0/0
fec0:0:0:2005::2 65100        11          12          0          0    4:05 Establ
    inet6.0: 0/0/0/0
```

You use the same operational **show** commands for BGP in IPv6 environments that you use in IPv4 environments. The inset illustrates the **show bgp summary** command for two IPv6 peers in different BGP groups.

Tunneling IPv6 Traffic

The second broad transition method suggests using tunnels to span IPv4 networks until all the intermediate routers have been upgraded to support IPv6.

Tunneling requires encapsulation of the IPv6 packet within an IPv4 header. The new IPv4 packet is then forwarded across the IPv4 network to the other side, where the IPv4 header is removed and the IPv6 packet is either processed or forwarded.

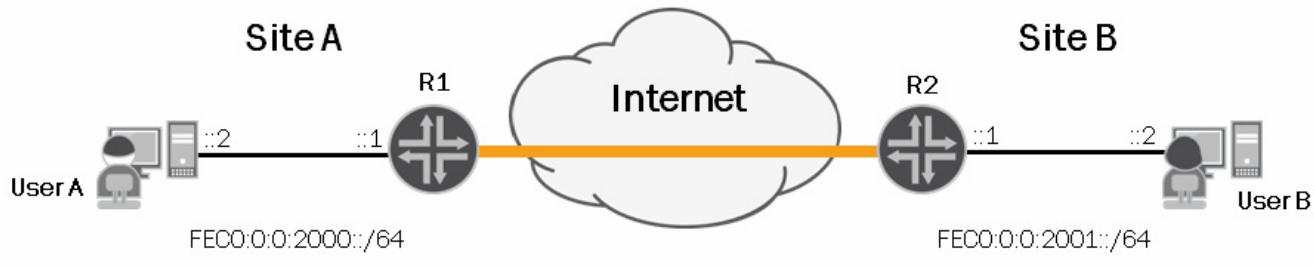
Multitude of Tunneling Approaches

Many approaches to tunneling have been defined. The following four approaches currently show the most promise:

- IPv4-compatible addressing;
- Configured tunnels;
- 6to4; and
- 6over4.

Case Study: Objective and Topology

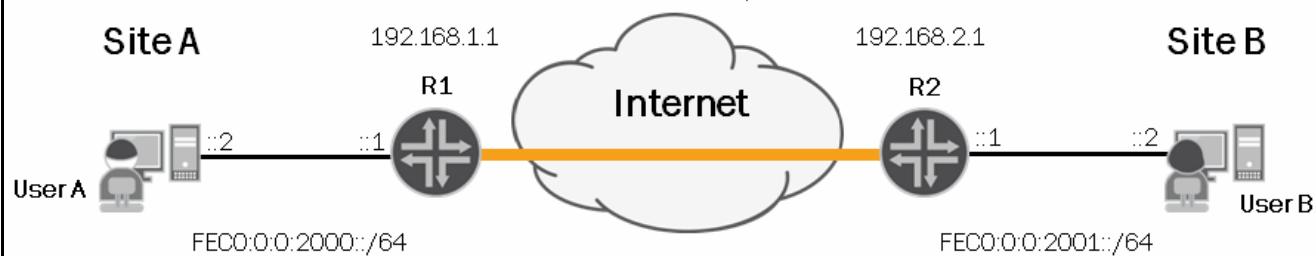
- Implement an IPv6-over-IPv4 tunnel to transport IPv6 traffic over the Internet and between sites A and B



The graphic provides the objective and topology used for this case study. In this case study, you focus on IPv6-over-IPv4 tunneling, which is also known as configured tunnels. Subsequent pages provide the configuration and monitoring tasks typically used when implementing an IPv6-over-IPv4 tunnel.

Case Study: Defining the Tunnel Interface

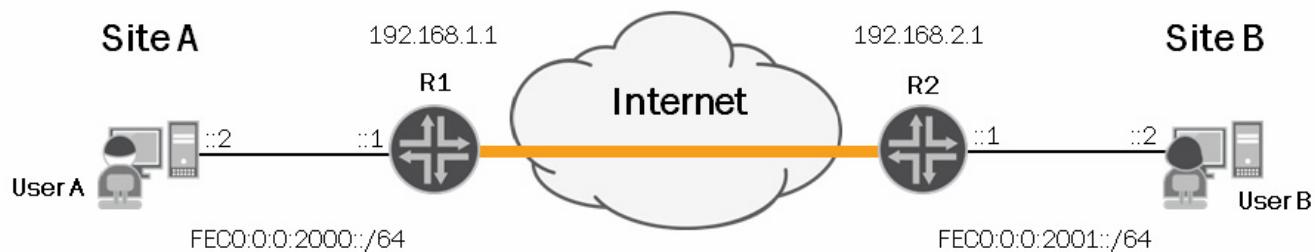
```
[edit]
user@R1# show interfaces gr-0/0/0
unit 0 {
    tunnel {
        source 192.168.1.1;
        destination 192.168.2.1;
    }
    family inet6 {
        address fec0:0:0:1000::1/126;
    }
}
[edit]
user@R2# show interfaces gr-0/0/0
unit 0 {
    tunnel {
        source 192.168.2.1;
        destination 192.168.1.1;
    }
    family inet6 {
        address fec0:0:0:1000::2/126;
    }
}
```



The graphic shows the tunnel interface configuration for the R1 and R2 devices.

Case Study: Defining the Required Routes

```
[edit]
user@R1# show routing-options
rib inet6.0 {
    static {
        route fec0:0:0:2001::/64 next-hop gr-0/0/0.0;
    }
    static {
        route 192.168.2.1/32 next-hop 172.18.1.1;
    }
}
```



The graphic shows the required routes for the R1 device. The IPv4 static route is used to establish the GRE tunnel, and the IPv6 static route is used to direct traffic destined to the remote IPv6 network through the GRE tunnel. The following output shows R2's configuration:

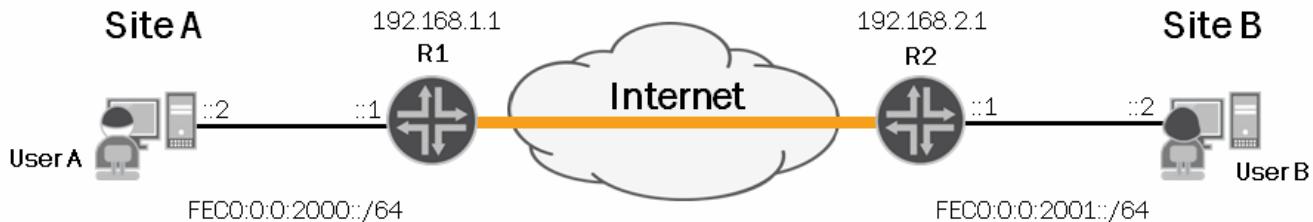
```
[edit]
user@R2# show routing-options
rib inet6.0 {
    static {
        route fec0:0:0:2000::/64 next-hop gr-0/0/0.0;
    }
}
static {
    route 192.168.1.1/32 next-hop 172.18.2.1;
}
```

Although not shown in this example, all intermediary devices between the tunnel endpoints (R1 and R2, in this example) must have a route to the IP addresses used for the tunnel (typically loopback addresses).

Case Study: Verifying Operations—Part 1

```
lab@R1> show interfaces gr-0/0/0 terse
Interface          Admin Link Proto   Local                  Remote
gr-0/0/0           up    up      inet6   fe80::226:88ff:fe02:7480/64
gr-0/0/0.0         up    up      inet6   fec0:0:0:1000::1/126
```

```
lab@R2> show interfaces gr-0/0/0 terse
Interface          Admin Link Proto   Local                  Remote
gr-0/0/0           up    up      inet6   fe80::226:88ff:fe02:6b80/64
gr-0/0/0.0         up    up      inet6   fec0:0:0:1000::2/126
```



This graphic and the next few graphics provide some verification tasks. This graphic specifically illustrates the **show interfaces terse** command, which you can use to quickly verify the tunnel interface is up.

Case Study: Verifying Operations—Part 2

```
user@R1> show route 192.168.2.1
inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
  192.168.2.1/32      *[Static/5] 01:49:40
                        > to 172.18.1.1 via ge-0/0/3.0

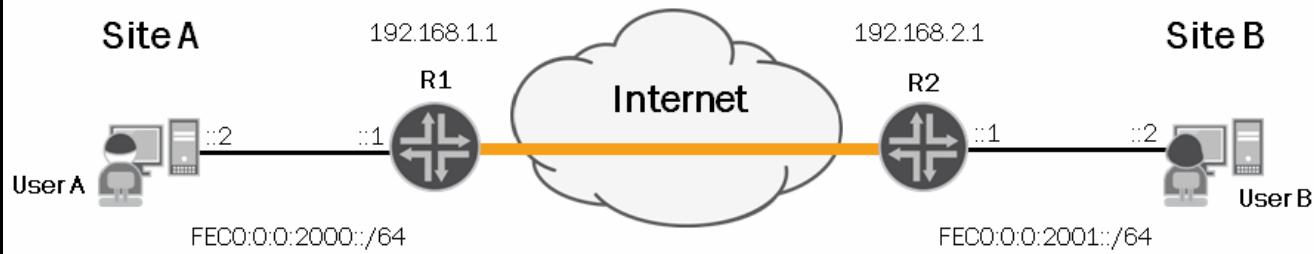
user@R1> show route table inet6.0 fec0:0:0:2001::/64
inet6.0: 7 destinations, 8 routes (7 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
  fec0:0:0:2001::/64 * [Static/5] 00:31:03
                        > via gr-0/0/0.0
```

As shown in the inset, you use the **show route** command to ensure the required routes are installed. You should perform this verification step on both tunnel endpoints.

Case Study: Verifying Operations—Part 3

```
user@R1> ping fec0:0:0:2001::2 source fec0:0:0:2000::1 rapid count 25
PING6(56=40+8+8 bytes) fec0:0:0:2000::1 --> fec0:0:0:2001::2
!!!!!!!!!!!!!!!
--- fec0:0:0:2001::2 ping6 statistics ---
25 packets transmitted, 25 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.492/3.185/8.064/2.521 ms
```

```
user@R1> show interfaces gr-0/0/0.0 detail | find "traffic statistics"
Traffic statistics:
Input packets: 25
Output packets: 25
...
```



The final verification tasks we highlight include sending IPv6 traffic over the IPv4-based tunnel. Once IPv6 traffic is sent and received over the tunnel, you can use the **show interfaces** command shown in the graphic to verify usage statistics on the GRE interface.

Review Questions

1. Name three benefits of switching from IPv4 to IPv6.
2. What types of unicast address exist?
3. What command would you use to view IPv6 OSPF neighbors?

Answers

1.

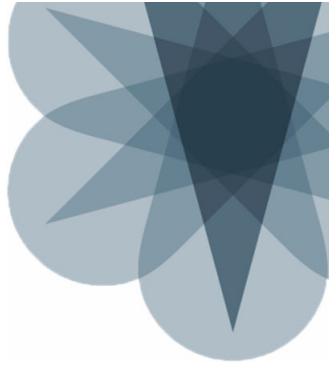
IPv6 provides extended IP address space, stateless autoconfiguration, a simplified header format, and improved support for options and extensions.

2.

The unicast address types include link-local addresses, site-local addresses, and global addresses.

3.

You can use the **show ospf3 neighbors** command to view IPv6 OSPF neighbors. Remember that when monitoring OSPFv3, you can simply replace **ospf** with **ospf3** in most of the operational monitoring commands.



JNCIS-SP Study Guide—Part 1

Appendix B: IS-IS

This Appendix Discusses:

- IS-IS protocol operations;
- IS-IS protocol data unit (PDUs);
- The purpose of the designated intermediate systems; and
- Configuration, monitoring, and troubleshooting of IS-IS.

The IS-IS Protocol

The IS-IS protocol is an interior gateway protocol (IGP) that uses link-state information to make routing decisions. It also uses a shortest-path-first (SPF) algorithm, similar to OSPF.

The ISO

The International Organization for Standardization (ISO) developed IS-IS to be the routing protocol for the ISO's Connectionless Network Protocol (CLNP) and is described in ISO 10589. Digital Equipment Corporation developed the protocol for its DECnet Phase V. The ISO was working on IS-IS at the same time that the Internet Advisory Board (IAB) was working on OSPF. ISO proposed that IS-IS be adopted as the routing protocol for TCP/IP in place of OSPF. This proposal was driven by the opinion that TCP/IP was an interim protocol suite that the Open Systems Interconnection (OSI) suite would eventually replace.

Dual IS-IS

To support the predicted transition from TCP/IP to OSI, an extension to IS-IS, known as *Integrated IS-IS*, was proposed. The purpose of integrated IS-IS, also known as *Dual IS-IS*, was to provide a single routing protocol with the capabilities of routing both Connectionless Network Service (CLNS) and IP. The protocol was designed to operate in a pure CLNS, pure IP, or dual CLNS and IP environment.

Single Algorithm

Like all integrated routing protocols, integrated IS-IS requires that all routers run a single routing algorithm. Link-state PDUs sent by routers running Integrated IS-IS include all destinations running either IP or CLNP Network Layer protocols. Routers running integrated IS-IS still must support protocols such as Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP) for IP, and the End System-to-Intermediate System (ES-IS) protocol for CLNP.

Link-State PDUs

Standard IS-IS packets must be modified to support multiple Network Layer protocols. IS-IS packet formats were designed to support the addition of new fields without a loss of compatibility with nonintegrated versions of IS-IS.

IS-IS adds type, length, and value (TLV) objects (discussed further on the following pages) to support integrated routing. These TLVs tell Intermediate Systems (ISs) which Network Layer protocols are supported by other ISs and whether end stations running other protocols are reached. They also include any other required Network Layer, protocol-specific information.

Junos Support

Not all devices running Junos operating system support CLNP or CLNS routing. Check the platform-specific documentation for your specific product.

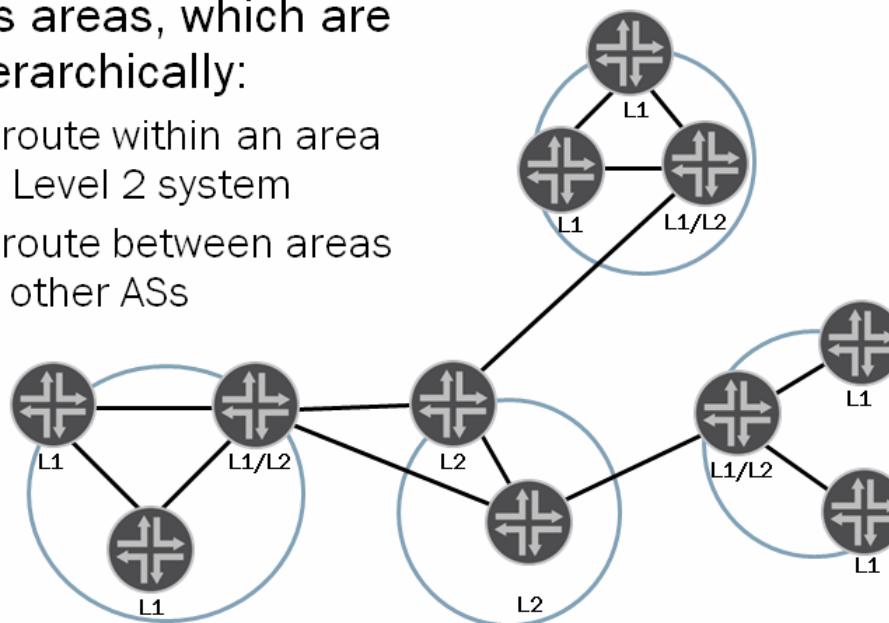
Operation of IS-IS

An IS-IS network is a single autonomous system (AS), also referred to as a routing domain, that consists of *end systems* (ESs) and *intermediate systems* (ISs). End systems are network entities that send and receive packets. Intermediate systems—which is the OSI term for a router—send and receive packets and relay, or forward, packets. IS-IS packets are referred to as PDUs.

IS-IS Areas

- A single AS can be divided into smaller groups referred to as areas, which are organized hierarchically:

- Level 1 ISs route within an area or toward a Level 2 system
- Level 2 ISs route between areas and toward other ASs



In IS-IS, a single AS can be divided into smaller groups referred to as areas. Routing between areas is organized hierarchically, allowing a domain to be divided administratively into smaller areas. IS-IS accomplishes this organization by configuring Level 1 and Level 2 ISs. Level 1 systems route within an area, and Level 2 ISs route between areas and toward other ASs. A Level 1 and Level 2 system routes within an area on one interface and between areas on another.

A Level 1 and Level 2 system sets the attached bit in the Level 1 PDUs that it generates into a Level 1 area to indicate that it is a Level 2-attached backbone router and that it can be used to reach prefixes outside the Level 1 area. Level 1 routers create a default route for interarea prefixes, which points to the closest (in terms of metrics) Level 1 and Level 2-attached router.

Both IS-IS and OSPF are link-state routing protocols with many similarities. Of course, differences exist as well. In IS-IS areas, a Level 1 and Level 2 router fulfills the same purpose as an area border router in OSPF. Likewise, the collection of Level 2 routers in IS-IS is the backbone, whereas Area 0 is the backbone in OSPF. However, in IS-IS, all routers are completely within an area, and the area borders are on the links, not on the routers. The routers that connect areas are Level 2 routers, and routers that have no direct connectivity to another area are Level 1 routers. An IS can be a Level 1 router, a Level 2 router, or both (an Level 1 and Level 2 router).

IS-IS and OSPF Common Features

- Maintain link-state databases and construct a shortest path tree
 - Dijkstra algorithm
- Use hello packets to form and maintain adjacencies
- Use a two-level hierarchy
- Provide for address summarization between areas
- Elect a designated router
- Have authentication capabilities

The inset lists the commonalities between IS-IS and OSPF.

IS-IS PDUs

IS-IS uses the following PDUs to exchange protocol information:

- *IS-IS Hello (IIH) PDUs*: IS-IS broadcasts these PDUs to discover the identity of neighboring IS-IS systems and to determine whether the neighbors are Level 1 or Level 2 ISs.
- *Link-state PDUs*: These PDUs contain information about the state of adjacencies to neighboring IS-IS systems. Link-state PDUs are flooded periodically throughout an area.
- *Complete sequence number PDUs (CSNPs)*: CSNPs contain a complete description of all link-state PDUs in the IS-IS database. IS-IS periodically sends CSNPs on all links, and the receiving systems use the information in the CSNP to update and synchronize their link-state PDU databases. The designated router multicasts CSNPs on broadcast links in place of sending explicit acknowledgments for each link-state PDU.
- *Partial sequence number PDUs (PSNPs)*: A receiver multicasts these PDUs when it detects that it is missing a link-state PDU or when its link-state PDU database is out of date. The receiver sends a PSNP to the system that transmitted the CSNP, effectively requesting that the missing link-state PDU be transmitted. That router, in turn, forwards the missing link-state PDU to the requesting router.
- *TLVs*: IS-IS PDUs use TLV encoding as the basic structure for all routing information. TLV encoding requires that the length of any field be defined explicitly when the field is used in a PDU.

Purpose of IIH PDUs

The purpose of the IIH PDU is to allow IS-IS routers to discover IS-IS neighbors on a link. Once the neighbors have been discovered and are adjacent, the IIH PDU acts as a keepalive to maintain the adjacency and to inform the neighbors of any changes in the adjacency parameters.

IIH PDU Types

Two kinds of IIH PDUs exist: LAN hello PDUs and point-to-point hello PDUs. The LAN hello PDUs can be divided further into Level 1 and Level 2 hello PDUs. The format of the two types of LAN hello PDUs is identical. Note that on broadcast networks IS-IS Level 1 and Level 2 hellos are coded with multicast address 01-80-C2-00-00-14 or 01-80-C2-00-00-15, respectively.

Hello Transmission

Routers send hello packets at a fixed interval on all interfaces to establish and maintain neighbor relationships. The hello interval field advertises this interval in the hello packet. By default, a DIS router sends hello packets every 3 seconds, and a non-DIS router sends hello packets every 9 seconds.

PDU Fields

The following list provides the details of the PDU fields:

- *Circuit type*: Defines the router as Level 1, Level 2, or a Level 1 and Level 2 router;
- *Source ID*: Identifies the system ID of the router that originated the hello PDU;
- *Holding time*: Specifies the period a neighbor should wait to receive the next hello PDU before declaring the originating router dead;
- *PDU length*: Specifies the length of the entire PDU in octets;
- *Priority*: Provides a value between 0 and 127 used for DIS election; and
- *LAN ID*: Identifies the system ID or the DIS plus one more octet (the pseudo-node ID) to differentiate this LAN ID from another LAN ID that might have the same designated router.

IS-IS Link-State PDUs

IS-IS sends link-state PDUs at regular intervals and when an IS discovers any of the following:

- Its link to a neighbor is down;
- It has a new neighbor; or
- The cost of a link to an existing neighbor has changed.

Once link-state PDUs are distributed appropriately, IS-IS runs the Dijkstra algorithm to compute optimal paths to each ES. This algorithm iterates on the length of a path, examining the link-state PDUs of all ISs working outward from the host IS. At the end of the computation, IS-IS forms a connectivity tree yielding the shortest paths, including all intermediate hops, to each IS.

PSNPs

A receiver multicasts PSNPs when it detects that it is missing a link-state PDU or when its link-state PDU database is out of date. The receiver sends a PSNP to the system that transmitted the CSNP, effectively requesting that the missing link-state PDU be transmitted. That router, in turn, forwards the missing link-state PDU to the requesting router.

CSNPs

CSNPs contain a complete description of all link-state PDUs in the IS-IS database. IS-IS sends CSNPs periodically on all links. The receiving systems use the information in the CSNP to update and synchronize their link-state PDU databases. The designated router multicasts CSNPs on broadcast links in place of sending explicit acknowledgments for each link-state PDU.

IS-IS Information Objects

IS-IS PDUs use TLV encoding as the basic structure for all routing information. TLV encoding requires that the length of any field be defined explicitly when a PDU uses that field. IS-IS ignores all unknown TLVs, making the protocol easily extensible.

Common TLVs

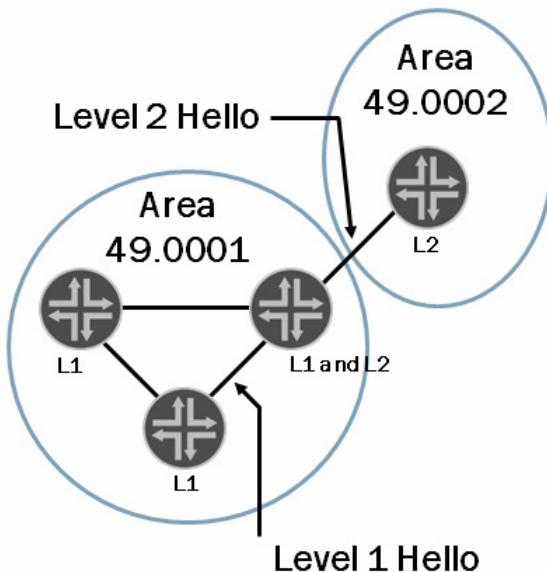
TLV Code	Defined in	Used for
1	ISO 10589	Area Addresses
2	ISO 10589	IS Neighbor Metrics
6	ISO 10589	Neighbor LAN ID
8	ISO 10589	Padding
9	ISO 10589	LSP Entries
10	ISO 10589	Authentication
128	RFC 1195	IP Prefix, Mask, and Metrics
129	RFC 1195	Protocols Supported
130	RFC 1195	IP External Information
132	RFC 1195	IP Interface Address
137	RFC 2763	Dynamic Hostname Mapping

The accompanying table shows the TLVs commonly used when using IS-IS as an IP routing protocol. Other TLVs are defined for MPLS traffic engineering that serve a purpose similar to OSPF Type 10 opaque link-state advertisements (LSAs).

Neighbors and Adjacencies

■ IS-IS adjacency rules:

- Level 1 routers never form an adjacency with a Level 2 router
 - The reverse is also true
- For Level 1 adjacencies, Area IDs must be the same
- For Level 2 adjacencies, Area IDs can be different



The graphic lists the rules Level 1 and Level 2 routers must follow when forming an adjacency.

DIS Election

The IS-IS DIS election process is achieved by assigning a Level 1 priority and a Level 2 priority on every IS-IS router interface, with a range of 0 through 127. The Junos OS uses a default priority of 64 for both levels. The router advertises its priority in the hello PDUs sent from each interface. The Level 1 priority is advertised in Level 1 hello PDUs, and the Level 2 priority is advertised in Level 2 hello PDUs. If the priority is 0, the router is ineligible to become the designated router. Interfaces to nonbroadcast networks automatically have a priority of 0. The router with the higher priority value becomes the designated router. In the event of a tie, the router with the numerically highest subnetwork point of attachment (SNPA), which is a fancy name for a MAC address, wins the election.

Pseudo-Node

IS-IS elects a designated router on broadcast and multi-access networks for the same reason as OSPF. Rather than having each router connected to the LAN advertise an adjacency with every other router connected to the LAN, the network itself is considered a router—a pseudo-node. Each router, including the designated router, advertises a single link to the pseudo-node. The designated router also advertises, as the representative of the pseudo-node, a link to all of the attached routers.

DIS Characteristics

Unlike OSPF, however, an IS-IS router attached to a broadcast, multi-access network establishes adjacencies with all of its neighbors on the network, not just the DIS. Each router multicasts its link-state PDUs to all of its neighbors, and the DIS uses a system of PDUs—named sequence number PDUs—to ensure that the flooding of link-state PDUs is reliable. Also unlike OSPF, no election of a backup designated router occurs in IS-IS. If the IS-IS DIS fails, a new DIS is elected. Another characteristic is that if a new router with a higher priority than the existing DIS becomes active, or if the new router has an equal priority and a higher SNPA (MAC address), it becomes the new DIS. When the DIS changes, a new set of link-state PDUs must be flooded.

IS-IS Metrics

IS-IS uses a single, required, default metric with a maximum path value of 1023. The metric is arbitrary and a network administrator typically assigns it.

Any single link can have a maximum value of 63; path metrics are calculated by summing link values. Maximum metric values were originally set at these levels to provide the granularity to support various link types while at the same time ensuring that the shortest-path algorithm used for route computation would be reasonably efficient.

IS-IS also defines three optional metrics, or costs:

- The *delay* cost metric reflects the amount of delay on the link;
- The *expense* cost metric reflects the communications cost associated with using the link; and
- The *error* cost metric reflects the error rate of the link.

IS-IS maintains a mapping of these four metrics to the quality-of-service (QoS) option in the CLNP packet header. IS-IS uses these mappings to compute routes through the internetwork.

Wide Metrics

IS-IS also has wide metrics. In most cases, a range of 1–63 is insufficient to properly distinguish between high-speed and low-speed links. You can set the metric up to 2^{24} (approximately 16 million). Wide metrics allow for the network diameter to be up to 256 hops. This diameter results in a maximum total path value of 2^{32} , or around 4.2 billion. The Junos OS sends both wide and standard metrics by default. However, to ensure a consistent topology, the software limits the wide metric value to 63 if it is also sending standard metrics. To benefit from the increased metric values, you must disable the sending of standard metrics on a per-level basis, with the **wide-metrics-only** statement.

Configuring IS-IS: Part 1

```
[edit protocols]
user@host# set isis interface ge-0/0/0.0 level 1 disable

[edit protocols]
user@host# set isis interface at-0/1/1.100 level 2 disable

[edit protocols]
user@host# show
isis {
    interface ge-0/0/0.0 {
        level 1 disable;
    }
    interface at-0/1/1.100 {
        level 2 disable;
    }
    interface lo0.0 {
        level 2 disable;
    }
}
```

By default, all interfaces specified as IS-IS interfaces are Level 1 and Level 2 interfaces. You might need to disable a particular level on a given interface.

Configuring IS-IS: Part 2

```
[edit]
user@host# show interfaces
ge-0/1/0 {
    unit 0 {
        family iso;
        family inet {
            address 10.0.24.1/24;
        }
    }
}
lo0 {
    unit 0 {
        family inet {
            address 192.168.2.1/32;
        }
        family iso {
            address 49.0001.0192.0168.0201.00;
        }
    }
}
```

For IS-IS to run on the router, you must enable IS-IS on the router, configure a network entity title (NET) on one of the router's interfaces (preferably the loopback interface, lo0), and configure the ISO family on all interfaces on which you want IS-IS to run.

The Junos OS supports the assignment of multiple ISO NETs to the router's loopback interface. Such a configuration might prove helpful when migrating two previously independent IS-IS domains into a single routing domain.

Displaying Interface Status

```

user@host> show isis interface ?
Possible completions:
<[Enter]>                                Execute this command
<interface-name>                           Interface name
brief                                         Show brief status
detail                                         Show detailed status

user@host> show isis interface
IS-IS interface database:
Interface      L CirID Level 1 DR      Level 2 DR      L1/L2 Metric
lo0.0          0 0x1 Disabled        Passive           0/0
so-0/1/0.0     2 0x1 Disabled        Point to Point   10/10
so-0/1/1.0     2 0x1 Disabled        Point to Point   10/10
so-0/1/2.0     2 0x1 Disabled        Point to Point   10/10

```

The **show isis interface** command displays the status of an interface. Use this command to ensure that IS-IS is configured correctly on the router. The output fields of this command are the following fields:

- *interface-name* (detail output only): Displays the name of the interface;
- Index (detail output only): Displays the interface index assigned by the Junos OS kernel;
- State (detail output only): Displays the internal implementation information;
- Circuit ID (detail output only): Displays the circuit identifier;
- Circuit type (detail output only): Displays the circuit type, which can be 1—Level 1 only, 2—Level 2 only, or 3—Level 1 and Level 2;
- LSP interval (detail output only): Displays the interface's link-state PDU interval;
- Sysid (detail output only): Displays the system identifier;
- Interface (brief output only): Displays the interface through which the adjacency is made.
- Level 1 DR/Level 2 DR (brief output only): Displays the Level 1 or Level 2 DIS;
- L1/L2 Metric: Displays the interface's metric for Level 1 and Level 2. If no information is present, the metric is 0;
- Adjacencies (detail output only): Displays the number of adjacencies established on the interface;
- Priority (detail output only): Displays the priority value for this interface;
- Metric (detail output only): Displays the metric value for this interface;
- Hello(s) (detail output only): Displays the interface's hello interval; and
- Hold(s) (detail output only): Displays the interface's hold time.

Displaying IS-IS Database Entries

```
user@host> show isis database
IS-IS level 1 link-state database:
LSP ID Sequence Checksum Lifetime Attributes
host.00-00 0x3 0x48ba 1132 L1 L2
 1 LSPs

IS-IS level 2 link-state database:
LSP ID Sequence Checksum Lifetime Attributes
Denver.00-00 0x6 0xb978 1154 L1 L2
host.00-00 0x4 0x140c 1154 L1 L2
wash-dc.00-00 0x1d7 0x7f37 683 L1 L2
Atlanta.00-00 0x1d3 0xe603 1024 L1 L2
Atlanta.02-00 0x9 0xea81 1175 L1 L2
Houston.00-00 0x1c8 0x8b1c 677 L1 L2
Dallas.00-00 0x1ca 0x1571 1103 L1 L2
NewYork.00-00 0x1dd 0x178a 779 L1 L2
 8 LSPs
```

The **show isis database** command displays a brief view of all the link-state PDUs in the IS-IS link-state database. The output fields of this command are the following:

- **LSP ID:** Displays the link-state PDU identifier;
- **Sequence:** Displays the sequence number of the link-state PDU;
- **Checksum:** Displays the checksum value of the link-state PDU;
- **Lifetime (secs):** Displays the remaining lifetime of the link-state PDU, in seconds;
- **IP prefix:** (detail and extensive output only) Displays the prefix advertised by the link-state PDU;
- **IS neighbor:** (detail output only): Displays an IS-IS neighbor of the advertising system; and
- **Metric (detail and extensive output only):** Displays the metric of the prefix or neighbor.

Displaying IS-IS Adjacency Status

```
user@host> show isis adjacency ?
Possible completions:
<[Enter]> Execute this command
brief Show brief status
detail Show detailed status
system-id Display entries for specified system

user@host> show isis adjacency
IS-IS adjacency database:
Interface System L State Hold (secs) SNPA
so-0/1/2.0 Denver 2 Up 24
so-0/1/3.0 SanFran 2 Up 28
so-0/2/2.0 Toronto 2 Up 23
so-0/2/3.0 Amsterdam 2 Up 26
```

The **show isis adjacency** command displays the status of IS-IS adjacencies. The output fields of this command are the following:

- **Interface:** Displays the interface through which the neighbor is reachable.
- **System:** (brief output only): Displays the system identifier, printed as a name if possible.
- **L:** Displays the level, which can be 1—Level 1 only; 2—Level 2 only; or 3—Level 1 and Level 2. An exclamation point (!) preceding the level number indicates that the adjacency is missing an IP address.
- **State:** Displays the state of the adjacency. It can be Up, Down, New, One-way, Initializing, or Rejected.

- Hold (secs) (brief/standard output only): Displays the remaining hold time of the adjacency. Note that the **show isis adjacency** command returns brief output by default.
- SNPA (brief output only): Displays the SNPA (MAC address of the next hop).

Displaying Detailed Adjacency Information

```
user@host> show isis adjacency detail
IS-IS adjacency database:

Denver
  Interface: so-0/1/2.0, Level: 2, State: Up, Expires in 25 secs
  Priority: 0, Up/Down transitions: 1, Last transition: 00:15:27 ago
  Circuit type: 2, Speaks: IP
  IP addresses: 10.0.18.2
```

The **show isis adjacency detail** command displays detailed IS-IS adjacency information. The output fields of this command are the following:

- Expires in (detail output only): Displays the time until the adjacency expires, in seconds;
- Priority (detail output only): Displays the priority to become the DIS;
- Up/Down transitions (detail output only): Displays the count of adjacency status changes from up to down or from down to up;
- Last transition (detail output only): Displays the time of the last up or down transition;
- Circuit type (detail output only): Displays the bit mask of levels on this interface, which can be 1—Level 1 router, 2—Level 2 router, or 1/2—both Level 1 and Level 2 routers;
- Speaks (detail output only): Displays the protocols supported by the neighbor; and
- IP addresses (detail output only): Displays the IP address of the neighbor.

Clearing and Restarting IS-IS Adjacencies

```
user@host> show isis adjacency
IS-IS adjacency database:
Interface      System      L State      Hold (secs)  SNPA
so-0/1/2.0     Denver      2 Up        24
so-0/1/3.0     SanFran    2 Up        28
so-0/2/2.0     Toronto    2 Up        23
so-0/2/3.0     Amsterdam  2 Up        26

user@host> clear isis adjacency Toronto

user@host> show isis adjacency
IS-IS adjacency database:
Interface      System      L State      Hold (secs)  SNPA
so-0/1/2.0     Denver      2 Up        22
so-0/1/3.0     SanFran    2 Up        26
so-0/2/3.0     Amsterdam  2 Up        24
```

The **clear isis adjacency** command clears and restarts the adjacency process with a particular IS-IS neighbor.

Displaying SPF Operation

```
user@host> show isis spf log
IS-IS level 1 SPF log:
Start time           Elapsed (secs) Count Reason
Tue Apr 17 16:13:26      0.000039   1 Reconfig
Tue Apr 17 16:13:32      0.000062   1 Updated LSP host.00-00
Tue Apr 17 16:28:26      0.000064   1 Periodic SPF

IS-IS level 2 SPF log:
Start time           Elapsed (secs) Count Reason
Tue Apr 17 16:13:26      0.000025   1 Reconfig
Tue Apr 17 16:13:32      0.000051   1 Updated LSP host.00-00
Tue Apr 17 16:13:57      0.000087   2 New adjacency Denver on so-0/1/2.0
Tue Apr 17 16:14:03      0.000241   6 Updated LSP Atlanta.00-00
Tue Apr 17 16:26:38      0.000298   1 Periodic SPF
```

The **show isis spf log** command displays the elapsed time to perform SPF calculations and the reasons why they were triggered. The output fields of this command are the following:

- **Node:** Displays the system ID of a node;
- **Metric:** Displays the metric to the node;
- **Interface:** Displays the interface of the next hop;
- **Via:** Displays the system ID of the next hop;
- **SNPA:** Displays the SNPA (MAC address of the next hop);
- **Start time (log output only):** Displays the time that the SPF computation started;
- **Elapsed time (log output only):** Displays the length of time required to complete the SPF computation in seconds;
- **Count (log output only):** Displays the number of times the SPF was triggered; and
- **Reason (log output only):** Displays the reason that the SPF computation was completed.

Displaying IS-IS Statistics

```
user@host> show isis statistics
IS-IS statistics for host:
PDU type    Received    Processed    Drops    Sent    Rexmit
LSP          17          17          0        2        0
I IH         190         190         0        571      0
CSNP         99          99          0        294      0
PSNP         3           3           0        12       0
Unknown      0           0           0        0        0
Totals       309         309         0        879      0

Total packets received: 309 Sent: 879

SNP queue length: 0 Drops: 0
LSP queue length: 0 Drops: 0
SPF runs: 17
Fragments rebuilt: 5
LSP regenerations: 2
Purges initiated: 0
```

The **show isis statistics** command displays statistics about IS-IS traffic. The output fields of this command are the following:

- **PDU type:** Displays the PDU type.
- **Received:** Displays the number of PDUs received since IS-IS started or since the statistics were zeroed.
- **Processed:** Displays the number of PDUs received minus the number dropped.
- **Drops:** Displays the number of dropped PDUs.

- Sent: Displays the number of PDUs transmitted since IS-IS started or since the statistics were zeroed.
- Rexmit: Displays the number of PDUs retransmitted since IS-IS started or since the statistics were zeroed.
- Total packets received/sent: Displays the total number of PDUs received and transmitted since IS-IS started or since the statistics were zeroed.
- SNP queue length: Displays the number of CSNPs and PSNPs sitting on the sequence number packets (SNP) queue waiting for processing. This value is almost always 0.
- LSP queue length: Displays the number of link-state PDUs sitting on the link-state PDU queue waiting for processing. This value is almost always 0.
- SPF runs: Displays the number of SPF calculations performed. If this number is incrementing rapidly, it indicates that the network is unstable.
- Fragments rebuilt: Displays the number of link-state PDU fragments that the local system has computed.
- LSP regenerations: Displays the number of link-state PDUs that were regenerated. A link-state PDU is regenerated when it is nearing the end of its lifetime and it has not changed.
- Purges initiated: Displays the number of purges that the system initiated. A purge is initiated if the software decides that a link-state PDU must be removed from the network.

Displaying IS-IS Routes

```

user@host> show isis route
IPv4/IPv6 Routes
-----
IS-IS routing table                                Current version: L1: 3 L2: 5
Prefix      L Version Metric Type Interface     Via
10.0.0.0/24 2      5      20 int   so-0/1/2.0  Denver
10.0.1.0/24 2      5      20 int   so-0/1/2.0  Denver
10.0.2.0/24 2      5      30 int   so-0/1/2.0  Denver
10.0.8.0/24 2      5      30 int   so-0/1/2.0  Denver
.
.
user@host> show route protocol isis

inet.0: 64 destinations, 64 routes (64 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.0.0/24      *[IS-IS/18] 00:00:59, metric 30
                  > to 10.0.21.1 via fe-0/0/2.0
10.0.1.0/24      *[IS-IS/18] 00:00:59, metric 30
                  > to 10.0.21.1 via fe-0/0/2.0
10.0.2.0/24      *[IS-IS/18] 00:00:59, metric 40
                  to 10.0.21.1 via fe-0/0/2.0
                  > to 10.0.22.2 via so-0/1/1.0

```

The **show isis route** command displays the routes in the IS-IS routing table. The output fields of this command are the following:

- Current version: Displays the number of the current version of the IS-IS routing table.
- L1: Displays the version of the Level 1 SPF that was run.
- L2: Displays the version of the Level 2 SPF that was run.
- Prefix: Displays the destination of the route.
- L: Displays the level, which can be 1—Level 1 only; 2—Level 2 only; and 3—Level 1 and Level 2.
- Version: Displays the version (or run) of SPF that generated the route.
- Metric: Displays the metric value associated with the route.
- Type: Displays the metric type. It can be int (internal) or ext (external).

- **Interface:** Displays the interface to the next hop.
- **Via:** Displays the system ID of the next hop, displayed as a name if possible.

Displaying Detailed IS-IS Database Information

```
lab@Tokyo> show isis database extensive
IS-IS level 1 link-state database:

Tokyo.00-00 Sequence: 0x1, Checksum: 0x1c90, Lifetime: 864 secs
  IP prefix: 192.168.20.0/24 Metric: 0 External Up
  IP prefix: 192.168.21.0/24 Metric: 0 External Up
  IP prefix: 192.168.22.0/24 Metric: 0 External Up
  IP prefix: 200.0.0.0/24 Metric: 0 External Up

Header: LSP ID: Tokyo.00-00, Length: 140 bytes
  Allocated length: 1492 bytes, Router ID: 192.168.20.1
  Remaining lifetime: 864 secs, Level: 1, Interface: 0
  Estimated free bytes: 1352, Actual free bytes: 1352
  Aging timer expires in: 864 secs
  Protocols: IP, IPv6

Packet: LSP ID: Tokyo.00-00, Length: 140 bytes, Lifetime : 1200 secs
  Checksum: 0x1c90, Sequence: 0x1, Attributes: 0x3 <L1 L2>
  NLPIID: 0x83, Fixed length: 27 bytes, Version: 1, Sysid length: 0 bytes
  Packet type: 18, Packet version: 1, Max area: 0

TLVs:
  Area address: 49.0001 (3)
  Speaks: IP
  Speaks: IPv6
  IP router id: 192.168.20.1
  IP address: 192.168.20.1
  Hostname: Tokyo
  IP external prefix: 192.168.20.0/24, Internal, Metric: default 0, Up
```

The **show isis database extensive** command provides detailed output for the contents of the IS-IS link-state database. The output fields of this command are the following:

- **LSP ID:** Displays the link-state PDU identifier;
- **Sequence:** Displays the sequence number of the link-state PDU;
- **Checksum:** Displays the checksum value of the link-state PDU;
- **Lifetime (in seconds):** Displays the remaining lifetime of the link-state PDU, in seconds;
- **IP prefix (detail and extensive output only):** Displays the prefix advertised by this link-state PDU;
- **IS neighbor (detail output only):** Displays an IS-IS neighbor of the advertising system; and
- **Metric (detail and extensive output only):** Displays the metric of the prefix or neighbor.

The command output then displays detailed packet content information.

IS-IS Tracing

To perform debugging functions on the IS-IS routing process, use the Junos OS traceoptions feature. The trace output (debug information) is directed to the named log file, which is stored in the /var/log directory on the router's hard drive. You can view the log file using the **monitor start** or **show log** operational mode commands. In addition to specifying the trace file, you also must tell the router what information you want to trace. You accomplish this task by specifying one or more **flag** keywords.

Although you can direct tracing only to a single file, you can trace many options by using the **flag** keyword multiple times. In addition, you can add granularity by using the **detail**, **receive**, and **send** flag modifiers.

```
[edit protocols isis]
user@host# show
traceoptions {
    file isis-trace;
    flag error detail;
    flag hello detail;
    flag lsp detail;
}
```

Several tracing flags for IS-IS are available, as shown in the following output:

all	Trace everything
csn	Trace complete sequence number (CSN) packets
error	Trace errored packets
general	Trace general events
graceful-restart	Trace graceful restart events
hello	Trace hello packets
lsp	Trace link-state packets
lsp-generation	Trace LSP generation
normal	Trace normal events
packets	Trace IS-IS packets
policy	Trace policy processing
psn	Trace partial sequence number (PSN) packets
route	Trace routing information
spf	Trace SPF events
state	Trace state transitions
task	Trace routing protocol task processing
timer	Trace routing protocol timer processing

IP Configuration Is Not Necessary

When establishing adjacencies in OSPF, all routers on a link must agree upon the IP subnet to which they belong, except on point-to-point links, which can be unnumbered or use /32 addressing. This agreement is not necessary with IS-IS. IS-IS does not rely on IP; it simply carries IP information within its TLVs. Thus, an IS-IS adjacency can possibly come up, even if the routers on a link do not agree on the IP subnet to which they belong. As a result, a router can have an IS-IS adjacency establish and not be able to ping its neighboring routers.

Troubleshooting No Adjacency

- If no adjacency, check for the following problems:
 - Physical Layer or Data Link Layer connectivity
 - Mismatched areas (if a Level 1 router) and levels
 - Failure to support minimum MTU of 1492
 - Lack of, or malformed, ISO-NET
 - No NET configured
 - Failure to include lo0 as an IS-IS interface

The inset provides a checklist to use when troubleshooting IS-IS adjacency problems.

Review Questions

1. What is the purpose of the IS-IS DIS?
2. What is the rationale behind a multi-level IS-IS design?
3. How would you configure a Junos OS device to run only Level 2 IS-IS on an interface?
4. List and describe three CLI commands you can use to monitor and troubleshoot the IS-IS protocol.

Answers

1.

On broadcast multi-access networks, a single router is elected as the DIS. The DIS creates the pseudo-node and acts on behalf of the pseudo-node.

2.

IS-IS introduces hierarchy in a routing domain through levels. Implementing a multi-level IS-IS design facilitates scalability and growth.

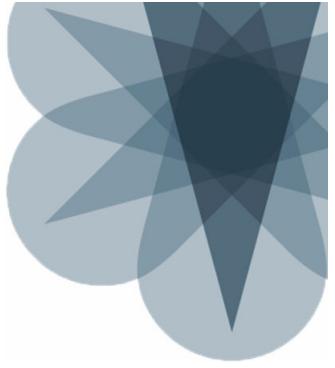
3.

You would need to disable Level 1 on the interface participating in IS-IS to configure a Junos OS device to run only Level 2 IS-IS on an interface.

4.

Although several operational mode commands exist that can be used to monitor and troubleshoot IS-IS, the following are some commonly used commands:

- a. The **show isis adjacency detail** command displays detailed IS-IS adjacency information;
- b. The **show isis statistics** command displays statistics about IS-IS traffic; and
- c. The **show isis route** command displays the routes in the IS-IS routing table.



JB7-GGD'Ghi Xm̄ i JXYÍ DUfh%

Appendix C: Routing Information Protocol

This Chapter Discusses:

- The operation of RIP;
- RIP configuration in the Junos operating system; and
- Using operational-mode commands to monitor and troubleshoot RIP operation.

RIP Is an Interior Gateway Protocol

RIP is an interior gateway protocol (IGP) used *within* an autonomous system. RIP advertises routes between devices within the autonomous system.

Two Versions

Two versions of RIP exist: RIPv1 and RIPv2. RIPv2 did not change the protocol; it expanded the capabilities of it. RFC 1058 defines RIPv1; RFC 2453 defines RIPv2. RIPv2 is the newer of the two protocols. RIPv1 and RIPv2 can interoperate if RIPv1 ignores all fields that must be zero. RIPv2 allows more information to be included in RIP packets and provides a simple authentication mechanism. It also supports variable-length subnet masking (VLSM).

RIP is based on the ROUTED program, originally distributed with version 4.3 of the Berkeley Software Division UNIX software. In most UNIX systems, the ROUTED routing daemon dynamically builds the routing table based on information it receives through RIP updates. When routing starts, it issues a request for routing updates and then listens for responses to its request. When a system configured to supply RIP information hears the request, it responds with an update packet based on the information in its routing table. The update packet contains the destination addresses from the routing table and the routing metric associated with each destination. Update packets are not just issued in response to requests, they are also issued periodically to keep routing information accurate.

Distance-Vector Routing Protocol

- Distance-vector routing protocol
- Hop count is used as the metric for path selection, based on the Bellman-Ford distance-vector routing algorithm
 - Maximum allowable hop count is 15
- Routing updates are broadcast every 30 seconds

When determining the best path to a destination, RIP uses a combination of hop count (that is, distance) and the next hop (that is, vector).

Hop Count

The longest network path in an RIP network is 15 hops between the source and the destination. The assumption here is that the metric count for each network or hop has a cost of 1. The 15-hop limitation exists to prevent the creation of an infinitely long network path. With an upper limit of 15 hops, the protocol treats a metric of 16, referred to as *infinity*, to mean that the destination network is unreachable, referred to as network *unreachable*.

When the routing daemon starts on a newly initialized router, it first determines all interfaces that are initialized and then sends them a request packet to each interface asking the router on the other end of the link to send its complete routing table.

Routing Updates

Upon receiving an update from another router, the requesting router validates the response and might or might not update its routing table. If updating is required, the update can take the form of adding a route to the table, modifying an existing entry, or deleting an existing entry. Upon receipt of all replies from connected routers, the requesting router builds and updates its routing table.

Each entry in the routing table consists of:

- Network reachability information, the network ID, and the metric;
- Next-hop information;
- The interface through which a packet must pass; and
- A timer indicating the age of a routing entry.

Every 30 seconds, RIP sends all or part of the router's routing table to each of its neighbor's directly connected routers. The routing table is either broadcast to its neighbors on an Ethernet segment or sent to the other end of a point-to-point link. These periodic updates allow a router running RIP to respond to network changes.

RIP also supports triggered updates. A triggered update occurs when a metric changes on a route and can include only the changed entry or entries.

Update Process

A request message asks neighboring routers to send an update, and a response message carries the update from the neighboring routers. When a router receives an update from a neighbor, RIP adds the cost of the network over which the update is received to the advertised metric. The new value is used when comparing routes. RIP stores unknown routes immediately. If a router must advertise more than 25 routes, it must send out an additional response message.

Route Updates

RIP evaluates known routes by comparing the metric, or cost, of the route presently in the table to the metric of the received route with the following decisions:

- If the cost is lower, RIP adds the new route to the table.
- In cases where the router advertising the network is the same one that originally provided the new route, RIP adopts the route, even where the metric is larger.
- If the advertised hop count is higher than the recorded hop count and the recorded next-hop router originated the update, RIP marks the route as unreachable for a specific hold-down period. At the end of the hold-down period, if the same neighbor is still advertising the higher hop count, RIP accepts the new metric.
- The router can receive both RIPv1 and RIPv2 update messages, with 25 route entries per message. RIP uses timers to enable the router to make the decisions described above.

Backward Compatibility

RIPv2 is totally *backward compatible* with RIPv1. If a RIPv2 router receives a RIPv1 request message, it should respond with a RIPv1 response message. If you configure the router to send only RIPv2 messages, it should not respond to a RIPv1 request message.

Authentication per Message

Authentication is possible with RIPv2. The authentication scheme for RIPv2 uses the space of an entire RIP entry. If the address family identifier of the first—and only the first—entry in the message is 0xFFFF, the remainder of the entry contains the authentication. Thus, at most, 24 RIP entries in the remainder of the message can exist. If authentication is not in use, then no entries in the message should have an address family identifier of 0xFFFF. Currently, the only *authentication* type is a simple password, and it is type 2. The remaining 16 octets contain the plain-text password. If the password is under 16 octets, it must be left-justified and padded to the right with nulls (0x00).

Multicast Updates

Multicasting was added to reduce unnecessary processing of RIP updates by hosts who are not involved in RIPv2 processing. The IP multicast address is 224.0.0.9. On nonbroadcast multi-access networks, such as Frame Relay or ATM, you can use unicast addressing.

Prefix Length

RIPv2 can perform *classless routing*, where the prefix length is included in the RIP updates. Another benefit of having a destination prefix length associated with an update is that you can use *variable-length destination prefixes*, thus eliminating the requirement that all destination prefixes in the Internet have the same length.

Next-Hop Address

With RIPv2, updates also include the *next-hop address*, providing functionality similar to an ICMP redirect. The purpose of the next-hop field is to eliminate packets being routed through extra hops in the system. This field is particularly useful when RIP is not running on all the routers on a network.

RIPv1/v2 Interoperability

RFC 1723 defines a compatibility mode switch with the following four settings, which allow versions 1 and 2 to interoperate:

- *RIP-1*: Only RIPv1 messages transmit.
- *RIP-1 Compatibility*: Causes RIPv2 to broadcast its messages instead of multicasting them so that RIPv1 hosts can receive them.
- *RIP-2*: RIPv2 messages are multicast to destination address 224.0.0.9.
- *None*: No updates are sent.

RIP Limitations

Due to its age and design, RIP has several inherent limitations.

- The designers believe that the basic protocol design is inappropriate for larger networks. Assuming a cost of 1, the protocol is limited to networks whose longest path involves 15 hops. If the system administrator chooses to use larger costs, the upper bound of 15 can become a problem easily.
- Updates occur every 30 seconds by default, and the entire routing table is sent in an update. In addition, a triggered update, resulting from a network change, occurs immediately and involves sending the entire routing table.
- Poison reverse aids in network convergence, but it also increases the size of update messages, which include valid and poisoned routes.
- The protocol depends upon counting to infinity to resolve certain unusual situations. Resolving a loop with counting to infinity involves time because a route's metric is increased by two in each update interval, and the loop is only broken when the count reaches 16.
- This protocol uses fixed metrics to compare alternative routes. This method is not appropriate, however, for situations where routes must be chosen based on real-time parameters, such as measured delay, reliability, or load.
- Broadcasting between neighbors forces processing of packets by each host, whether involved in the routing process or not.

- RIP cannot distinguish between subnets. RIPv1 cannot advertise destination prefix lengths; thus, all networks involved in an RIPv1 network must use the same mask.
- RIP provides no authentication mechanism, so a RIP router accepts all RIP-compliant updates.
- Convergence on the network can be slow, leading to loops and suboptimal paths.

Junos RIP Support

- RIPv1
- RIPv2
- Peer groups
 - Neighbors defined in peer group
- Default setting of no route export
 - Need export policy to readvertise RIP or to advertise other routes
- Default preference of 100
- Modification of metrics in or out

The graphic shows aspects of RIPv1 and RIPv2 supported by the Junos OS.

Minimum RIP Configuration

```
protocols {
    rip {
        group group-name {
            neighbor interface-name;
        }
    }
}
```

The minimum configuration, as shown in the graphic, starts RIP on the interface specified. After you commit this configuration, this router will understand and evaluate any RIP routes advertised by another router on the same network segment as the interface specified.

Advertising and Creating Export Policy

```
policy-options {
    policy-statement statics-to-rip {
        from protocol static;
        then accept;
    }
}
```

By default, a router with the configuration from the previous page cannot create and advertise any RIP routes that it has not learned from another router. To create and advertise RIP routes to its neighbors, you must configure a router using policy to advertise the specific routes.

Exporting Policy

```

protocols {
    rip {
        group rip-neighbors {
            export statics-to-rip;
            neighbor fe-0/0/0.0;
            neighbor fe-0/0/1.0;
        }
    }
}

```

The example in the graphic uses policy to advertise this router's static routes using RIP.

State of RIP Interfaces

Neighbor	State	Source Address	Destination Address	Send Mode	Receive Mode	In Met
fe-0/0/1.0	Up	10.0.31.2	224.0.0.9	mcast	both	1
fe-0/0/0.0	Up	10.0.13.2	224.0.0.9	mcast	both	1

The **show rip neighbor** command lists interfaces currently running RIP. The output fields of this command are:

- Neighbor: Displays the name of the RIP neighbor.
- State: Displays the state of the connection. The interface can be either up or down.
- Source Address: Displays the source address.
- Destination Address: Displays the destination of RIP updates, which can be either broadcast or multicast.
- Send Mode: Displays the send options, which can be broadcast, multicast, none, or version 1.
- Receive Mode: Displays the type of packets to accept, which can be both, none, version 1, or version 2.
- In Met: Displays the metric added to incoming routes when advertising into RIP routes that were learned from other protocols.

RIP Routes

- Show all routes learned via RIP using the **show route protocol rip** command

```

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

172.16.0.0/16      [RIP/100] 00:07:02, metric 2
                  > to 10.0.13.1 via fe-0/0/0.0
192.168.8.1/32    *[RIP/100] 00:07:02, metric 2
                  > to 10.0.13.1 via fe-0/0/0.0
224.0.0.9/32      *[RIP/100] 00:11:25, metric 1
...

```

To view the routes in the unicast routing table, issue the **show route protocol rip**. This command filters your routing table and shows only entries learned using RIP.

Advertised RIP Routes

```
user@host> show route advertising-protocol rip
10.0.21.1
inet.0: 12 destinations, 12 routes (11 active, 1
holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
3.0.0.0/8          *[Static/5] 00:10:56
                  Reject
30.0.0.0/16        *[Static/5] 00:12:20
                  Reject
```

Use the **show route advertising-protocol rip neighbor** command to view the routes that are advertised out a RIP interface as a result of your RIP export policy. The **neighbor** argument in this command takes the form of the IP address assigned to the local router's RIP interface.

Note that to help guard against routing loops, the RIP protocol requires that a router continue to advertise a newly unreachable prefix with an infinite metric for a period of time after the route's status changes.

This poison reverse behavior can make it seem as though export policy changes are not taking effect because you might see the continued advertisement of prefixes that the current export policy should be rejecting when using the **show route advertising-protocol rip neighbor** command. When you adjust RIP export policy to reject routes previously being accepted, you should expect to see ongoing advertisement of the rejected prefixes for three RIP update cycles (approximately 90 seconds).

Received RIP Routes

```
inet.0: 17 destinations, 18 routes (17 active, 0 holddown,
0 hidden)
+ = Active Route, - = Last Active, * = Both
172.20.4.0/24      *[RIP/100] 00:01:01, metric 2
                  > to 10.100.3.2 via fe-0/0/0.0
192.168.28.0/24    *[RIP/100] 00:01:01, metric 2
                  > to 10.100.3.2 via fe-0/0/0.0
```

Issue a **show route receive-protocol rip neighbor** command to view the routes being received on a RIP interface from the neighbor address specified. Note that the **neighbor** argument, in this case, is the IP address of the remote RIP neighbor.

Also note that the routes are displayed *before* your RIP import policy has a chance to manipulate their attributes, but *after* the RIP protocol has discarded nonbest routes. To confirm the operation of your RIP import policy, display the properties of the routes as they reside in the routing table with a **show route protocol rip** command.

RIP Statistic Gathering

```

user@host> show rip statistics

RIP info: port 520; update interval 30s; holddown 180s; timeout
120s.

      rts learned   rts held down   rqsts dropped   resps dropped
                  1                   1                   0                   0

fe-0/0/0.0: 1 routes learned; 3 routes advertised

Counter          Total    Last 5 min    Last minute
-----
Updates Sent      28        11            2
Triggered Updates Sent  1        0            0
Responses Sent     0        0            0
Bad Messages       0        0            0
RIPv1 Updates Received  0        0            0
RIPv1 Bad Route Entries  0        0            0
RIPv1 Updates Ignored  0        0            0
RIPv2 Updates Received  14       11            3
RIPv2 Bad Route Entries  0        0            0
RIPv2 Updates Ignored  0        0            0
Authentication Failures  0        0            0
RIP Requests Received  0        0            0
RIP Requests Ignored  0        0            0

```

The **show rip statistics** command displays general RIP protocol statistics. The output fields of this command are:

- **RIP info:** Displays the information about RIP on the specified interface.
- **port:** Displays the UDP port number used for RIP.
- **update interval:** Displays the number of seconds since the last update.
- **holddown:** Displays the hold-down interval, in seconds.
- **timeout:** Displays the timeout interval, in seconds.
- **bad msgs:** Displays the number of bad messages received.
- **rts learned:** Displays the number of routes learned through RIP.
- **rts held down:** Displays the number of routes held down by RIP.
- **rqst dropped:** Displays the number of request messages dropped by RIP.
- **resp dropped:** Displays the number of response messages dropped by RIP.
- **Counter:** Displays the list of counter types.
- **Total:** Displays the total number of packets for the selected counter.

See the Command Reference Guide at <http://www.juniper.net/customers/support/> for a description of all the various counters.

RIP Tracing

To perform debugging functions on the RIP routing process, use the the Junos OS **traceoptions** function. The trace output (debug information) is directed to the named log file, which is stored in the **/var/log** directory on the router's hard drive. You can view the log file using the **monitor start** or **show log** operational mode commands.

In addition to specifying the trace file, you also must tell the router what information you want to trace. You accomplish this by specifying one or more **flag** keywords.

Although you can only direct tracing to a single file, you can trace many options by using the **flag** keyword multiple times. In addition, you can add granularity by using the **detail**, **receive**, and **send** flag modifiers.

```

[edit protocols rip]
user@host# show
traceoptions {
    file rip-trace;
    flag error detail;
    flag update detail;
}

```

Available tracing flags for RIP include:

all	Trace everything
auth	Trace RIP authentication
error	Trace RIP errors
expiration	Trace RIP route expiration processing
general	Trace general events
holddown	Trace RIP hold-down processing
normal	Trace normal events
packets	Trace all RIP packets
policy	Trace policy processing
request	Trace RIP information packets
route	Trace routing information
state	Trace state transitions
task	Trace routing protocol task processing
timer	Trace routing protocol timer processing
trigger	Trace RIP triggered updates
update	Trace RIP update packets

Review Questions

1. By default, which RIP version does the Junos OS use?
2. Do you need export policy to have RIP export RIP routes learned from other neighbors?
3. Which version of RIP supports VLSM?
4. Which command would you use to verify that your router is sending and receiving RIP updates?

Answers

1.

The Junos OS uses RIPv2 by default.

2.

Yes, you do need export policy.

3.

RIPv2 supports VLSM.

4.

You would use the **show rip statistics** command.