FRR User Manual

Release latest

FRR

Apr 30, 2019

Contents

1 Overview **1**
- 1.1 About FRR . . . 1
- 1.2 System Architecture . . . 2
- 1.3 Supported Platforms . . . 2
- 1.4 Supported RFCs . . . 3
- 1.5 How to get FRR . . . 4
- 1.6 Mailing Lists . . . 4
- 1.7 Bug Reports . . . 4

2 Installation **5**
- 2.1 Configure the Software . . . 5
- 2.2 Build the Software . . . 9
- 2.3 Install the Software . . . 9

3 Basic commands **11**
- 3.1 Config Commands . . . 11
- 3.2 Terminal Mode Commands . . . 14
- 3.3 Common Invocation Options . . . 14
- 3.4 Loadable Module Support . . . 15
- 3.5 Virtual Terminal Interfaces . . . 15

4 VTY shell **19**
- 4.1 Permissions and setup requirements . . . 19
- 4.2 Integrated configuration mode . . . 20

5 Filtering **23**
- 5.1 IP Access List . . . 23
- 5.2 IP Prefix List . . . 23

6 Route Maps **27**
- 6.1 Route Map Command . . . 28
- 6.2 Route Map Match Command . . . 28
- 6.3 Route Map Set Command . . . 29
- 6.4 Route Map Call Command . . . 30
- 6.5 Route Map Exit Action Command . . . 30
- 6.6 Route Map Examples . . . 30

7 IPv6 Support **31**
7.1 Router Advertisement . . . 31

8 Kernel Interface **35**

9 SNMP Support **37**
9.1 Getting and installing an SNMP agent . . . 37
9.2 AgentX configuration . . . 37
9.3 SMUX configuration . . . 38
9.4 MIB and command reference . . . 39
9.5 Handling SNMP Traps . . . 39

10 Zebra **43**
10.1 Invoking zebra . . . 43
10.2 Configuration Addresses behaviour . . . 44
10.3 Interface Commands . . . 44
10.4 Static Route Commands . . . 46
10.5 VRF (Virtual Routing and Forwarding) . . . 47
10.6 Multicast RIB Commands . . . 49
10.7 zebra Route Filtering . . . 50
10.8 zebra FIB push interface . . . 50
10.9 zebra Terminal Mode Commands . . . 51

11 BGP **53**
11.1 Starting BGP . . . 53
11.2 BGP router . . . 53
11.3 BGP MED . . . 55
11.4 BGP network . . . 58
11.5 BGP Peer . . . 60
11.6 BGP Peer Group . . . 62
11.7 BGP Address Family . . . 62
11.8 Autonomous System . . . 63
11.9 BGP Communities Attribute . . . 63
11.10 BGP Extended Communities Attribute . . . 68
11.11 BGP Large Communities Attribute . . . 69
11.12 BGP VRFs . . . 70
11.13 Displaying BGP information . . . 72
11.14 Capability Negotiation . . . 73
11.15 Route Reflector . . . 74
11.16 Route Server . . . 74
11.17 BGP Regular Expressions . . . 77
11.18 How to set up a 6-Bone connection . . . 77
11.19 Dump BGP packets and table . . . 77
11.20 BGP Configuration Examples . . . 78
11.21 Configuring FRR as a Route Server . . . 82
11.22 Prefix Origin Validation Using RPKI . . . 92
11.23 Flowspec . . . 95

12 Babel **101**
12.1 Configuring babeld . . . 101
12.2 Babel configuration . . . 101
12.3 Babel redistribution . . . 103
12.4 Show Babel information . . . 103
12.5 Babel debugging commands . . . 103

13 EIGRP **105**
13.1 Starting and Stopping eigrpd . . . 105
13.2 EIGRP Configuration . . . 106
13.3 How to Announce EIGRP route . . . 106
13.4 Show EIGRP Information . . . 107
13.5 EIGRP Debug Commands . . . 107

14 ISIS **109**
14.1 Configuring isisd . . . 109
14.2 ISIS router . . . 109
14.3 ISIS Timer . . . 110
14.4 ISIS region . . . 111
14.5 ISIS interface . . . 111
14.6 Showing ISIS information . . . 112
14.7 Traffic Engineering . . . 113
14.8 Debugging ISIS . . . 113
14.9 ISIS Configuration Examples . . . 114

15 NHRP **117**
15.1 Routing Design . . . 117
15.2 Configuring NHRP . . . 118
15.3 Hub Functionality . . . 118
15.4 Integration with IKE . . . 118
15.5 NHRP Events . . . 118
15.6 Configuration Example . . . 119

16 OSPFv2 **121**
16.1 OSPF Fundamentals . . . 121
16.2 Configuring ospfd . . . 128
16.3 OSPF router . . . 128
16.4 OSPF area . . . 131
16.5 OSPF interface . . . 133
16.6 Redistribute routes to OSPF . . . 135
16.7 Showing OSPF information . . . 136
16.8 Opaque LSA . . . 137
16.9 Traffic Engineering . . . 137
16.10 Router Information . . . 137
16.11 Segment Routing . . . 138
16.12 Debugging OSPF . . . 138
16.13 OSPF Configuration Examples . . . 139

17 OSPFv3 **143**
17.1 OSPF6 router . . . 143
17.2 OSPF6 area . . . 144
17.3 OSPF6 interface . . . 144
17.4 Redistribute routes to OSPF6 . . . 144
17.5 Showing OSPF6 information . . . 145
17.6 OSPF6 Configuration Examples . . . 145

18 PIM **147**
18.1 Starting and Stopping pimd . . . 147
18.2 PIM Interface Configuration . . . 149
18.3 PIM Multicast RIB insertion: . . . 149
18.4 Show PIM Information . . . 149
18.5 PIM Debug Commands . . . 151

19 PBR **153**
19.1 Starting PBR . . . 153
19.2 Nexthop Groups . . . 153
19.3 PBR Maps . . . 153
19.4 PBR Policy . . . 154
19.5 PBR Details . . . 154

20 RIP **155**
20.1 Starting and Stopping ripd . . . 155
20.2 RIP Configuration . . . 156
20.3 RIP Version Control . . . 157
20.4 How to Announce RIP route . . . 158
20.5 Filtering RIP Routes . . . 159
20.6 RIP Metric Manipulation . . . 159
20.7 RIP distance . . . 159
20.8 RIP route-map . . . 160
20.9 RIP Authentication . . . 161
20.10 RIP Timers . . . 161
20.11 Show RIP Information . . . 162
20.12 RIP Debug Commands . . . 162

21 RIPng **165**
21.1 Invoking ripngd . . . 165
21.2 ripngd Configuration . . . 165
21.3 ripngd Terminal Mode Commands . . . 166
21.4 ripngd Filtering Commands . . . 166

22 Starting SHARP **167**

23 USING SHARP **169**

24 VNC and VNC-GW **171**
24.1 Configuring VNC . . . 171
24.2 Manual Address Control . . . 179
24.3 Other VNC-Related Commands . . . 179
24.4 Example VNC and VNC-GW Configurations . . . 180

25 Glossary **193**

26 Packet Binary Dump Format **195**

Bibliography **199**

CHAPTER 1

Overview

FRR is a routing software package that provides TCP/IP based routing services with routing protocols support such as RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, IS-IS, BGP-4, and BGP-4+ (*Supported RFCs*). FRR also supports special BGP Route Reflector and Route Server behavior. In addition to traditional IPv4 routing protocols, FRR also supports IPv6 routing protocols. With SNMP daemon which supports SMUX and AgentX protocol, FRR provides routing protocol MIBs (*SNMP Support*).

FRR uses an advanced software architecture to provide you with a high quality, multi server routing engine. FRR has an interactive user interface for each routing protocol and supports common client commands. Due to this design, you can add new protocol daemons to FRR easily. You can use FRR library as your program's client user interface.

FRR is distributed under the GNU General Public License.

1.1 About FRR

Today, TCP/IP networks are covering all of the world. The Internet has been deployed in many countries, companies, and to the home. When you connect to the Internet your packet will pass many routers which have TCP/IP routing functionality.

A system with FRR installed acts as a dedicated router. With FRR, your machine exchanges routing information with other routers using routing protocols. FRR uses this information to update the kernel routing table so that the right data goes to the right place. You can dynamically change the configuration and you may view routing table information from the FRR terminal interface.

Adding to routing protocol support, FRR can setup interface's flags, interface's address, static routes and so on. If you have a small network, or a stub network, or xDSL connection, configuring the FRR routing software is very easy. The only thing you have to do is to set up the interfaces and put a few commands about static routes and/or default routes. If the network is rather large, or if the network structure changes frequently, you will want to take advantage of FRR's dynamic routing protocol support for protocols such as RIP, OSPF, IS-IS or BGP.

Traditionally, UNIX based router configuration is done by *ifconfig* and *route* commands. Status of routing table is displayed by *netstat* utility. Almost of these commands work only if the user has root privileges. FRR has a different system administration method. There are two user modes in FRR. One is normal mode, the other is enable mode.

Normal mode user can only view system status, enable mode user can change system configuration. This UNIX account independent feature will be great help to the router administrator.

Currently, FRR supports common unicast routing protocols, that is BGP, OSPF, RIP and IS-IS. Upcoming for MPLS support, an implementation of LDP is currently being prepared for merging. Implementations of BFD and PIM-SSM (IPv4) also exist, but are not actively being worked on.

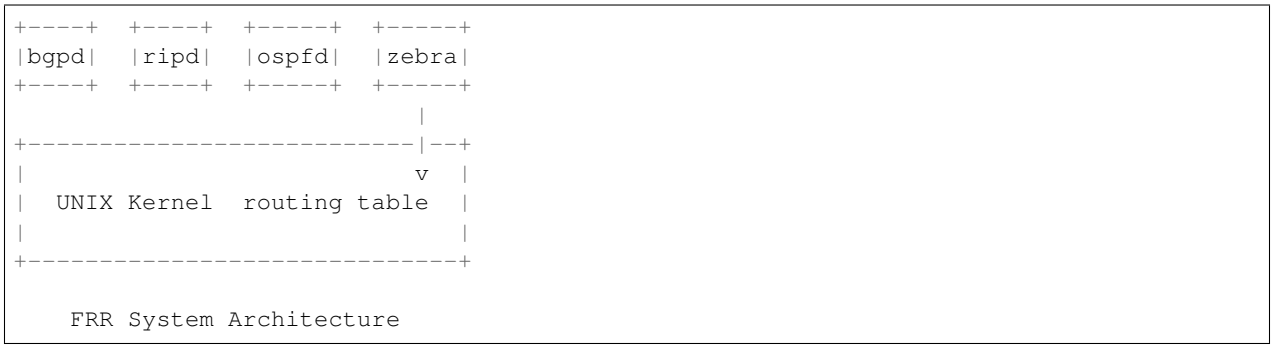
The ultimate goal of the FRR project is making a productive, quality, free TCP/IP routing software package.

1.2 System Architecture

Traditional routing software is made as a one process program which provides all of the routing protocol functionalities. FRR takes a different approach. It is made from a collection of several daemons that work together to build the routing table. There may be several protocol-specific routing daemons and zebra the kernel routing manager.

The *ripd* daemon handles the RIP protocol, while *ospfd* is a daemon which supports OSPF version 2. *bgpd* supports the BGP-4 protocol. For changing the kernel routing table and for redistribution of routes between different routing protocols, there is a kernel routing table manager *zebra* daemon. It is easy to add a new routing protocol daemons to the entire routing system without affecting any other software. You need to run only the protocol daemon associated with routing protocols in use. Thus, user may run a specific daemon and send routing reports to a central routing console.

There is no need for these daemons to be running on the same machine. You can even run several same protocol daemons on the same machine. This architecture creates new possibilities for the routing system.



FRR System Architecture

Multi-process architecture brings extensibility, modularity and maintainability. At the same time it also brings many configuration files and terminal interfaces. Each daemon has it's own configuration file and terminal interface. When you configure a static route, it must be done in *zebra* configuration file. When you configure BGP network it must be done in *bgpd* configuration file. This can be a very annoying thing. To resolve the problem, FRR provides integrated user interface shell called *vtysh*. *vtysh* connects to each daemon with UNIX domain socket and then works as a proxy for user input.

FRR was planned to use multi-threaded mechanism when it runs with a kernel that supports multi-threads. But at the moment, the thread library which comes with GNU/Linux or FreeBSD has some problems with running reliable services such as routing software, so we don't use threads at all. Instead we use the *select(2)* system call for multiplexing the events.

1.3 Supported Platforms

Currently FRR supports GNU/Linux and BSD. Porting FRR to other platforms is not too difficult as platform dependent code should most be limited to the *zebra* daemon. Protocol daemons are mostly platform independent. Please let us know when you find out FRR runs on a platform which is not listed below.

The list of officially supported platforms are listed below. Note that FRR may run correctly on other platforms, and may run with partial functionality on further platforms.

- GNU/Linux
- FreeBSD
- NetBSD
- OpenBSD

Versions of these platforms that are older than around 2 years from the point of their original release (in case of GNU/Linux, this is since the kernel's release on https://kernel.org/) may need some work. Similarly, the following platforms may work with some effort:

- Solaris
- MacOS

Also note that, in particular regarding proprietary platforms, compiler and C library choice will affect FRR. Only recent versions of the following C compilers are well-tested:

- GNU's GCC
- LLVM's clang
- Intel's ICC

1.4 Supported RFCs

FRR implements the following RFCs:

- **RFC 1058** *Routing Information Protocol. C.L. Hedrick. Jun-01-1988.*
- **RFC 2082** *RIP-2 MD5 Authentication. F. Baker, R. Atkinson. January 1997.*
- **RFC 2453** *RIP Version 2. G. Malkin. November 1998.*
- **RFC 2080** *RIPng for IPv6. G. Malkin, R. Minnear. January 1997.*
- **RFC 2328** *OSPF Version 2. J. Moy. April 1998.*
- **RFC 2370** *The OSPF Opaque LSA Option R. Coltun. July 1998.*
- **RFC 3101** *The OSPF Not-So-Stubby Area (NSSA) Option P. Murphy. January 2003.*
- **RFC 2740** *OSPF for IPv6. R. Coltun, D. Ferguson, J. Moy. December 1999.*
- **RFC 1771** *A Border Gateway Protocol 4 (BGP-4). Y. Rekhter & T. Li. March 1995.*
- **RFC 1965** *Autonomous System Confederations for BGP. P. Traina. June 1996.*
- **RFC 1997** *BGP Communities Attribute. R. Chandra, P. Traina & T. Li. August 1996.*
- **RFC 2545** *Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing. P. Marques, F. Dupont. March 1999.*
- **RFC 2796** *BGP Route Reflection An alternative to full mesh IBGP. T. Bates & R. Chandrasekeran. June 1996.*
- **RFC 2858** *Multiprotocol Extensions for BGP-4. T. Bates, Y. Rekhter, R. Chandra, D. Katz. June 2000.*
- **RFC 2842** *Capabilities Advertisement with BGP-4. R. Chandra, J. Scudder. May 2000.*
- **RFC 3137** *OSPF Stub Router Advertisement, A. Retana, L. Nguyen, R. White, A. Zinin, D. McPherson. June 2001*

When SNMP support is enabled, the following RFCs are also supported:

- **RFC 1227** *SNMP MUX protocol and MIB. M.T. Rose. May-01-1991.*
- **RFC 1657** *Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2. S. Willis, J. Burruss, J. Chu, Editor. July 1994.*
- **RFC 1724** *RIP Version 2 MIB Extension. G. Malkin & F. Baker. November 1994.*
- **RFC 1850** *OSPF Version 2 Management Information Base. F. Baker, R. Coltun. November 1995.*
- **RFC 2741** *Agent Extensibility (AgentX) Protocol. M. Daniele, B. Wijnen. January 2000.*

1.5 How to get FRR

The official FRR website is located at https://frrouting.org/ and contains further information, as well as links to additional resources.

FRR is a fork of Quagga.

1.6 Mailing Lists

Italicized lists are private.

Topic	List
Development	dev@lists.frrouting.org
Users & Operators	frog@lists.frrouting.org
Announcements	announce@lists.frrouting.org
Security	security@lists.frrouting.org
Technical Steering Committee	tsc@lists.frrouting.org

The Development list is used to discuss and document general issues related to project development and governance. The public Slack instance and weekly technical meetings provide a higher bandwidth channel for discussions. The results of such discussions are reflected in updates, as appropriate, to code (i.e., merges), GitHub issues tracked issues, and for governance or process changes, updates to the Development list and either this file or information posted at FRR.

1.7 Bug Reports

If you think you have found a bug, please file a bug report on our GitHub issues page.

When you send a bug report, please be careful about the points below.

- Please note what kind of OS you are using. If you use the IPv6 stack please note that as well.
- Please show us the results of *netstat -rn* and *ifconfig -a*. Information from zebra's VTY command *show ip route* will also be helpful.
- Please send your configuration file with the report. If you specify arguments to the configure script please note that too.

Bug reports help us improve FRR and are very much appreciated.

CHAPTER 2

Installation

Several distributions provide packages for FRR. Check your distribution's repositories to find out if a suitable version is available.

FRR depends on various libraries depending on your operating system.

After installing these dependencies, change to the frr source directory and issue the following commands:

```
$ ./bootstrap.sh
$ ./configure
$ make
$ make install
```

2.1 Configure the Software

2.1.1 The Configure Script

FRR has an excellent configure script which automatically detects most host configurations. There are several additional configure options to customize the build to include or exclude specific features and dependencies.

--disable-zebra
Do not build zebra daemon.

--disable-ripd
Do not build ripd.

--disable-ripngd
Do not build ripngd.

--disable-ospfd
Do not build ospfd.

--disable-ospf6d
Do not build ospf6d.

--disable-bgpd
Do not build bgpd.

--disable-bgp-announce
Make *bgpd* which does not make bgp announcements at all. This feature is good for using *bgpd* as a BGP announcement listener.

--enable-datacenter
Enable system defaults to work as if in a Data Center. See defaults.h for what is changed by this configure option.

--enable-snmp
Enable SNMP support. By default, SNMP support is disabled.

--disable-ospfapi
Disable support for OSPF-API, an API to interface directly with ospfd. OSPF-API is enabled if –enable-opaque-lsa is set.

--disable-ospfclient
Disable building of the example OSPF-API client.

--disable-ospf-ri
Disable support for OSPF Router Information (RFC4970 & RFC5088) this requires support for Opaque LSAs and Traffic Engineering.

--disable-isisd
Do not build isisd.

--enable-isis-topology
Enable IS-IS topology generator.

--enable-isis-te
Enable Traffic Engineering Extension for ISIS (RFC5305)

--enable-realms
Enable the support of Linux Realms. Convert tag values from 1-255 into a realm value when inserting into the Linux kernel. Then routing policy can be assigned to the realm. See the tc man page.

--disable-rtadv
Disable support IPV6 router advertisement in zebra.

--enable-gcc-rdynamic
Pass the `-rdynamic` option to the linker driver. This is in most cases necessary for getting usable backtraces. This option defaults to on if the compiler is detected as gcc, but giving an explicit enable/disable is suggested.

--disable-backtrace
Controls backtrace support for the crash handlers. This is autodetected by default. Using the switch will enforce the requested behaviour, failing with an error if support is requested but not available. On BSD systems, this needs libexecinfo, while on glibc support for this is part of libc itself.

--enable-dev-build
Turn on some options for compiling FRR within a development environment in mind. Specifically turn on -g3 -O0 for compiling options and add inclusion of grammar sandbox.

--enable-fuzzing
Turn on some compile options to allow you to run fuzzing tools against the system. This flag is intended as a developer only tool and should not be used for normal operations.

--disable-snmp
Build without SNMP support.

--disable-vtysh
Build without VTYSH.

--enable-fpm
Build with FPM module support.

--enable-numeric-version
Alpine Linux does not allow non-numeric characters in the version string. With this option, we provide a way to strip out these characters for APK dev package builds.

--enable-multipath=X
Compile FRR with up to X way ECMP supported. This number can be from 0-999. For backwards compatability with older configure options when setting X = 0, we will build FRR with 64 way ECMP. This is needed because there are hardcoded arrays that FRR builds towards, so we need to know how big to make these arrays at build time.

You may specify any combination of the above options to the configure script. By default, the executables are placed in `/usr/local/sbin` and the configuration files in `/usr/local/etc`. The `/usr/local/` installation prefix and other directories may be changed using the following options to the configuration script.

--prefix <prefix>
Install architecture-independent files in *prefix* [/usr/local].

--sysconfdir <dir>
Look for configuration files in *dir* [*prefix*/etc]. Note that sample configuration files will be installed here.

--localstatedir <dir>
Configure zebra to use *dir* for local state files, such as pid files and unix sockets.

2.1.2 Least-Privilege Support

Additionally, you may configure zebra to drop its elevated privileges shortly after startup and switch to another user. The configure script will automatically try to configure this support. There are three configure options to control the behaviour of FRR daemons.

--enable-user <user>
Switch to user *user shortly after startup, and run as user 'user* in normal operation.

--enable-group <user>
Switch real and effective group to *group* shortly after startup.

--enable-vty-group <group>
Create Unix Vty sockets (for use with vtysh) with group ownership set to *group*. This allows one to create a separate group which is restricted to accessing only the vty sockets, hence allowing one to delegate this group to individual users, or to run vtysh setgid to this group.

The default user and group which will be configured is 'frr' if no user or group is specified. Note that this user or group requires write access to the local state directory (see *--localstatedir*) and requires at least read access, and write access if you wish to allow daemons to write out their configuration, to the configuration directory (see *--sysconfdir*).

On systems which have the 'libcap' capabilities manipulation library (currently only Linux), FRR will retain only minimal capabilities required and will only raise these capabilities for brief periods. On systems without libcap, FRR will run as the user specified and only raise its UID to 0 for brief periods.

2.1.3 Linux Notes

There are several options available only to GNU/Linux systems[1]. If you use GNU/Linux, make sure that the current kernel configuration is what you want. FRR will run with any kernel configuration but some recommendations do exist.

- **CONFIG_NETLINK** Kernel/User Netlink socket. This is a brand new feature which enables an advanced interface between the Linux kernel and zebra (*Kernel Interface*).
- **CONFIG_RTNETLINK** Routing messages. This makes it possible to receive Netlink routing messages. If you specify this option, *zebra* can detect routing information updates directly from the kernel (*Kernel Interface*).
- **CONFIG_IP_MULTICAST** IP: multicasting. This option should be specified when you use *ripd* (*RIP*) or *ospfd* (*OSPFv2*) because these protocols use multicast.

IPv6 support has been added in GNU/Linux kernel version 2.2. If you try to use the FRR IPv6 feature on a GNU/Linux kernel, please make sure the following libraries have been installed. Please note that these libraries will not be needed when you uses GNU C library 2.1 or upper.

- inet6-apps

  The *inet6-apps* package includes basic IPv6 related libraries such as *inet_ntop* and *inet_pton*. Some basic IPv6 programs such as *ping*, *ftp*, and *inetd* are also included. The *inet-apps* can be found at ftp://ftp.inner.net/pub/ipv6/.

- net-tools

  The *net-tools* package provides an IPv6 enabled interface and routing utility. It contains *ifconfig*, *route*, *netstat*, and other tools. *net-tools* may be found at http://www.tazenda.demon.co.uk/phil/net-tools/.

Linux sysctl settings and kernel modules

There are several kernel parameters that impact overall operation of FRR when using Linux as a router. Generally these parameters should be set in a sysctl related configuration file, e.g., `/etc/sysctl.conf` on Ubuntu based systems and a new file `/etc/sysctl.d/90-routing-sysctl.conf` on Centos based systems. Additional kernel modules are also needed to support MPLS forwarding.

IPv4 and IPv6 forwarding The following are set to enable IP forwarding in the kernel:

```
net.ipv4.conf.all.forwarding=1
net.ipv6.conf.all.forwarding=1
```

MPLS forwarding Basic MPLS kernel support was introduced 4.1, additional capability was introduced in 4.3 and 4.5. For some general information on Linux MPLS support see https://www.netdevconf.org/1.1/proceedings/slides/prabhu-mpls-tutorial.pdf. The following modules should be loaded to support MPLS forwarding, and are generally added to a configuration file such as `/etc/modules-load.d/modules.conf`:

```
# Load MPLS Kernel Modules
mpls_router
mpls_iptunnel
```

The following is an example to enable MPLS forwarding in the kernel:

```
# Enable MPLS Label processing on all interfaces
net.mpls.conf.eth0.input=1
net.mpls.conf.eth1.input=1
```

(continues on next page)

[1] GNU/Linux has very flexible kernel configuration features.

(continued from previous page)

```
net.mpls.conf.eth2.input=1
net.mpls.platform_labels=100000
```

Make sure to add a line equal to `net.mpls.conf.<if>.input` for each interface *'<if>'* used with MPLS and to set labels to an appropriate value.

VRF forwarding General information on Linux VRF support can be found in https://www.kernel.org/doc/Documentation/networking/vrf.txt. Kernel support for VRFs was introduced in 4.3 and improved upon through 4.13, which is the version most used in FRR testing (as of June 2018). Additional background on using Linux VRFs and kernel specific features can be found in http://schd.ws/hosted_files/ossna2017/fe/vrf-tutorial-oss.pdf.

The following impacts how BGP TCP sockets are managed across VRFs:

```
net.ipv4.tcp_l3mdev_accept=0
```

With this setting a BGP TCP socket is opened per VRF. This setting ensures that other TCP services, such as SSH, provided for non-VRF purposes are blocked from VRF associated Linux interfaces.

```
net.ipv4.tcp_l3mdev_accept=1
```

With this setting a single BGP TCP socket is shared across the system. This setting exposes any TCP service running on the system, e.g., SSH, to all VRFs. Generally this setting is not used in environments where VRFs are used to support multiple administrative groups.

Important note as of June 2018, Kernel versions 4.14-4.18 have a known bug where VRF-specific TCP sockets are not properly handled. When running these kernel versions, if unable to establish any VRF BGP adjacencies, either downgrade to 4.13 or set 'net.ipv4.tcp_l3mdev_accept=1'. The fix for this issue is planned to be included in future kernel versions so upgrading your kernel may also address this issue.

2.2 Build the Software

After configuring the software, you will need to compile it for your system. Simply issue the command *make* in the root of the source directory and the software will be compiled. Cliff Notes versions of different compilation examples can be found in the Developer's Manual Appendix. If you have *any* problems at this stage, please send a bug report *Bug Reports*.

```
$ ./bootstrap.sh
$ ./configure <appropriate to your system>
$ make
```

2.3 Install the Software

Installing the software to your system consists of copying the compiled programs and supporting files to a standard location. After the installation process has completed, these files have been copied from your work directory to `/usr/local/bin`, and `/usr/local/etc`.

To install the FRR suite, issue the following command at your shell prompt::

```
$ make install
```

FRR daemons have their own terminal interface or VTY. After installation, you have to setup each beast's port number to connect to them. Please add the following entries to `/etc/services`.

```
zebrasrv        2600/tcp                # zebra service
zebra           2601/tcp                # zebra vty
ripd            2602/tcp                # RIPd vty
ripngd          2603/tcp                # RIPngd vty
ospfd           2604/tcp                # OSPFd vty
bgpd            2605/tcp                # BGPd vty
ospf6d          2606/tcp                # OSPF6d vty
ospfapi         2607/tcp                # ospfapi
isisd           2608/tcp                # ISISd vty
nhrpd           2610/tcp                # nhrpd vty
pimd            2611/tcp                # PIMd vty
```

If you use a FreeBSD newer than 2.2.8, the above entries are already added to `/etc/services` so there is no need to add it. If you specify a port number when starting the daemon, these entries may not be needed.

You may need to make changes to the config files in /etc/frr. *Config Commands*.

CHAPTER 3

Basic commands

There are five routing daemons in use, and there is one manager daemon. These daemons may be located on separate machines from the manager daemon. Each of these daemons will listen on a particular port for incoming VTY connections. The routing daemons are:

- *ripd*
- *ripngd*
- *ospfd*
- *ospf6d*
- *bgpd*
- *zebra*

The following sections discuss commands common to all the routing daemons.

3.1 Config Commands

In a config file, you can write the debugging options, a vty's password, routing daemon configurations, a log file name, and so forth. This information forms the initial command set for a routing beast as it is starting.

Config files are generally found in /etc/frr.

Each of the daemons has its own config file. The daemon name plus `.conf` is the default config file name. For example, zebra's default config file name is `zebra.conf`. You can specify a config file using the *-f* or *--config_file* options when starting the daemon.

3.1.1 Basic Config Commands

hostname HOSTNAME
Set hostname of the router.

[no] password PASSWORD
Set password for vty interface. The `no` form of the command deletes the password. If there is no password, a vty won't accept connections.

[no] enable password PASSWORD
Set enable password. The `no` form of the command deletes the enable password.

[no] log trap LEVEL
These commands are deprecated and are present only for historical compatibility. The log trap command sets the current logging level for all enabled logging destinations, and it sets the default for all future logging commands that do not specify a level. The normal default logging level is debugging. The `no` form of the command resets the default level for future logging commands to debugging, but it does not change the logging level of existing logging destinations.

[no] log stdout LEVEL
Enable logging output to stdout. If the optional second argument specifying the logging level is not present, the default logging level (typically debugging, but can be changed using the deprecated `log trap` command) will be used. The `no` form of the command disables logging to stdout. The `LEVEL` argument must have one of these values: emergencies, alerts, critical, errors, warnings, notifications, informational, or debugging. Note that the existing code logs its most important messages with severity `errors`.

[no] log file [FILENAME [LEVEL]]
If you want to log into a file, please specify `filename` as in this example:

```
log file /var/log/frr/bgpd.log informational
```

If the optional second argument specifying the logging level is not present, the default logging level (typically debugging, but can be changed using the deprecated `log trap` command) will be used. The `no` form of the command disables logging to a file. *Note:* if you do not configure any file logging, and a daemon crashes due to a signal or an assertion failure, it will attempt to save the crash information in a file named /var/tmp/frr.<daemon name>.crashlog. For security reasons, this will not happen if the file exists already, so it is important to delete the file after reporting the crash information.

[no] log syslog [LEVEL]
Enable logging output to syslog. If the optional second argument specifying the logging level is not present, the default logging level (typically debugging, but can be changed using the deprecated `log trap` command) will be used. The `no` form of the command disables logging to syslog.

[no] log monitor [LEVEL]
Enable logging output to vty terminals that have enabled logging using the `terminal monitor` command. By default, monitor logging is enabled at the debugging level, but this command (or the deprecated `log trap` command) can be used to change the monitor logging level. If the optional second argument specifying the logging level is not present, the default logging level (typically debugging, but can be changed using the deprecated `log trap` command) will be used. The `no` form of the command disables logging to terminal monitors.

[no] log facility [FACILITY]
This command changes the facility used in syslog messages. The default facility is `daemon`. The `no` form of the command resets the facility to the default `daemon` facility.

[no] log record-priority
To include the severity in all messages logged to a file, to stdout, or to a terminal monitor (i.e. anything except syslog), use the `log record-priority` global configuration command. To disable this option, use the `no` form of the command. By default, the severity level is not included in logged messages. Note: some versions of syslogd (including Solaris) can be configured to include the facility and level in the messages emitted.

[no] log timestamp precision [(0-6)]
This command sets the precision of log message timestamps to the given number of digits after the decimal point. Currently, the value must be in the range 0 to 6 (i.e. the maximum precision is microseconds). To restore the default behavior (1-second accuracy), use the `no` form of the command, or set the precision explicitly to 0.

```
  log timestamp precision 3

In this example, the precision is set to provide timestamps with
millisecond accuracy.
```

log commands
This command enables the logging of all commands typed by a user to all enabled log destinations. The note that logging includes full command lines, including passwords. Once set, command logging can only be turned off by restarting the daemon.

service password-encryption
Encrypt password.

service advanced-vty
Enable advanced mode VTY.

service terminal-length (0-512)
Set system wide line configuration. This configuration command applies to all VTY interfaces.

line vty
Enter vty configuration mode.

banner motd default
Set default motd string.

no banner motd
No motd banner string will be printed.

exec-timeout MINUTE [SECOND]
Set VTY connection timeout value. When only one argument is specified it is used for timeout value in minutes. Optional second argument is used for timeout value in seconds. Default timeout value is 10 minutes. When timeout value is zero, it means no timeout.

no exec-timeout
Do not perform timeout at all. This command is as same as *exec-timeout 0 0*.

access-class ACCESS-LIST
Restrict vty connections with an access list.

3.1.2 Sample Config File

Below is a sample configuration file for the zebra daemon.

```
!
! Zebra configuration file
!
hostname Router
password zebra
enable password zebra
!
log stdout
!
!
```

'!' and '#' are comment characters. If the first character of the word is one of the comment characters then from the rest of the line forward will be ignored as a comment.

```
password zebra!password
```

If a comment character is not the first character of the word, it's a normal character. So in the above example '!' will not be regarded as a comment and the password is set to 'zebra!password'.

3.2 Terminal Mode Commands

write terminal
Displays the current configuration to the vty interface.

write file
Write current configuration to configuration file.

configure terminal
Change to configuration mode. This command is the first step to configuration.

terminal length (0-512)
Set terminal display length to `(0-512)`. If length is 0, no display control is performed.

who
Show a list of currently connected vty sessions.

list
List all available commands.

show version
Show the current version of frr and its build host information.

show logging
Shows the current configuration of the logging system. This includes the status of all logging destinations.

logmsg LEVEL MESSAGE
Send a message to all logging destinations that are enabled for messages of the given severity.

3.3 Common Invocation Options

These options apply to all frr daemons.

-d, --daemon
Run in daemon mode.

-f, --config_file `<file>`
Set configuration file name.

-h, --help
Display this help and exit.

-i, --pid_file `<file>`
Upon startup the process identifier of the daemon is written to a file, typically in `/var/run`. This file can be used by the init system to implement commands such as `.../init.d/zebra status`, `.../init.d/zebra restart` or `.../init.d/zebra stop`.

The file name is an run-time option rather than a configure-time option so that multiple routing daemons can be run simultaneously. This is useful when using frr to implement a routing looking glass. One machine can be used to collect differing routing views from differing points in the network.

-A, --vty_addr `<address>`
Set the VTY local address to bind to. If set, the VTY socket will only be bound to this address.

-P, --vty_port <port>
Set the VTY TCP port number. If set to 0 then the TCP VTY sockets will not be opened.

-u <user>
Set the user and group to run as.

-v, --version
Print program version.

3.4 Loadable Module Support

FRR supports loading extension modules at startup. Loading, reloading or unloading modules at runtime is not supported (yet). To load a module, use the following command line option at daemon startup:

-M, --module <module:options>
Load the specified module, optionally passing options to it. If the module name contains a slash (/), it is assumed to be a full pathname to a file to be loaded. If it does not contain a slash, the /usr/lib/frr/modules directory is searched for a module of the given name; first with the daemon name prepended (e.g. `zebra_mod` for `mod`), then without the daemon name prepended.

This option is available on all daemons, though some daemons may not have any modules available to be loaded.

3.4.1 The SNMP Module

If SNMP is enabled during compile-time and installed as part of the package, the `snmp` module can be loaded for the *zebra*, *bgpd*, *ospfd*, *ospf6d* and *ripd* daemons.

The module ignores any options passed to it. Refer to *SNMP Support* for information on its usage.

3.4.2 The FPM Module

If FPM is enabled during compile-time and installed as part of the package, the `fpm` module can be loaded for the *zebra* daemon. This provides the Forwarding Plane Manager ("FPM") API.

The module expects its argument to be either `Netlink` or `protobuf`, specifying the encapsulation to use. `Netlink` is the default, and `protobuf` may not be available if the module was built without protobuf support. Refer to *zebra FIB push interface* for more information.

3.5 Virtual Terminal Interfaces

VTY – Virtual Terminal [aka TeletYpe] Interface is a command line interface (CLI) for user interaction with the routing daemon.

3.5.1 VTY Overview

VTY stands for Virtual TeletYpe interface. It means you can connect to the daemon via the telnet protocol.

To enable a VTY interface, you have to setup a VTY password. If there is no VTY password, one cannot connect to the VTY interface at all.

```
% telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is |PACKAGE_NAME| (version |PACKAGE_VERSION|)
|COPYRIGHT_STR|

User Access Verification

Password: XXXXX
Router> ?
  enable .  .  .  Turn on privileged commands
  exit   .  .  .  Exit current mode and down to previous mode
  help   .  .  .  Description of the interactive help system
  list   .  .  .  Print command list
  show   .  .  .  Show system inform

  wh. . .  Display who is on a vty
Router> enable
Password: XXXXX
Router# configure terminal
Router(config)# interface eth0
Router(config-if)# ip address 10.0.0.1/8
Router(config-if)# ^Z
Router#
```

`?` and the `find` command are very useful for looking up commands.

3.5.2 VTY Modes

There are three basic VTY modes:

There are commands that may be restricted to specific VTY modes.

VTY View Mode

This mode is for read-only access to the CLI. One may exit the mode by leaving the system, or by entering *enable* mode.

VTY Enable Mode

This mode is for read-write access to the CLI. One may exit the mode by leaving the system, or by escaping to view mode.

VTY Other Modes

This page is for describing other modes.

3.5.3 VTY CLI Commands

Commands that you may use at the command-line are described in the following three subsubsections.

CLI Movement Commands

These commands are used for moving the CLI cursor. The `C` character means press the Control Key.

C-f / LEFT Move forward one character.

C-b / RIGHT Move backward one character.

M-f Move forward one word.

M-b Move backward one word.

C-a Move to the beginning of the line.

C-e Move to the end of the line.

CLI Editing Commands

These commands are used for editing text on a line. The `C` character means press the Control Key.

C-h / DEL Delete the character before point.

C-d Delete the character after point.

M-d Forward kill word.

C-w Backward kill word.

C-k Kill to the end of the line.

C-u Kill line from the beginning, erasing input.

C-t Transpose character.

CLI Advanced Commands

There are several additional CLI commands for command line completions, insta-help, and VTY session management.

C-c Interrupt current input and moves to the next line.

C-z End current configuration session and move to top node.

C-n / DOWN Move down to next line in the history buffer.

C-p / UP Move up to previous line in the history buffer.

TAB Use command line completion by typing `TAB`.

? You can use command line help by typing `help` at the beginning of the line. Typing `?` at any point in the line will show possible completions.

CHAPTER 4

VTY shell

vtysh provides a combined frontend to all FRR daemons in a single combined session. It is enabled by default at build time, but can be disabled through the `--disable-vtysh` option to the configure script.

vtysh has a configuration file, `vtysh.conf`. The location of that file cannot be changed from /etc/frr since it contains options controlling authentication behavior. This file will also not be written by configuration-save commands, it is intended to be updated manually by an administrator with an external editor.

Warning: This also means the `hostname` and `banner motd` commands (which both do have effect for vtysh) need to be manually updated in `vtysh.conf`.

4.1 Permissions and setup requirements

vtysh connects to running daemons through Unix sockets located in /var/run/frr. Running vtysh thus requires access to that directory, plus membership in the frrvty group (which is the group that the daemons will change ownership of their sockets to).

To restrict access to FRR configuration, make sure no unauthorized users are members of the frrvty group.

Warning: VTYSH implements a CLI option `-u, --user` that disallows entering the characters "en" on the command line, which ideally restricts access to configuration commands. However, VTYSH was never designed to be a privilege broker and is not built using secure coding practices. No guarantees of security are provided for this option and under no circumstances should this option be used to provide any semblance of security or read-only access to FRR.

4.1.1 PAM support (experimental)

vtysh has working (but rather useless) PAM support. It will perform an "authenticate" PAM call using frr as service name. No other (accounting, session, password change) calls will be performed by vtysh.

Users using vtysh still need to have appropriate access to the daemons' VTY sockets, usually by being member of the frrvty group. If they have this membership, PAM support is useless since they can connect to daemons and issue commands using some other tool. Alternatively, the *vtysh* binary could be made SGID (set group ID) to the frrvty group.

> **Warning:** No security guarantees are made for this configuration.

username USERNAME nopassword
: If PAM support is enabled at build-time, this command allows disabling the use of PAM on a per-user basis. If vtysh finds that an user is trying to use vtysh and a "nopassword" entry is found, no calls to PAM will be made at all.

4.2 Integrated configuration mode

Integrated configuration mode uses a single configuration file, `frr.conf`, for all daemons. This replaces the individual files like `zebra.conf` or `bgpd.conf`.

`frr.conf` is located in /etc/frr. All daemons check for the existence of this file at startup, and if it exists will not load their individual configuration files. Instead, `vtysh -b` must be invoked to process `frr.conf` and apply its settings to the individual daemons.

> **Warning:** *vtysh -b* must also be executed after restarting any daemon.

4.2.1 Configuration saving, file ownership and permissions

The `frr.conf` file is not written by any of the daemons; instead *vtysh* contains the necessary logic to collect configuration from all of the daemons, combine it and write it out.

> **Warning:** Daemons must be running for *vtysh* to be able to collect their configuration. Any configuration from non-running daemons is permanently lost after doing a configuration save.

Since the *vtysh* command may be running as ordinary user on the system, configuration writes will be tried through *watchfrr*, using the `write integrated` command internally. Since *watchfrr* is running as superuser, *vtysh* is able to ensure correct ownership and permissions on `frr.conf`.

If *watchfrr* is not running or the configuration write fails, *vtysh* will attempt to directly write to the file. This is likely to fail if running as unprivileged user; alternatively it may leave the file with incorrect owner or permissions.

Writing the configuration can be triggered directly by invoking *vtysh -w*. This may be useful for scripting. Note this command should be run as either the superuser or the FRR user.

We recommend you do not mix the use of the two types of files. Further, it is better not to use the integrated `frr.conf` file, as any syntax error in it can lead to /all/ of your daemons being unable to start up. Per daemon files are more robust as impact of errors in configuration are limited to the daemon in whose file the error is made.

service integrated-vtysh-config

no service integrated-vtysh-config
: Control whether integrated `frr.conf` file is written when 'write file' is issued.

These commands need to be placed in `vtysh.conf` to have any effect. Note that since `vtysh.conf` is not written by FRR itself, they therefore need to be manually placed in that file.

This command has 3 states:

service integrated-vtysh-config *vtysh* will always write `frr.conf`.

no service integrated-vtysh-config *vtysh* will never write `frr.conf`; instead it will ask daemons to write their individual configuration files.

Neither option present (default) *vtysh* will check whether `frr.conf` exists. If it does, configuration writes will update that file. Otherwise, writes are performed through the individual daemons.

This command is primarily intended for packaging/distribution purposes, to preset one of the two operating modes and ensure consistent operation across installations.

`write integrated`

Unconditionally (regardless of `service integrated-vtysh-config` setting) write out integrated `frr.conf` file through *watchfrr*. If *watchfrr* is not running, this command is unavailable.

Warning: Configuration changes made while some daemon is not running will be invisible to that daemon. The daemon will start up with its saved configuration (either in its individual configuration file, or in `frr.conf`). This is particularly troublesome for route-maps and prefix lists, which would otherwise be synchronized between daemons.

CHAPTER 5

Filtering

FRR provides many very flexible filtering features. Filtering is used for both input and output of the routing information. Once filtering is defined, it can be applied in any direction.

5.1 IP Access List

access-list NAME permit IPV4-NETWORK

access-list NAME deny IPV4-NETWORK
Basic filtering is done by *access-list* as shown in the following example.

```
access-list filter deny 10.0.0.0/9
access-list filter permit 10.0.0.0/8
```

5.2 IP Prefix List

ip prefix-list provides the most powerful prefix based filtering mechanism. In addition to *access-list* functionality, *ip prefix-list* has prefix length range specification and sequential number specification. You can add or delete prefix based filters to arbitrary points of prefix-list using sequential number specification.

If no ip prefix-list is specified, it acts as permit. If *ip prefix-list* is defined, and no match is found, default deny is applied.

ip prefix-list NAME (permit|deny) PREFIX [le LEN] [ge LEN]

ip prefix-list NAME seq NUMBER (permit|deny) PREFIX [le LEN] [ge LEN]
You can create *ip prefix-list* using above commands.

seq seq *number* can be set either automatically or manually. In the case that sequential numbers are set manually, the user may pick any number less than 4294967295. In the case that sequential number are set automatically, the sequential number will increase by a unit of five (5) per list. If a list with no specified sequential number is created after a list with a specified sequential number, the list will automatically pick

the next multiple of five (5) as the list number. For example, if a list with number 2 already exists and a new list with no specified number is created, the next list will be numbered 5. If lists 2 and 7 already exist and a new list with no specified number is created, the new list will be numbered 10.

le Specifies prefix length. The prefix list will be applied if the prefix length is less than or equal to the le prefix length.

ge Specifies prefix length. The prefix list will be applied if the prefix length is greater than or equal to the ge prefix length.

Less than or equal to prefix numbers and greater than or equal to prefix numbers can be used together. The order of the le and ge commands does not matter.

If a prefix list with a different sequential number but with the exact same rules as a previous list is created, an error will result. However, in the case that the sequential number and the rules are exactly similar, no error will result.

If a list with the same sequential number as a previous list is created, the new list will overwrite the old list.

Matching of IP Prefix is performed from the smaller sequential number to the larger. The matching will stop once any rule has been applied.

In the case of no le or ge command, the prefix length must match exactly the length specified in the prefix list.

no ip prefix-list NAME

5.2.1 ip prefix-list description

ip prefix-list NAME description DESC
Descriptions may be added to prefix lists. This command adds a description to the prefix list.

no ip prefix-list NAME description [DESC]
Deletes the description from a prefix list. It is possible to use the command without the full description.

5.2.2 ip prefix-list sequential number control

ip prefix-list sequence-number
With this command, the IP prefix list sequential number is displayed. This is the default behavior.

no ip prefix-list sequence-number
With this command, the IP prefix list sequential number is not displayed.

5.2.3 Showing ip prefix-list

show ip prefix-list
Display all IP prefix lists.

show ip prefix-list NAME
Show IP prefix list can be used with a prefix list name.

show ip prefix-list NAME seq NUM
Show IP prefix list can be used with a prefix list name and sequential number.

show ip prefix-list NAME A.B.C.D/M
If the command longer is used, all prefix lists with prefix lengths equal to or longer than the specified length will be displayed. If the command first match is used, the first prefix length match will be displayed.

show ip prefix-list NAME A.B.C.D/M longer

show ip prefix-list NAME A.B.C.D/M first-match

show ip prefix-list summary

show ip prefix-list summary NAME

show ip prefix-list detail

show ip prefix-list detail NAME

5.2.4 Clear counter of ip prefix-list

clear ip prefix-list
Clears the counters of all IP prefix lists. Clear IP Prefix List can be used with a specified name and prefix.

clear ip prefix-list NAME

clear ip prefix-list NAME A.B.C.D/M

CHAPTER 6

Route Maps

Route maps provide a means to both filter and/or apply actions to route, hence allowing policy to be applied to routes.

Route maps are an ordered list of route map entries. Each entry may specify up to four distinct sets of clauses:

Matching Conditions A route-map entry may, optionally, specify one or more conditions which must be matched if the entry is to be considered further, as governed by the Match Policy. If a route-map entry does not explicitly specify any matching conditions, then it always matches.

Set Actions A route-map entry may, optionally, specify one or more Set Actions to set or modify attributes of the route.

Matching Policy This specifies the policy implied if the *Matching Conditions* are met or not met, and which actions of the route-map are to be taken, if any. The two possibilities are:

- *permit*: If the entry matches, then carry out the *Set Actions*. Then finish processing the route-map, permitting the route, unless an *Exit Policy* action indicates otherwise.
- *deny*: If the entry matches, then finish processing the route-map and deny the route (return *deny*).

The *Matching Policy* is specified as part of the command which defines the ordered entry in the route-map. See below.

Call Action Call to another route-map, after any *Set Actions* have been carried out. If the route-map called returns *deny* then processing of the route-map finishes and the route is denied, regardless of the :term:Matching Policy‘ or the *Exit Policy*. If the called route-map returns *permit*, then *Matching Policy* and *Exit Policy* govern further behaviour, as normal.

Exit Policy An entry may, optionally, specify an alternative *Exit Policy* to take if the entry matched, rather than the normal policy of exiting the route-map and permitting the route. The two possibilities are:

- *next*: Continue on with processing of the route-map entries.
- *goto N*: Jump ahead to the first route-map entry whose order in the route-map is >= N. Jumping to a previous entry is not permitted.

The default action of a route-map, if no entries match, is to deny. I.e. a route-map essentially has as its last entry an empty *deny* entry, which matches all routes. To change this behaviour, one must specify an empty *permit* entry as the last entry in the route-map.

To summarise the above:

	Match	No Match
Permit	action	cont
Deny	deny	cont

action

- Apply *set* statements
- If *call* is present, call given route-map. If that returns a `deny`, finish processing and return `deny`.
- If *Exit Policy* is *next*, goto next route-map entry
- If *Exit Policy* is *goto*, goto first entry whose order in the list is >= the given order.
- Finish processing the route-map and permit the route.

deny The route is denied by the route-map (return `deny`).

cont goto next route-map entry

6.1 Route Map Command

route-map ROUTE-MAP-NAME (permit|deny) ORDER
Configure the *order*'th entry in *route-map-name* with `Match Policy` of either *permit* or *deny*.

6.2 Route Map Match Command

match ip address ACCESS_LIST
Matches the specified *access_list*

match ip address PREFIX-LIST
Matches the specified *prefix-list*

match ip address prefix-len 0-32
Matches the specified *prefix-len*. This is a Zebra specific command.

match ipv6 address ACCESS_LIST
Matches the specified *access_list*

match ipv6 address PREFIX-LIST
Matches the specified *prefix-list*

match ipv6 address prefix-len 0-128
Matches the specified *prefix-len*. This is a Zebra specific command.

match ip next-hop IPV4_ADDR
Matches the specified *ipv4_addr*.

match aspath AS_PATH
Matches the specified *as_path*.

match metric METRIC
Matches the specified *metric*.

match tag TAG
Matches the specified tag value associated with the route. This tag value can be in the range of (1-4294967295).

match local-preference METRIC
Matches the specified *local-preference*.

match community COMMUNITY_LIST
Matches the specified *community_list*

match peer IPV4_ADDR
This is a BGP specific match command. Matches the peer ip address if the neighbor was specified in this manner.

match peer IPV6_ADDR
This is a BGP specific match command. Matches the peer ipv6 address if the neighbor was specified in this manner.

match peer INTERFACE_NAME
This is a BGP specific match command. Matches the peer interface name specified if the neighbor was specified in this manner.

match source-protocol PROTOCOL_NAME
This is a ZEBRA specific match command. Matches the originating protocol specified.

match source-instance NUMBER
This is a ZEBRA specific match command. The number is a range from (0-255). Matches the originating protocols instance specified.

6.3 Route Map Set Command

set tag TAG
Set a tag on the matched route. This tag value can be from (1-4294967295). Additionally if you have compiled with the *--enable-realms* configure option. Tag values from (1-255) are sent to the Linux kernel as a realm value. Then route policy can be applied. See the tc man page.

set ip next-hop IPV4_ADDRESS
Set the BGP nexthop address to the specified IPV4_ADDRESS. For both incoming and outgoing route-maps.

set ip next-hop peer-address
Set the BGP nexthop address to the address of the peer. For an incoming route-map this means the ip address of our peer is used. For an outgoing route-map this means the ip address of our self is used to establish the peering with our neighbor.

set ip next-hop unchanged
Set the route-map as unchanged. Pass the route-map through without changing it's value.

set ipv6 next-hop peer-address
Set the BGP nexthop address to the address of the peer. For an incoming route-map this means the ipv6 address of our peer is used. For an outgoing route-map this means the ip address of our self is used to establish the peering with our neighbor.

set ipv6 next-hop prefer-global
For Incoming and Import Route-maps if we receive a v6 global and v6 LL address for the route, then prefer to use the global address as the nexthop.

set ipv6 next-hop global IPV6_ADDRESS
Set the next-hop to the specified IPV6_ADDRESS for both incoming and outgoing route-maps.

set local-preference LOCAL_PREF
Set the BGP local preference to *local_pref*.

set weight WEIGHT
Set the route's weight.

set metric METRIC
Set the BGP attribute MED.

set as-path prepend AS_PATH
Set the BGP AS path to prepend.

set community COMMUNITY
Set the BGP community attribute.

set ipv6 next-hop local IPV6_ADDRESS
Set the BGP-4+ link local IPv6 nexthop address.

6.4 Route Map Call Command

call NAME
Call route-map *name*. If it returns deny, deny the route and finish processing the route-map.

6.5 Route Map Exit Action Command

on-match next

continue
Proceed on to the next entry in the route-map.

on-match goto N

continue N
Proceed processing the route-map at the first entry whose order is >= N

6.6 Route Map Examples

A simple example of a route-map:

```
route-map test permit 10
 match ip address 10
 set local-preference 200
```

This means that if a route matches ip access-list number 10 it's local-preference value is set to 200.

See *BGP Configuration Examples* for examples of more sophisticated usage of route-maps, including of the `call` action.

CHAPTER 7

IPv6 Support

FRR fully supports IPv6 routing. As described so far, FRR supports RIPng, OSPFv3, and BGP-4+. You can give IPv6 addresses to an interface and configure static IPv6 routing information. FRR IPv6 also provides automatic address configuration via a feature called `address auto configuration`. To do it, the router must send router advertisement messages to the all nodes that exist on the network.

Previous versions of FRR could be built without IPv6 support. This is no longer possible.

7.1 Router Advertisement

no ipv6 nd suppress-ra
Send router advertisement messages.

ipv6 nd suppress-ra
Don't send router advertisement messages.

ipv6 nd prefix ipv6prefix [valid-lifetime] [preferred-lifetime] [off-link] [no-autoconfig]
Configuring the IPv6 prefix to include in router advertisements. Several prefix specific optional parameters and flags may follow:

- `valid-lifetime`: the length of time in seconds during what the prefix is valid for the purpose of on-link determination. Value `infinite` represents infinity (i.e. a value of all one bits (`0xffffffff`)). Range: `(0-4294967295)` Default: `2592000`
- `preferred-lifetime`: the length of time in seconds during what addresses generated from the prefix remain preferred. Value `infinite` represents infinity. Range: `(0-4294967295)` Default: `604800`
- `off-link`: indicates that advertisement makes no statement about on-link or off-link properties of the prefix. Default: not set, i.e. this prefix can be used for on-link determination.
- `no-autoconfig`: indicates to hosts on the local link that the specified prefix cannot be used for IPv6 autoconfiguration.

  Default: not set, i.e. prefix can be used for autoconfiguration.

- `router-address`: indicates to hosts on the local link that the specified prefix contains a complete IP address by setting R flag.

  Default: not set, i.e. hosts do not assume a complete IP address is placed.

[no] ipv6 nd ra-interval [(1-1800)]
The maximum time allowed between sending unsolicited multicast router advertisements from the interface, in seconds. Default: `600`

[no] ipv6 nd ra-interval [msec (70-1800000)]
The maximum time allowed between sending unsolicited multicast router advertisements from the interface, in milliseconds. Default: `600000`

[no] ipv6 nd ra-lifetime [(0-9000)]
The value to be placed in the Router Lifetime field of router advertisements sent from the interface, in seconds. Indicates the usefulness of the router as a default router on this interface. Setting the value to zero indicates that the router should not be considered a default router on this interface. Must be either zero or between value specified with `ipv6 nd ra-interval` (or default) and 9000 seconds. Default: `1800`

[no] ipv6 nd reachable-time [(1-3600000)]
The value to be placed in the Reachable Time field in the Router Advertisement messages sent by the router, in milliseconds. The configured time enables the router to detect unavailable neighbors. The value zero means unspecified (by this router). Default: `0`

[no] ipv6 nd managed-config-flag
Set/unset flag in IPv6 router advertisements which indicates to hosts that they should use managed (stateful) protocol for addresses autoconfiguration in addition to any addresses autoconfigured using stateless address autoconfiguration. Default: not set

[no] ipv6 nd other-config-flag
Set/unset flag in IPv6 router advertisements which indicates to hosts that they should use administered (stateful) protocol to obtain autoconfiguration information other than addresses. Default: not set

[no] ipv6 nd home-agent-config-flag
Set/unset flag in IPv6 router advertisements which indicates to hosts that the router acts as a Home Agent and includes a Home Agent Option. Default: not set

[no] ipv6 nd home-agent-preference [(0-65535)]
The value to be placed in Home Agent Option, when Home Agent config flag is set, which indicates to hosts Home Agent preference. The default value of 0 stands for the lowest preference possible. Default: `0`

[no] ipv6 nd home-agent-lifetime [(0-65520)]
The value to be placed in Home Agent Option, when Home Agent config flag is set, which indicates to hosts Home Agent Lifetime. The default value of 0 means to place the current Router Lifetime value.

Default: `0`

[no] ipv6 nd adv-interval-option
Include an Advertisement Interval option which indicates to hosts the maximum time, in milliseconds, between successive unsolicited Router Advertisements. Default: not set

[no] ipv6 nd router-preference [(high|medium|low)]
Set default router preference in IPv6 router advertisements per RFC4191. Default: medium

[no] ipv6 nd mtu [(1-65535)]
Include an MTU (type 5) option in each RA packet to assist the attached hosts in proper interface configuration. The announced value is not verified to be consistent with router interface MTU.

Default: don't advertise any MTU option.::

interface eth0 no ipv6 nd suppress-ra ipv6 nd prefix 2001:0DB8:5009::/64

See also:

- **RFC 2462** (IPv6 Stateless Address Autoconfiguration)
- **RFC 4861** (Neighbor Discovery for IP Version 6 (IPv6))
- **RFC 6275** (Mobility Support in IPv6)
- **RFC 4191** (Default Router Preferences and More-Specific Routes)

CHAPTER 8

Kernel Interface

There are several different methods for reading kernel routing table information, updating kernel routing tables, and for looking up interfaces.

- **ioctl** This method is a very traditional way for reading or writing kernel information. *ioctl* can be used for looking up interfaces and for modifying interface addresses, flags, mtu settings and other types of information. Also, *ioctl* can insert and delete kernel routing table entries. It will soon be available on almost any platform which zebra supports, but it is a little bit ugly thus far, so if a better method is supported by the kernel, zebra will use that.
- **sysctl** This is a program that can lookup kernel information using MIB (Management Information Base) syntax. Normally, it only provides a way of getting information from the kernel. So one would usually want to change kernel information using another method such as *ioctl*.
- **proc filesystem** This is a special filesystem mount that provides an easy way of getting kernel information.
- **routing socket / Netlink** On recent Linux kernels (2.0.x and 2.2.x), there is a kernel/user communication support called *Netlink*. It makes asynchronous communication between kernel and FRR possible, similar to a routing socket on BSD systems.

  Before you use this feature, be sure to select (in kernel configuration) the kernel/Netlink support option 'Kernel/User network link driver' and 'Routing messages'.

  Today, the `/dev/route` special device file is obsolete. Netlink communication is done by reading/writing over Netlink socket.

  After the kernel configuration, please reconfigure and rebuild FRR. You can use Netlink as a dynamic routing update channel between FRR and the kernel.

CHAPTER 9

SNMP Support

SNMP (Simple Network Managing Protocol) is a widely implemented feature for collecting network information from router and/or host. FRR itself does not support SNMP agent (server daemon) functionality but is able to connect to a SNMP agent using the SMUX protocol (**RFC 1227**) or the AgentX protocol (**RFC 2741**) and make the routing protocol MIBs available through it.

Note that SNMP Support needs to be enabled at compile-time and loaded as module on daemon startup. Refer to *Loadable Module Support* on the latter.

9.1 Getting and installing an SNMP agent

There are several SNMP agent which support SMUX or AgentX. We recommend to use the latest version of *net-snmp* which was formerly known as *ucd-snmp*. It is free and open software and available at http://www.net-snmp.org/ and as binary package for most Linux distributions. *net-snmp* has to be compiled with *–with-mib-modules=agentx* to be able to accept connections from FRR using AgentX protocol or with *–with-mib-modules=smux* to use SMUX protocol.

Nowadays, SMUX is a legacy protocol. The AgentX protocol should be preferred for any new deployment. Both protocols have the same coverage.

9.2 AgentX configuration

To enable AgentX protocol support, FRR must have been build with the `--enable-snmp` or *–enable-snmp=agentx* option. Both the master SNMP agent (snmpd) and each of the FRR daemons must be configured. In `/etc/snmp/snmpd.conf`, the `master agentx` directive should be added. In each of the FRR daemons, `agentx` command will enable AgentX support.

`/etc/snmp/snmpd.conf`: # # example access restrictions setup # com2sec readonly default public group MyROGroup v1 readonly view all included .1 80 access MyROGroup “” any noauth exact all none none # # enable master agent for AgentX subagents # master agentx

`/etc/frr/ospfd.conf`:

```
! ... the rest of ospfd.conf has been omitted for clarity ...
!
agentx
!
```

Upon successful connection, you should get something like this in the log of each FRR daemons:

```
2012/05/25 11:39:08 ZEBRA: snmp[info]: NET-SNMP version 5.4.3 AgentX subagent␣
↪connected
```

Then, you can use the following command to check everything works as expected:

```
# snmpwalk -c public -v1 localhost .1.3.6.1.2.1.14.1.1
OSPF-MIB::ospfRouterId.0 = IpAddress: 192.168.42.109
[...]
```

The AgentX protocol can be transported over a Unix socket or using TCP or UDP. It usually defaults to a Unix socket and depends on how NetSNMP was built. If need to configure FRR to use another transport, you can configure it through `/etc/snmp/frr.conf`:

```
[snmpd]
# Use a remote master agent
agentXSocket tcp:192.168.15.12:705
```

9.3 SMUX configuration

To enable SMUX protocol support, FRR must have been build with the *--enable-snmp* option.

A separate connection has then to be established between the SNMP agent (snmpd) and each of the FRR daemons. This connections each use different OID numbers and passwords. Be aware that this OID number is not the one that is used in queries by clients, it is solely used for the intercommunication of the daemons.

In the following example the ospfd daemon will be connected to the snmpd daemon using the password "frr_ospfd". For testing it is recommending to take exactly the below snmpd.conf as wrong access restrictions can be hard to debug.

`/etc/snmp/snmpd.conf`: # # example access restrictions setup # com2sec readonly default public group MyROGroup v1 readonly view all included .1 80 access MyROGroup "" any noauth exact all none none # # the following line is relevant for FRR # smuxpeer .1.3.6.1.4.1.3317.1.2.5 frr_ospfd

`/etc/frr/ospf`: ! ... the rest of ospfd.conf has been omitted for clarity ... ! smux peer .1.3.6.1.4.1.3317.1.2.5 frr_ospfd !

After restarting snmpd and frr, a successful connection can be verified in the syslog and by querying the SNMP daemon:

```
snmpd[12300]: [smux_accept] accepted fd 12 from 127.0.0.1:36255
snmpd[12300]: accepted smux peer: \\
   oid GNOME-PRODUCT-ZEBRA-MIB::ospfd, frr-0.96.5

# snmpwalk -c public -v1 localhost .1.3.6.1.2.1.14.1.1
OSPF-MIB::ospfRouterId.0 = IpAddress: 192.168.42.109
```

Be warned that the current version (5.1.1) of the Net-SNMP daemon writes a line for every SNMP connect to the syslog which can lead to enormous log file sizes. If that is a problem you should consider to patch snmpd and comment out the troublesome *snmp_log()* line in the function *netsnmp_agent_check_packet()* in *agent/snmp_agent.c*.

9.4 MIB and command reference

The following OID numbers are used for the interprocess communication of snmpd and the FRR daemons with SMUX only.:

```
.     (OIDs below .iso.org.dod.internet.private.enterprises)
zebra .1.3.6.1.4.1.3317.1.2.1 .gnome.gnomeProducts.zebra.zserv
bgpd  .1.3.6.1.4.1.3317.1.2.2 .gnome.gnomeProducts.zebra.bgpd
ripd  .1.3.6.1.4.1.3317.1.2.3 .gnome.gnomeProducts.zebra.ripd
ospfd .1.3.6.1.4.1.3317.1.2.5 .gnome.gnomeProducts.zebra.ospfd
ospf6d        .1.3.6.1.4.1.3317.1.2.6 .gnome.gnomeProducts.zebra.ospf6d
```

Sadly, SNMP has not been implemented in all daemons yet. The following OID numbers are used for querying the SNMP daemon by a client::

```
zebra .1.3.6.1.2.1.4.24    .iso.org.dot.internet.mgmt.mib-2.ip.ipForward
ospfd .1.3.6.1.2.1.14      .iso.org.dot.internet.mgmt.mib-2.ospf
bgpd  .1.3.6.1.2.1.15      .iso.org.dot.internet.mgmt.mib-2.bgp
ripd  .1.3.6.1.2.1.23      .iso.org.dot.internet.mgmt.mib-2.rip2
ospf6d        .1.3.6.1.3.102       .iso.org.dod.internet.experimental.ospfv3
```

The following syntax is understood by the FRR daemons for configuring SNMP using SMUX:

smux peer OID

no smux peer OID

smux peer OID PASSWORD

no smux peer OID PASSWORD

Here is the syntax for using AgentX:

agentx

no agentx

9.5 Handling SNMP Traps

To handle snmp traps make sure your snmp setup of frr works correctly as described in the frr documentation in *SNMP Support*.

The BGP4 mib will send traps on peer up/down events. These should be visible in your snmp logs with a message similar to:

```
snmpd[13733]: Got trap from peer on fd 14
```

To react on these traps they should be handled by a trapsink. Configure your trapsink by adding the following lines to `/etc/snmpd/snmpd.conf`:

```
# send traps to the snmptrapd on localhost
trapsink localhost
```

This will send all traps to an snmptrapd running on localhost. You can of course also use a dedicated management station to catch traps. Configure the snmptrapd daemon by adding the following line to `/etc/snmpd/snmptrapd.conf`:

```
traphandle .1.3.6.1.4.1.3317.1.2.2 /etc/snmp/snmptrap_handle.sh
```

This will use the bash script `/etc/snmp/snmptrap_handle.sh` to handle the BGP4 traps. To add traps for other protocol daemons, lookup their appropriate OID from their mib. (For additional information about which traps are supported by your mib, lookup the mib on http://www.oidview.com/mibs/detail.html).

Make sure *snmptrapd* is started.

The snmptrap_handle.sh script I personally use for handling BGP4 traps is below. You can of course do all sorts of things when handling traps, like sound a siren, have your display flash, etc., be creative ;).

```
#!/bin/bash

# routers name
ROUTER=`hostname -s`

#email address use to sent out notification
EMAILADDR="john@doe.com"
#email address used (allongside above) where warnings should be sent
EMAILADDR_WARN="sms-john@doe.com"

# type of notification
TYPE="Notice"

# local snmp community for getting AS belonging to peer
COMMUNITY="<community>"

# if a peer address is in $WARN_PEERS a warning should be sent
WARN_PEERS="192.0.2.1"

# get stdin
INPUT=`cat -`

# get some vars from stdin
uptime=`echo $INPUT | cut -d' ' -f5`
peer=`echo $INPUT | cut -d' ' -f8 | sed -e 's/SNMPv2-SMI::mib-2.15.3.1.14.//g'`
peerstate=`echo $INPUT | cut -d' ' -f13`
errorcode=`echo $INPUT | cut -d' ' -f9 | sed -e 's/\\"//g'`
suberrorcode=`echo $INPUT | cut -d' ' -f10 | sed -e 's/\\"//g'`
remoteas=`snmpget -v2c -c $COMMUNITY localhost SNMPv2-SMI::mib-2.15.3.1.9.$peer | cut␣
↪-d' ' -f4`

WHOISINFO=`whois -h whois.ripe.net " -r AS$remoteas" | egrep '(as-name|descr)'`
asname=`echo "$WHOISINFO" | grep "^as-name:" | sed -e 's/^as-name://g' -e 's/  //g' -
↪e 's/^ //g' | uniq`
asdescr=`echo "$WHOISINFO" | grep "^descr:" | sed -e 's/^descr://g' -e 's/  //g' -e
↪'s/^ //g' | uniq`

# if peer address is in $WARN_PEER, the email should also
# be sent to $EMAILADDR_WARN
for ip in $WARN_PEERS; do
if [ "x$ip" == "x$peer" ]; then
EMAILADDR="$EMAILADDR,$EMAILADDR_WARN"
TYPE="WARNING"
break
fi
done
```

(continues on next page)

(continued from previous page)

```
# convert peer state
case "$peerstate" in
1) peerstate="Idle" ;;
2) peerstate="Connect" ;;
3) peerstate="Active" ;;
4) peerstate="Opensent" ;;
5) peerstate="Openconfirm" ;;
6) peerstate="Established" ;;
*) peerstate="Unknown" ;;
esac

# get textual messages for errors
case "$errorcode" in
00)
error="No error"
suberror=""
;;
01)
error="Message Header Error"
case "$suberrorcode" in
01) suberror="Connection Not Synchronized" ;;
02) suberror="Bad Message Length" ;;
03) suberror="Bad Message Type" ;;
*) suberror="Unknown" ;;
esac
;;
02)
error="OPEN Message Error"
case "$suberrorcode" in
01) suberror="Unsupported Version Number" ;;
02) suberror="Bad Peer AS" ;;
03) suberror="Bad BGP Identifier" ;;
04) suberror="Unsupported Optional Parameter" ;;
05) suberror="Authentication Failure" ;;
06) suberror="Unacceptable Hold Time" ;;
*) suberror="Unknown" ;;
esac
;;
03)
error="UPDATE Message Error"
case "$suberrorcode" in
01) suberror="Malformed Attribute List" ;;
02) suberror="Unrecognized Well-known Attribute" ;;
03) suberror="Missing Well-known Attribute" ;;
04) suberror="Attribute Flags Error" ;;
05) suberror="Attribute Length Error" ;;
06) suberror="Invalid ORIGIN Attribute" ;;
07) suberror="AS Routing Loop" ;;
08) suberror="Invalid NEXT_HOP Attribute" ;;
09) suberror="Optional Attribute Error" ;;
10) suberror="Invalid Network Field" ;;
11) suberror="Malformed AS_PATH" ;;
*) suberror="Unknown" ;;
esac
;;
04)
error="Hold Timer Expired"
```

(continues on next page)

(continued from previous page)

```
suberror=""
;;
05)
error="Finite State Machine Error"
suberror=""
;;
06)
error="Cease"
case "$suberrorcode" in
01) suberror="Maximum Number of Prefixes Reached" ;;
02) suberror="Administratively Shutdown" ;;
03) suberror="Peer Unconfigured" ;;
04) suberror="Administratively Reset" ;;
05) suberror="Connection Rejected" ;;
06) suberror="Other Configuration Change" ;;
07) suberror="Connection collision resolution" ;;
08) suberror="Out of Resource" ;;
09) suberror="MAX" ;;
*) suberror="Unknown" ;;
esac
;;
*)
error="Unknown"
suberror=""
;;
esac

# create textual message from errorcodes
if [ "x$suberror" == "x" ]; then
NOTIFY="$errorcode ($error)"
else
NOTIFY="$errorcode/$suberrorcode ($error/$suberror)"
fi

# form a decent subject
SUBJECT="$TYPE: $ROUTER [bgp] $peer is $peerstate: $NOTIFY"
# create the email body
MAIL=`cat << EOF
BGP notification on router $ROUTER.

Peer: $peer
AS: $remoteas
New state: $peerstate
Notification: $NOTIFY

Info:
$asname
$asdescr

Snmpd uptime: $uptime
EOF`

# mail the notification
echo "$MAIL" | mail -s "$SUBJECT" $EMAILADDR
```

CHAPTER 10

Zebra

zebra is an IP routing manager. It provides kernel routing table updates, interface lookups, and redistribution of routes between different routing protocols.

10.1 Invoking zebra

Besides the common invocation options (*Common Invocation Options*), the *zebra* specific invocation options are listed below.

-b, --batch
Runs in batch mode. *zebra* parses configuration file and terminates immediately.

-k, --keep_kernel
When zebra starts up, don't delete old self inserted routes.

-r, --retain
When program terminates, retain routes added by zebra.

-e X**, --ecmp** X
Run zebra with a limited ecmp ability compared to what it is compiled to. If you are running zebra on hardware limited functionality you can force zebra to limit the maximum ecmp allowed to X. This number is bounded by what you compiled FRR with as the maximum number.

-n, --vrfwnetns
When *Zebra* starts with this option, the VRF backend is based on Linux network namespaces. That implies that all network namespaces discovered by ZEBRA will create an associated VRF. The other daemons will operate on the VRF defined by *Zebra*, as usual.

See also:

VRF (Virtual Routing and Forwarding)

10.2 Configuration Addresses behaviour

At startup, *Zebra* will first discover the underlying networking objects from the operating system. This includes interfaces, addresses of interfaces, static routes, etc. Then, it will read the configuration file, including its own interface addresses, static routes, etc. All this information comprises the operational context from *Zebra*. But configuration context from *Zebra* will remain the same as the one from `zebra.conf` config file. As an example, executing the following `show running-config` will reflect what was in `zebra.conf`. In a similar way, networking objects that are configured outside of the *Zebra* like *iproute2* will not impact the configuration context from *Zebra*. This behaviour permits you to continue saving your own config file, and decide what is really to be pushed on the config file, and what is dependent on the underlying system. Note that inversely, from *Zebra*, you will not be able to delete networking objects that were previously configured outside of *Zebra*.

10.3 Interface Commands

10.3.1 Standard Commands

interface IFNAME

interface IFNAME vrf VRF

shutdown

no shutdown
Up or down the current interface.

ip address ADDRESS/PREFIX

ipv6 address ADDRESS/PREFIX

no ip address ADDRESS/PREFIX

no ipv6 address ADDRESS/PREFIX
Set the IPv4 or IPv6 address/prefix for the interface.

ip address LOCAL-ADDR peer PEER-ADDR/PREFIX

no ip address LOCAL-ADDR peer PEER-ADDR/PREFIX
Configure an IPv4 Point-to-Point address on the interface. (The concept of PtP addressing does not exist for IPv6.)

local-addr has no subnet mask since the local side in PtP addressing is always a single (/32) address. *peer-addr/prefix* can be an arbitrary subnet behind the other end of the link (or even on the link in Point-to-Multipoint setups), though generally /32s are used.

ip address ADDRESS/PREFIX secondary

no ip address ADDRESS/PREFIX secondary
Set the secondary flag for this address. This causes ospfd to not treat the address as a distinct subnet.

description DESCRIPTION ...
Set description for the interface.

multicast

no multicast
Enable or disables multicast flag for the interface.

bandwidth (1-10000000)

no bandwidth (1-10000000)
Set bandwidth value of the interface in kilobits/sec. This is for calculating OSPF cost. This command does not affect the actual device configuration.

link-detect

no link-detect
Enable/disable link-detect on platforms which support this. Currently only Linux and Solaris, and only where network interface drivers support reporting link-state via the `IFF_RUNNING` flag.

10.3.2 Link Parameters Commands

link-params

no link-param
Enter into the link parameters sub node. At least 'enable' must be set to activate the link parameters, and consequently Traffic Engineering on this interface. MPLS-TE must be enable at the OSPF (*Traffic Engineering*) or ISIS (*Traffic Engineering*) router level in complement to this. Disable link parameters for this interface.

Under link parameter statement, the following commands set the different TE values:

link-params [enable]
Enable link parameters for this interface.

link-params [metric (0-4294967295)]

link-params max-bw BANDWIDTH

link-params max-rsv-bw BANDWIDTH

link-params unrsv-bw (0-7) BANDWIDTH

link-params admin-grp BANDWIDTH
These commands specifies the Traffic Engineering parameters of the interface in conformity to RFC3630 (OSPF) or RFC5305 (ISIS). There are respectively the TE Metric (different from the OSPF or ISIS metric), Maximum Bandwidth (interface speed by default), Maximum Reservable Bandwidth, Unreserved Bandwidth for each 0-7 priority and Admin Group (ISIS) or Resource Class/Color (OSPF).

Note that BANDIWDTH is specified in IEEE floating point format and express in Bytes/second.

link-param delay (0-16777215) [min (0-16777215) | max (0-16777215)]

link-param delay-variation (0-16777215)

link-param packet-loss PERCENTAGE

link-param res-bw BANDWIDTH

link-param ava-bw BANDWIDTH

link-param use-bw BANDWIDTH
These command specifies additional Traffic Engineering parameters of the interface in conformity to draft-ietf-ospf-te-metrics-extension-05.txt and draft-ietf-isis-te-metrics-extension-03.txt. There are respectively the delay, jitter, loss, available bandwidth, reservable bandwidth and utilized bandwidth.

Note that BANDWIDTH is specified in IEEE floating point format and express in Bytes/second. Delays and delay variation are express in micro-second (µs). Loss is specified in PERCENTAGE ranging from 0 to 50.331642% by step of 0.000003.

link-param neighbor <A.B.C.D> as (0-65535)

link-param no neighbor
Specifies the remote ASBR IP address and Autonomous System (AS) number for InterASv2 link in OSPF (RFC5392). Note that this option is not yet supported for ISIS (RFC5316).

10.4 Static Route Commands

Static routing is a very fundamental feature of routing technology. It defines static prefix and gateway.

ip route NETWORK GATEWAY
NETWORK is destination prefix with format of A.B.C.D/M. GATEWAY is gateway for the prefix. When GATEWAY is A.B.C.D format. It is taken as a IPv4 address gateway. Otherwise it is treated as an interface name. If the interface name is `null0` then zebra installs a blackhole route.

Some example configuration:

```
ip route 10.0.0.0/8 10.0.0.2
ip route 10.0.0.0/8 ppp0
ip route 10.0.0.0/8 null0
```

First example defines 10.0.0.0/8 static route with gateway 10.0.0.2. Second one defines the same prefix but with gateway to interface ppp0. The third install a blackhole route.

ip route NETWORK NETMASK GATEWAY
This is alternate version of above command. When NETWORK is A.B.C.D format, user must define NETMASK value with A.B.C.D format. GATEWAY is same option as above command.

```
ip route 10.0.0.0 255.255.255.0 10.0.0.2
ip route 10.0.0.0 255.255.255.0 ppp0
ip route 10.0.0.0 255.255.255.0 null0
```

These statements are equivalent to those in the previous example.

ip route NETWORK GATEWAY DISTANCE
Installs the route with the specified distance.

Multiple nexthop static route:

```
ip route 10.0.0.1/32 10.0.0.2
ip route 10.0.0.1/32 10.0.0.3
ip route 10.0.0.1/32 eth0
```

If there is no route to 10.0.0.2 and 10.0.0.3, and interface eth0 is reachable, then the last route is installed into the kernel.

If zebra has been compiled with multipath support, and both 10.0.0.2 and 10.0.0.3 are reachable, zebra will install a multipath route via both nexthops, if the platform supports this.

```
zebra> show ip route
S>  10.0.0.1/32 [1/0] via 10.0.0.2 inactive
    via 10.0.0.3 inactive
  *       is directly connected, eth0
```

```
ip route 10.0.0.0/8 10.0.0.2
ip route 10.0.0.0/8 10.0.0.3
ip route 10.0.0.0/8 null0 255
```

This will install a multihop route via the specified next-hops if they are reachable, as well as a high-metric blackhole route, which can be useful to prevent traffic destined for a prefix to match less-specific routes (e.g. default) should the specified gateways not be reachable. E.g.:

```
zebra> show ip route 10.0.0.0/8
Routing entry for 10.0.0.0/8
  Known via "static", distance 1, metric 0
    10.0.0.2 inactive
    10.0.0.3 inactive

Routing entry for 10.0.0.0/8
  Known via "static", distance 255, metric 0
    directly connected, Null0
```

ipv6 route NETWORK GATEWAY

ipv6 route NETWORK GATEWAY DISTANCE
These behave similarly to their ipv4 counterparts.

ipv6 route NETWORK from SRCPREFIX GATEWAY

ipv6 route NETWORK from SRCPREFIX GATEWAY DISTANCE
Install a static source-specific route. These routes are currently supported on Linux operating systems only, and perform AND matching on packet's destination and source addresses in the kernel's forwarding path. Note that destination longest-prefix match is "more important" than source LPM, e.g. *"2001:db8:1::/64 from 2001:db8::/48"* will win over *"2001:db8::/48 from 2001:db8:1::/64"* if both match.

table TABLENO
Select the primary kernel routing table to be used. This only works for kernels supporting multiple routing tables (like GNU/Linux 2.2.x and later). After setting TABLENO with this command, static routes defined after this are added to the specified table.

10.5 VRF (Virtual Routing and Forwarding)

Currently, the user has the possibility to configure VRFs. VRF is a way to separate networking contexts on the same machine. Those networking contexts are associated with separate interfaces, thus making it possible to associate one interface with a specific VRF. VRF can be used, for example, when instantiating per enterprise networking services, without having to instantiate the physical host machine or the routing management daemons for each enterprise. As a result, interfaces are separate for each set of VRF, and routing daemons can have their own context for each VRF.

This conceptual view introduces the *Default VRF* case. If the user does not configure any specific VRF, then by default, the user will however configure the *Default VRF*. On the *Zebra* context, this can be done when being in configuration mode, when configuring a static route clicmd:*ip route NETWORK GATEWAY*.

```
# case without VRF
configure terminal
 ip route 10.0.0.0 255.255.255.0 10.0.0.2
exit
```

Configuring VRF networking contexts can be done in various ways on FRR. The VRF interfaces can be configured by entering in interface configuration mode : `interface IFNAME vrf VRF`. Also, if the user wants to configure a static route for a specific VRF, then a specific VRF configuration mode is available. After entering into that mode by following command: `vrf VRF`. the user can enter the same route command as before, but this time, the route command will apply to vrf VRF.

```
# case with VRF
configure terminal
vrf r1-cust1
 ip route 10.0.0.0 255.255.255.0 10.0.0.2
exit-vrf
```

A VRF backend mode is chosen when running *Zebra*.

If no option is chosen, then the *Linux VRF* implementation as references in *https://www.kernel.org/doc/Documentation/networking/vrf.txt* will be mapped over the *Zebra* VRF. The routing table associated to that VRF is a Linux table identifier located in the same *Linux network namespace* where *Zebra* started.

If the `-n` option is chosen, then the *Linux network namespace* will be mapped over the *Zebra* VRF. That implies that *Zebra* is able to configure several *Linux network namespaces*. The routing table associated to that VRF is the whole routing tables located in that namespace. For instance, this mode matches OpenStack Network Namespaces. It matches also OpenFastPath. The default behavior remains Linux VRF which is supported by the Linux kernel community, see *https://www.kernel.org/doc/Documentation/networking/vrf.txt*.

Because of that difference, there are some subtle differences when running some commands in relationship to VRF. Here is an extract of some of those commands:

vrf VRF
: This command is available on configuration mode. By default, above command permits accessing the vrf configuration mode. This mode is available for both VRFs. It is to be noted that *Zebra* does not create Linux VRF. The network administrator can however decide to provision this command in configuration file to provide more clarity about the intended configuration.

netns NAMESPACE
: This command is based on VRF configuration mode. This command is available when *Zebra* is run in `-n` mode. This command reflects which *Linux network namespace* is to be mapped with *Zebra* VRF. It is to be noted that *Zebra* creates and detects added/suppressed VRFs from the Linux environment (in fact, those managed with iproute2). The network administrator can however decide to provision this command in configuration file to provide more clarity about the intended configuration.

ip route NETWORK NETMASK GATEWAY NEXTHOPVRF
: This command is based on VRF configuration mode or in configuration mode. If on configuration mode, this applies to default VRF. Otherwise, this command applies to the VRF of the vrf configuration mode. This command is used to configure a vrf route leak across 2 VRFs. This command is only available when *Zebra* is launched without `-n` option.

ip route NETWORK NETMASK GATEWAY table TABLENO
: This command is based on VRF configuration mode. There, this command is only available with `-n` command. This commands permits configuring a network route in the given `TABLENO` of the *Linux network namespace*.

ip route NETWORK NETMASK GATEWAY table TABLENO
: This command is based on configuration mode. There, for default VRF, this command is available for all modes. The `TABLENO` configured is one of the tables from Default *Linux network namespace*.

show ip route vrf VRF
: The show command permits dumping the routing table associated to the VRF. If *Zebra* is launched with default settings, this will be the `TABLENO` of the VRF configured on the kernel, thanks to information provided in *https://www.kernel.org/doc/Documentation/networking/vrf.txt*. If *Zebra* is launched with `-n` option, this will be the default routing table of the *Linux network namespace* `VRF`.

show ip route vrf VRF table TABLENO
: The show command is only available with `-n` option. This command will dump the routing table `TABLENO` of the *Linux network namespace* `VRF`.

```
ip route 10.0.0.0 255.255.255.0 10.0.0.2 vrf r1-cust1 table 43
show ip table vrf r1-cust1 table 43
```

10.6 Multicast RIB Commands

The Multicast RIB provides a separate table of unicast destinations which is used for Multicast Reverse Path Forwarding decisions. It is used with a multicast source's IP address, hence contains not multicast group addresses but unicast addresses.

This table is fully separate from the default unicast table. However, RPF lookup can include the unicast table.

WARNING: RPF lookup results are non-responsive in this version of FRR, i.e. multicast routing does not actively react to changes in underlying unicast topology!

ip multicast rpf-lookup-mode MODE

no ip multicast rpf-lookup-mode [MODE]

MODE sets the method used to perform RPF lookups. Supported modes:

urib-only Performs the lookup on the Unicast RIB. The Multicast RIB is never used.

mrib-only Performs the lookup on the Multicast RIB. The Unicast RIB is never used.

mrib-then-urib Tries to perform the lookup on the Multicast RIB. If any route is found, that route is used. Otherwise, the Unicast RIB is tried.

lower-distance Performs a lookup on the Multicast RIB and Unicast RIB each. The result with the lower administrative distance is used; if they're equal, the Multicast RIB takes precedence.

longer-prefix Performs a lookup on the Multicast RIB and Unicast RIB each. The result with the longer prefix length is used; if they're equal, the Multicast RIB takes precedence.

The *mrib-then-urib* setting is the default behavior if nothing is configured. If this is the desired behavior, it should be explicitly configured to make the configuration immune against possible changes in what the default behavior is.

Warning: Unreachable routes do not receive special treatment and do not cause fallback to a second lookup.

show ip rpf ADDR

Performs a Multicast RPF lookup, as configured with `ip multicast rpf-lookup-mode MODE`. ADDR specifies the multicast source address to look up.

```
> show ip rpf 192.0.2.1
Routing entry for 192.0.2.0/24 using Unicast RIB

Known via "kernel", distance 0, metric 0, best
* 198.51.100.1, via eth0
```

Indicates that a multicast source lookup for 192.0.2.1 would use an Unicast RIB entry for 192.0.2.0/24 with a gateway of 198.51.100.1.

show ip rpf

Prints the entire Multicast RIB. Note that this is independent of the configured RPF lookup mode, the Multicast RIB may be printed yet not used at all.

ip mroute PREFIX NEXTHOP [DISTANCE]

no ip mroute PREFIX NEXTHOP [DISTANCE]
Adds a static route entry to the Multicast RIB. This performs exactly as the `ip route` command, except that it inserts the route in the Multicast RIB instead of the Unicast RIB.

10.7 zebra Route Filtering

Zebra supports *prefix-list* s and *Route Maps* s to match routes received from other FRR components. The permit/deny facilities provided by these commands can be used to filter which routes zebra will install in the kernel.

ip protocol PROTOCOL route-map ROUTEMAP
Apply a route-map filter to routes for the specified protocol. PROTOCOL can be **any** or one of

- system,
- kernel,
- connected,
- static,
- rip,
- ripng,
- ospf,
- ospf6,
- isis,
- bgp,
- hsls.

set src ADDRESS
Within a route-map, set the preferred source address for matching routes when installing in the kernel.

The following creates a prefix-list that matches all addresses, a route-map that sets the preferred source address, and applies the route-map to all *rip* routes.

```
ip prefix-list ANY permit 0.0.0.0/0 le 32
route-map RM1 permit 10
     match ip address prefix-list ANY
     set src 10.0.0.1

ip protocol rip route-map RM1
```

10.8 zebra FIB push interface

Zebra supports a 'FIB push' interface that allows an external component to learn the forwarding information computed by the FRR routing suite. This is a loadable module that needs to be enabled at startup as described in *Loadable Module Support*.

In FRR, the Routing Information Base (RIB) resides inside zebra. Routing protocols communicate their best routes to zebra, and zebra computes the best route across protocols for each prefix. This latter information makes up the Forwarding Information Base (FIB). Zebra feeds the FIB to the kernel, which allows the IP stack in the kernel to forward packets according to the routes computed by FRR. The kernel FIB is updated in an OS-specific way. For example, the *Netlink* interface is used on Linux, and route sockets are used on FreeBSD.

The FIB push interface aims to provide a cross-platform mechanism to support scenarios where the router has a forwarding path that is distinct from the kernel, commonly a hardware-based fast path. In these cases, the FIB needs to be maintained reliably in the fast path as well. We refer to the component that programs the forwarding plane (directly or indirectly) as the Forwarding Plane Manager or FPM.

The FIB push interface comprises of a TCP connection between zebra and the FPM. The connection is initiated by zebra – that is, the FPM acts as the TCP server.

The relevant zebra code kicks in when zebra is configured with the *--enable-fpm* flag. Zebra periodically attempts to connect to the well-known FPM port. Once the connection is up, zebra starts sending messages containing routes over the socket to the FPM. Zebra sends a complete copy of the forwarding table to the FPM, including routes that it may have picked up from the kernel. The existing interaction of zebra with the kernel remains unchanged – that is, the kernel continues to receive FIB updates as before.

The encapsulation header for the messages exchanged with the FPM is defined by the file `fpm/fpm.h` in the frr tree. The routes themselves are encoded in Netlink or protobuf format, with Netlink being the default.

Protobuf is one of a number of new serialization formats wherein the message schema is expressed in a purpose-built language. Code for encoding/decoding to/from the wire format is generated from the schema. Protobuf messages can be extended easily while maintaining backward-compatibility with older code. Protobuf has the following advantages over Netlink:

- Code for serialization/deserialization is generated automatically. This reduces the likelihood of bugs, allows third-party programs to be integrated quickly, and makes it easy to add fields.
- The message format is not tied to an OS (Linux), and can be evolved independently.

As mentioned before, zebra encodes routes sent to the FPM in Netlink format by default. The format can be controlled via the FPM module's load-time option to zebra, which currently takes the values *Netlink* and *protobuf*.

The zebra FPM interface uses replace semantics. That is, if a 'route add' message for a prefix is followed by another 'route add' message, the information in the second message is complete by itself, and replaces the information sent in the first message.

If the connection to the FPM goes down for some reason, zebra sends the FPM a complete copy of the forwarding table(s) when it reconnects.

10.9 zebra Terminal Mode Commands

show ip route
Display current routes which zebra holds in its database.

```
Router# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
 B - BGP * - FIB route.

K* 0.0.0.0/0         203.181.89.241
S  0.0.0.0/0         203.181.89.1
C* 127.0.0.0/8       lo
C* 203.181.89.240/28      eth0
```

show ipv6 route

show interface

show ip prefix-list [NAME]

show route-map [NAME]

show ip protocol

show ipforward
Display whether the host's IP forwarding function is enabled or not. Almost any UNIX kernel can be configured with IP forwarding disabled. If so, the box can't work as a router.

show ipv6forward
Display whether the host's IP v6 forwarding is enabled or not.

show zebra
Display various statistics related to the installation and deletion of routes, neighbor updates, and LSP's into the kernel.

show zebra fpm stats
Display statistics related to the zebra code that interacts with the optional Forwarding Plane Manager (FPM) component.

clear zebra fpm stats
Reset statistics related to the zebra code that interacts with the optional Forwarding Plane Manager (FPM) component.

CHAPTER 11

BGP

BGP stands for a Border Gateway Protocol. The latest BGP version is 4. BGP-4 is one of the Exterior Gateway Protocols and the de facto standard interdomain routing protocol. BGP-4 is described in **RFC 1771**.

Many extensions have been added to **RFC 1771**. **RFC 2858** adds multiprotocol support to BGP-4.

11.1 Starting BGP

Default configuration file of *bgpd* is `bgpd.conf`. *bgpd* searches the current directory first then /etc/frr/bgpd.conf. All of bgpd's command must be configured in `bgpd.conf`.

bgpd specific invocation options are described below. Common options may also be specified (*Common Invocation Options*).

-p, --bgp_port <port>
: Set the bgp protocol's port number. When port number is 0, that means do not listen bgp port.

-r, --retain
: When program terminates, retain BGP routes added by zebra.

-l, --listenon
: Specify a specific IP address for bgpd to listen on, rather than its default of INADDR_ANY / IN6ADDR_ANY. This can be useful to constrain bgpd to an internal address, or to run multiple bgpd processes on one host.

11.2 BGP router

First of all you must configure BGP router with *router bgp* command. To configure BGP router, you need AS number. AS number is an identification of autonomous system. BGP protocol uses the AS number for detecting whether the BGP connection is internal one or external one.

router bgp ASN
: Enable a BGP protocol process with the specified ASN. After this statement you can input any *BGP Commands*.

You can not create different BGP process under different ASN without specifying *multiple-instance* (*Multiple instance*).

no router bgp ASN
Destroy a BGP protocol process with the specified ASN.

bgp router-id A.B.C.D
This command specifies the router-ID. If *bgpd* connects to *zebra* it gets interface and address information. In that case default router ID value is selected as the largest IP Address of the interfaces. When *router zebra* is not enabled *bgpd* can't get interface information so *router-id* is set to 0.0.0.0. So please set router-id by hand.

11.2.1 BGP distance

distance bgp (1-255) (1-255) (1-255)
This command change distance value of BGP. Each argument is distance value for external routes, internal routes and local routes.

distance (1-255) A.B.C.D/M

distance (1-255) A.B.C.D/M word

11.2.2 BGP decision process

The decision process FRR BGP uses to select routes is as follows:

1. *Weight check* Prefer higher local weight routes to lower routes.
2. *Local preference check* Prefer higher local preference routes to lower.
3. *Local route check* Prefer local routes (statics, aggregates, redistributed) to received routes.
4. *AS path length check* Prefer shortest hop-count AS_PATHs.
5. *Origin check* Prefer the lowest origin type route. That is, prefer IGP origin routes to EGP, to Incomplete routes.
6. *MED check* Where routes with a MED were received from the same AS, prefer the route with the lowest MED. *BGP MED*.
7. *External check* Prefer the route received from an external, eBGP peer over routes received from other types of peers.
8. *IGP cost check* Prefer the route with the lower IGP cost.
9. *Multi-path check* If multi-pathing is enabled, then check whether the routes not yet distinguished in preference may be considered equal. If `bgp bestpath as-path multipath-relax` is set, all such routes are considered equal, otherwise routes received via iBGP with identical AS_PATHs or routes received from eBGP neighbours in the same AS are considered equal.
10. *Already-selected external check* Where both routes were received from eBGP peers, then prefer the route which is already selected. Note that this check is not applied if `bgp bestpath compare-routerid` is configured. This check can prevent some cases of oscillation.
11. *Router-ID check* Prefer the route with the lowest *router-ID*. If the route has an *ORIGINATOR_ID* attribute, through iBGP reflection, then that router ID is used, otherwise the *router-ID* of the peer the route was received from is used.
12. *Cluster-List length check* The route with the shortest cluster-list length is used. The cluster-list reflects the iBGP reflection path the route has taken.
13. *Peer address* Prefer the route received from the peer with the higher transport layer address, as a last-resort tie-breaker.

bgp bestpath as-path confed
This command specifies that the length of confederation path sets and sequences should should be taken into account during the BGP best path decision process.

bgp bestpath as-path multipath-relax
This command specifies that BGP decision process should consider paths of equal AS_PATH length candidates for multipath computation. Without the knob, the entire AS_PATH must match for multipath computation.

bgp bestpath compare-routerid
Ensure that when comparing routes where both are equal on most metrics, including local-pref, AS_PATH length, IGP cost, MED, that the tie is broken based on router-ID.

If this option is enabled, then the already-selected check, where already selected eBGP routes are preferred, is skipped.

If a route has an *ORIGINATOR_ID* attribute because it has been reflected, that *ORIGINATOR_ID* will be used. Otherwise, the router-ID of the peer the route was received from will be used.

The advantage of this is that the route-selection (at this point) will be more deterministic. The disadvantage is that a few or even one lowest-ID router may attract all traffic to otherwise-equal paths because of this check. It may increase the possibility of MED or IGP oscillation, unless other measures were taken to avoid these. The exact behaviour will be sensitive to the iBGP and reflection topology.

11.2.3 BGP route flap dampening

bgp dampening (1-45) (1-20000) (1-20000) (1-255)
This command enables BGP route-flap dampening and specifies dampening parameters.

half-life Half-life time for the penalty

reuse-threshold Value to start reusing a route

suppress-threshold Value to start suppressing a route

max-suppress Maximum duration to suppress a stable route

The route-flap damping algorithm is compatible with **RFC 2439**. The use of this command is not recommended nowadays.

See also:

http://www.ripe.net/ripe/docs/ripe-378,,RIPE-378

11.3 BGP MED

The BGP MED (Multi Exit Discriminator) attribute has properties which can cause subtle convergence problems in BGP. These properties and problems have proven to be hard to understand, at least historically, and may still not be widely understood. The following attempts to collect together and present what is known about MED, to help operators and FRR users in designing and configuring their networks.

The BGP MED attribute is intended to allow one AS to indicate its preferences for its ingress points to another AS. The MED attribute will not be propagated on to another AS by the receiving AS - it is 'non-transitive' in the BGP sense.

E.g., if AS X and AS Y have 2 different BGP peering points, then AS X might set a MED of 100 on routes advertised at one and a MED of 200 at the other. When AS Y selects between otherwise equal routes to or via AS X, AS Y should prefer to take the path via the lower MED peering of 100 with AS X. Setting the MED allows an AS to influence the routing taken to it within another, neighbouring AS.

In this use of MED it is not really meaningful to compare the MED value on routes where the next AS on the paths differs. E.g., if AS Y also had a route for some destination via AS Z in addition to the routes from AS X, and AS Z had also set a MED, it wouldn't make sense for AS Y to compare AS Z's MED values to those of AS X. The MED values have been set by different administrators, with different frames of reference.

The default behaviour of BGP therefore is to not compare MED values across routes received from different neighbouring ASes. In FRR this is done by comparing the neighbouring, left-most AS in the received AS_PATHs of the routes and only comparing MED if those are the same.

Unfortunately, this behaviour of MED, of sometimes being compared across routes and sometimes not, depending on the properties of those other routes, means MED can cause the order of preference over all the routes to be undefined. That is, given routes A, B, and C, if A is preferred to B, and B is preferred to C, then a well-defined order should mean the preference is transitive (in the sense of orders[1]) and that A would be preferred to C.

However, when MED is involved this need not be the case. With MED it is possible that C is actually preferred over A. So A is preferred to B, B is preferred to C, but C is preferred to A. This can be true even where BGP defines a deterministic 'most preferred' route out of the full set of A,B,C. With MED, for any given set of routes there may be a deterministically preferred route, but there need not be any way to arrange them into any order of preference. With unmodified MED, the order of preference of routes literally becomes undefined.

That MED can induce non-transitive preferences over routes can cause issues. Firstly, it may be perceived to cause routing table churn locally at speakers; secondly, and more seriously, it may cause routing instability in iBGP topologies, where sets of speakers continually oscillate between different paths.

The first issue arises from how speakers often implement routing decisions. Though BGP defines a selection process that will deterministically select the same route as best at any given speaker, even with MED, that process requires evaluating all routes together. For performance and ease of implementation reasons, many implementations evaluate route preferences in a pair-wise fashion instead. Given there is no well-defined order when MED is involved, the best route that will be chosen becomes subject to implementation details, such as the order the routes are stored in. That may be (locally) non-deterministic, e.g.: it may be the order the routes were received in.

This indeterminism may be considered undesirable, though it need not cause problems. It may mean additional routing churn is perceived, as sometimes more updates may be produced than at other times in reaction to some event .

This first issue can be fixed with a more deterministic route selection that ensures routes are ordered by the neighbouring AS during selection. `bgp deterministic-med`. This may reduce the number of updates as routes are received, and may in some cases reduce routing churn. Though, it could equally deterministically produce the largest possible set of updates in response to the most common sequence of received updates.

A deterministic order of evaluation tends to imply an additional overhead of sorting over any set of n routes to a destination. The implementation of deterministic MED in FRR scales significantly worse than most sorting algorithms at present, with the number of paths to a given destination. That number is often low enough to not cause any issues, but where there are many paths, the deterministic comparison may quickly become increasingly expensive in terms of CPU.

Deterministic local evaluation can *not* fix the second, more major, issue of MED however. Which is that the non-transitive preference of routes MED can cause may lead to routing instability or oscillation across multiple speakers in iBGP topologies. This can occur with full-mesh iBGP, but is particularly problematic in non-full-mesh iBGP topologies that further reduce the routing information known to each speaker. This has primarily been documented with iBGP route-reflection topologies. However, any route-hiding technologies potentially could also exacerbate oscillation with MED.

This second issue occurs where speakers each have only a subset of routes, and there are cycles in the preferences between different combinations of routes - as the undefined order of preference of MED allows - and the routes are

[1] For some set of objects to have an order, there *must* be some binary ordering relation that is defined for *every* combination of those objects, and that relation *must* be transitive. I.e.:, if the relation operator is <, and if a < b and b < c then that relation must carry over and it *must* be that a < c for the objects to have an order. The ordering relation may allow for equality, i.e. a < b and b < a may both be true and imply that a and b are equal in the order and not distinguished by it, in which case the set has a partial order. Otherwise, if there is an order, all the objects have a distinct place in the order and the set has a total order)

distributed in a way that causes the BGP speakers to 'chase' those cycles. This can occur even if all speakers use a deterministic order of evaluation in route selection.

E.g., speaker 4 in AS A might receive a route from speaker 2 in AS X, and from speaker 3 in AS Y; while speaker 5 in AS A might receive that route from speaker 1 in AS Y. AS Y might set a MED of 200 at speaker 1, and 100 at speaker 3. I.e, using ASN:ID:MED to label the speakers:

```
.
           /---------------\\
X:2------|--A:4-------A:5--|-Y:1:200
             Y:3:100--|-/    |
           \\---------------/
```

Assuming all other metrics are equal (AS_PATH, ORIGIN, 0 IGP costs), then based on the RFC4271 decision process speaker 4 will choose X:2 over Y:3:100, based on the lower ID of 2. Speaker 4 advertises X:2 to speaker 5. Speaker 5 will continue to prefer Y:1:200 based on the ID, and advertise this to speaker 4. Speaker 4 will now have the full set of routes, and the Y:1:200 it receives from 5 will beat X:2, but when speaker 4 compares Y:1:200 to Y:3:100 the MED check now becomes active as the ASes match, and now Y:3:100 is preferred. Speaker 4 therefore now advertises Y:3:100 to 5, which will also agrees that Y:3:100 is preferred to Y:1:200, and so withdraws the latter route from 4. Speaker 4 now has only X:2 and Y:3:100, and X:2 beats Y:3:100, and so speaker 4 implicitly updates its route to speaker 5 to X:2. Speaker 5 sees that Y:1:200 beats X:2 based on the ID, and advertises Y:1:200 to speaker 4, and the cycle continues.

The root cause is the lack of a clear order of preference caused by how MED sometimes is and sometimes is not compared, leading to this cycle in the preferences between the routes:

```
.
 /---> X:2 ---beats---> Y:3:100 --\\
|                                  |
|                                  |
 \\---beats--- Y:1:200 <---beats---/
```

This particular type of oscillation in full-mesh iBGP topologies can be avoided by speakers preferring already selected, external routes rather than choosing to update to new a route based on a post-MED metric (e.g. router-ID), at the cost of a non-deterministic selection process. FRR implements this, as do many other implementations, so long as it is not overridden by setting `bgp bestpath compare-routerid`, and see also *BGP decision process*.

However, more complex and insidious cycles of oscillation are possible with iBGP route-reflection, which are not so easily avoided. These have been documented in various places. See, e.g.:

- [bgp-route-osci-cond]
- [stable-flexible-ibgp]
- [ibgp-correctness]

for concrete examples and further references.

There is as of this writing *no* known way to use MED for its original purpose; *and* reduce routing information in iBGP topologies; *and* be sure to avoid the instability problems of MED due the non-transitive routing preferences it can induce; in general on arbitrary networks.

There may be iBGP topology specific ways to reduce the instability risks, even while using MED, e.g.: by constraining the reflection topology and by tuning IGP costs between route-reflector clusters, see **RFC 3345** for details. In the near future, the Add-Path extension to BGP may also solve MED oscillation while still allowing MED to be used as intended, by distributing "best-paths per neighbour AS". This would be at the cost of distributing at least as many routes to all speakers as a full-mesh iBGP would, if not more, while also imposing similar CPU overheads as the "Deterministic MED" feature at each Add-Path reflector.

More generally, the instability problems that MED can introduce on more complex, non-full-mesh, iBGP topologies may be avoided either by:

- Setting `bgp always-compare-med`, however this allows MED to be compared across values set by different neighbour ASes, which may not produce coherent desirable results, of itself.
- Effectively ignoring MED by setting MED to the same value (e.g.: 0) using `set metric METRIC` on all received routes, in combination with setting `bgp always-compare-med` on all speakers. This is the simplest and most performant way to avoid MED oscillation issues, where an AS is happy not to allow neighbours to inject this problematic metric.

As MED is evaluated after the AS_PATH length check, another possible use for MED is for intra-AS steering of routes with equal AS_PATH length, as an extension of the last case above. As MED is evaluated before IGP metric, this can allow cold-potato routing to be implemented to send traffic to preferred hand-offs with neighbours, rather than the closest hand-off according to the IGP metric.

Note that even if action is taken to address the MED non-transitivity issues, other oscillations may still be possible. E.g., on IGP cost if iBGP and IGP topologies are at cross-purposes with each other - see the Flavel and Roughan paper above for an example. Hence the guideline that the iBGP topology should follow the IGP topology.

bgp deterministic-med
Carry out route-selection in way that produces deterministic answers locally, even in the face of MED and the lack of a well-defined order of preference it can induce on routes. Without this option the preferred route with MED may be determined largely by the order that routes were received in.

Setting this option will have a performance cost that may be noticeable when there are many routes for each destination. Currently in FRR it is implemented in a way that scales poorly as the number of routes per destination increases.

The default is that this option is not set.

Note that there are other sources of indeterminism in the route selection process, specifically, the preference for older and already selected routes from eBGP peers, *BGP decision process*.

bgp always-compare-med
Always compare the MED on routes, even when they were received from different neighbouring ASes. Setting this option makes the order of preference of routes more defined, and should eliminate MED induced oscillations.

If using this option, it may also be desirable to use `set metric METRIC` to set MED to 0 on routes received from external neighbours.

This option can be used, together with `set metric METRIC` to use MED as an intra-AS metric to steer equal-length AS_PATH routes to, e.g., desired exit points.

11.4 BGP network

11.4.1 BGP route

network A.B.C.D/M
This command adds the announcement network.

```
router bgp 1
 address-family ipv4 unicast
  network 10.0.0.0/8
 exit-address-family
```

This configuration example says that network 10.0.0.0/8 will be announced to all neighbors. Some vendors' routers don't advertise routes if they aren't present in their IGP routing tables; *bgpd* doesn't care about IGP routes when announcing its routes.

no network A.B.C.D/M

11.4.2 Route Aggregation

aggregate-address A.B.C.D/M
This command specifies an aggregate address.

aggregate-address A.B.C.D/M as-set
This command specifies an aggregate address. Resulting routes include AS set.

aggregate-address A.B.C.D/M summary-only
This command specifies an aggregate address. Aggregated routes will not be announce.

no aggregate-address A.B.C.D/M

11.4.3 Redistribute to BGP

redistribute kernel
Redistribute kernel route to BGP process.

redistribute static
Redistribute static route to BGP process.

redistribute connected
Redistribute connected route to BGP process.

redistribute rip
Redistribute RIP route to BGP process.

redistribute ospf
Redistribute OSPF route to BGP process.

redistribute vpn
Redistribute VNC routes to BGP process.

update-delay MAX-DELAY

update-delay MAX-DELAY ESTABLISH-WAIT
This feature is used to enable read-only mode on BGP process restart or when BGP process is cleared using 'clear ip bgp *'. When applicable, read-only mode would begin as soon as the first peer reaches Established status and a timer for max-delay seconds is started.

During this mode BGP doesn't run any best-path or generate any updates to its peers. This mode continues until:

1. All the configured peers, except the shutdown peers, have sent explicit EOR (End-Of-RIB) or an implicit-EOR. The first keep-alive after BGP has reached Established is considered an implicit-EOR. If the establish-wait optional value is given, then BGP will wait for peers to reach established from the beginning of the update-delay till the establish-wait period is over, i.e. the minimum set of established peers for which EOR is expected would be peers established during the establish-wait window, not necessarily all the configured neighbors.
2. max-delay period is over.

On hitting any of the above two conditions, BGP resumes the decision process and generates updates to its peers.

Default max-delay is 0, i.e. the feature is off by default.

table-map ROUTE-MAP-NAME
This feature is used to apply a route-map on route updates from BGP to Zebra. All the applicable match operations are allowed, such as match on prefix, next-hop, communities, etc. Set operations for this attach-point are limited to metric and next-hop only. Any operation of this feature does not affect BGPs internal RIB.

Supported for ipv4 and ipv6 address families. It works on multi-paths as well, however, metric setting is based on the best-path only.

11.5 BGP Peer

11.5.1 Defining Peer

neighbor PEER remote-as ASN
Creates a new neighbor whose remote-as is ASN. PEER can be an IPv4 address or an IPv6 address or an interface to use for the connection.

```
router bgp 1
 neighbor 10.0.0.1 remote-as 2
```

In this case my router, in AS-1, is trying to peer with AS-2 at 10.0.0.1.

This command must be the first command used when configuring a neighbor. If the remote-as is not specified, *bgpd* will complain like this:

```
can't find neighbor 10.0.0.1
```

neighbor PEER remote-as internal
Create a peer as you would when you specify an ASN, except that if the peers ASN is different than mine as specified under the `router bgp ASN` command the connection will be denied.

neighbor PEER remote-as external
Create a peer as you would when you specify an ASN, except that if the peers ASN is the same as mine as specified under the `router bgp ASN` command the connection will be denied.

11.5.2 BGP Peer commands

In a *router bgp* clause there are neighbor specific configurations required.

neighbor PEER shutdown

no neighbor PEER shutdown
Shutdown the peer. We can delete the neighbor's configuration by `no neighbor PEER remote-as ASN` but all configuration of the neighbor will be deleted. When you want to preserve the configuration, but want to drop the BGP peer, use this syntax.

neighbor PEER ebgp-multihop

no neighbor PEER ebgp-multihop

neighbor PEER description ...

no neighbor PEER description ...
Set description of the peer.

neighbor PEER version VERSION
Set up the neighbor's BGP version. *version* can be *4*, *4+* or *4-*. BGP version *4* is the default value used for BGP peering. BGP version *4+* means that the neighbor supports Multiprotocol Extensions for BGP-4. BGP version

4- is similar but the neighbor speaks the old Internet-Draft revision 00's Multiprotocol Extensions for BGP-4. Some routing software is still using this version.

neighbor PEER interface IFNAME

no neighbor PEER interface IFNAME

When you connect to a BGP peer over an IPv6 link-local address, you have to specify the IFNAME of the interface used for the connection. To specify IPv4 session addresses, see the `neighbor PEER update-source` command below.

This command is deprecated and may be removed in a future release. Its use should be avoided.

neighbor PEER next-hop-self [all]

no neighbor PEER next-hop-self [all]

This command specifies an announced route's nexthop as being equivalent to the address of the bgp router if it is learned via eBGP. If the optional keyword *all* is specified the modification is done also for routes learned via iBGP.

neighbor PEER update-source <IFNAME|ADDRESS>

no neighbor PEER update-source

Specify the IPv4 source address to use for the BGP session to this neighbour, may be specified as either an IPv4 address directly or as an interface name (in which case the *zebra* daemon MUST be running in order for *bgpd* to be able to retrieve interface state).

```
router bgp 64555
 neighbor foo update-source 192.168.0.1
 neighbor bar update-source lo0
```

neighbor PEER default-originate

no neighbor PEER default-originate

bgpd's default is to not announce the default route (0.0.0.0/0) even if it is in routing table. When you want to announce default routes to the peer, use this command.

neighbor PEER port PORT

neighbor PEER send-community

neighbor PEER weight WEIGHT

no neighbor PEER weight WEIGHT

This command specifies a default *weight* value for the neighbor's routes.

neighbor PEER maximum-prefix NUMBER

no neighbor PEER maximum-prefix NUMBER

neighbor PEER local-as AS-NUMBER

neighbor PEER local-as AS-NUMBER no-prepend

neighbor PEER local-as AS-NUMBER no-prepend replace-as

no neighbor PEER local-as

Specify an alternate AS for this BGP process when interacting with the specified peer. With no modifiers, the specified local-as is prepended to the received AS_PATH when receiving routing updates from the peer, and prepended to the outgoing AS_PATH (after the process local AS) when transmitting local routes to the peer.

If the no-prepend attribute is specified, then the supplied local-as is not prepended to the received AS_PATH.

If the replace-as attribute is specified, then only the supplied local-as is prepended to the AS_PATH when transmitting local-route updates to this peer.

Note that replace-as can only be specified if no-prepend is.

This command is only allowed for eBGP peers.

neighbor PEER ttl-security hops NUMBER

no neighbor PEER ttl-security hops NUMBER
This command enforces Generalized TTL Security Mechanism (GTSM), as specified in RFC 5082. With this command, only neighbors that are the specified number of hops away will be allowed to become neighbors. This command is mutually exclusive with *ebgp-multihop*.

11.5.3 Peer filtering

neighbor PEER distribute-list NAME [in|out]
This command specifies a distribute-list for the peer. *direct* is `in` or `out`.

neighbor PEER prefix-list NAME [in|out]

neighbor PEER filter-list NAME [in|out]

neighbor PEER route-map NAME [in|out]
Apply a route-map on the neighbor. *direct* must be *in* or *out*.

bgp route-reflector allow-outbound-policy
By default, attribute modification via route-map policy out is not reflected on reflected routes. This option allows the modifications to be reflected as well. Once enabled, it affects all reflected routes.

11.6 BGP Peer Group

neighbor WORD peer-group
This command defines a new peer group.

neighbor PEER peer-group WORD
This command bind specific peer to peer group WORD.

11.7 BGP Address Family

Multiprotocol BGP enables BGP to carry routing information for multiple Network Layer protocols. BGP supports multiple Address Family Identifier (AFI), namely IPv4 and IPv6. Support is also provided for multiple sets of per-AFI information via Subsequent Address Family Identifiers (SAFI). In addition to unicast information, VPN information **RFC 4364** and **RFC 4659**, and Encapsulation attribute **RFC 5512** is supported.

show ip bgp ipv4 vpn

show ipv6 bgp ipv6 vpn
Print active IPV4 or IPV6 routes advertised via the VPN SAFI.

show bgp ipv4 vpn summary

show bgp ipv6 vpn summary
Print a summary of neighbor connections for the specified AFI/SAFI combination.

11.8 Autonomous System

The AS (Autonomous System) number is one of the essential element of BGP. BGP is a distance vector routing protocol, and the AS-Path framework provides distance vector metric and loop detection to BGP. **RFC 1930** provides some background on the concepts of an AS.

The AS number is a two octet value, ranging in value from 1 to 65535. The AS numbers 64512 through 65535 are defined as private AS numbers. Private AS numbers must not to be advertised in the global Internet.

11.8.1 Display BGP Routes by AS Path

To show BGP routes which has specific AS path information *show ip bgp* command can be used.

show bgp ipv4|ipv6 regexp LINE
This commands displays BGP routes that matches a regular expression *line* (*BGP Regular Expressions*).

11.8.2 AS Path Access List

AS path access list is user defined AS path.

ip as-path access-list WORD permit|deny LINE
This command defines a new AS path access list.

no ip as-path access-list WORD

no ip as-path access-list WORD permit|deny LINE

11.8.3 Using AS Path in Route Map

match as-path WORD

set as-path prepend AS-PATH
Prepend the given string of AS numbers to the AS_PATH.

set as-path prepend last-as NUM
Prepend the existing last AS number (the leftmost ASN) to the AS_PATH.

11.8.4 Private AS Numbers

11.9 BGP Communities Attribute

BGP communities attribute is widely used for implementing policy routing. Network operators can manipulate BGP communities attribute based on their network policy. BGP communities attribute is defined in **RFC 1997** and **RFC 1998**. It is an optional transitive attribute, therefore local policy can travel through different autonomous system.

Communities attribute is a set of communities values. Each communities value is 4 octet long. The following format is used to define communities value.

AS:VAL This format represents 4 octet communities value. `AS` is high order 2 octet in digit format. `VAL` is low order 2 octet in digit format. This format is useful to define AS oriented policy value. For example, `7675:80` can be used when AS 7675 wants to pass local policy value 80 to neighboring peer.

internet *internet* represents well-known communities value 0.

no-export `no-export` represents well-known communities value `NO_EXPORT 0xFFFFFF01`. All routes carry this value must not be advertised to outside a BGP confederation boundary. If neighboring BGP peer is part of BGP confederation, the peer is considered as inside a BGP confederation boundary, so the route will be announced to the peer.

no-advertise `no-advertise` represents well-known communities value `NO_ADVERTISE 0xFFFFFF02`. All routes carry this value must not be advertise to other BGP peers.

local-AS `local-AS` represents well-known communities value `NO_EXPORT_SUBCONFED 0xFFFFFF03`. All routes carry this value must not be advertised to external BGP peers. Even if the neighboring router is part of confederation, it is considered as external BGP peer, so the route will not be announced to the peer.

When BGP communities attribute is received, duplicated communities value in the communities attribute is ignored and each communities values are sorted in numerical order.

11.9.1 BGP Community Lists

BGP community list is a user defined BGP communities attribute list. BGP community list can be used for matching or manipulating BGP communities attribute in updates.

There are two types of community list. One is standard community list and another is expanded community list. Standard community list defines communities attribute. Expanded community list defines communities attribute string with regular expression. Standard community list is compiled into binary format when user define it. Standard community list will be directly compared to BGP communities attribute in BGP updates. Therefore the comparison is faster than expanded community list.

ip community-list standard NAME permit|deny COMMUNITY
This command defines a new standard community list. COMUNITY is communities value. The COMUNITY is compiled into community structure. We can define multiple community list under same name. In that case match will happen user defined order. Once the community list matches to communities attribute in BGP updates it return permit or deny by the community list definition. When there is no matched entry, deny will be returned. When COMUNITY is empty it matches to any routes.

ip community-list expanded NAME permit|deny LINE
This command defines a new expanded community list. COMUNITY is a string expression of communities attribute. COMUNITY can be a regular expression (*BGP Regular Expressions*) to match the communities attribute in BGP updates.

no ip community-list NAME

no ip community-list standard NAME

no ip community-list expanded NAME
These commands delete community lists specified by NAME. All of community lists shares a single name space. So community lists can be removed simply specifying community lists name.

show ip community-list

show ip community-list NAME
This command displays current community list information. When NAME is specified the specified community list's information is shown.

```
# show ip community-list
Named Community standard list CLIST
permit 7675:80 7675:100 no-export
deny internet
  Named Community expanded list EXPAND
permit :
```

(continues on next page)

(continued from previous page)

```
  # show ip community-list CLIST
  Named Community standard list CLIST
permit 7675:80 7675:100 no-export
deny internet
```

11.9.2 Numbered BGP Community Lists

When number is used for BGP community list name, the number has special meanings. Community list number in the range from 1 and 99 is standard community list. Community list number in the range from 100 to 199 is expanded community list. These community lists are called as numbered community lists. On the other hand normal community lists is called as named community lists.

ip community-list (1-99) permit|deny COMMUNITY
: This command defines a new community list. (1-99) is standard community list number. Community list name within this range defines standard community list. When *community* is empty it matches to any routes.

ip community-list (100-199) permit|deny COMMUNITY
: This command defines a new community list. (100-199) is expanded community list number. Community list name within this range defines expanded community list.

ip community-list NAME permit|deny COMMUNITY
: When community list type is not specified, the community list type is automatically detected. If COMMUNITY can be compiled into communities attribute, the community list is defined as a standard community list. Otherwise it is defined as an expanded community list. This feature is left for backward compatibility. Use of this feature is not recommended.

11.9.3 BGP Community in Route Map

In Route Map (*Route Maps*), we can match or set BGP communities attribute. Using this feature network operator can implement their network policy based on BGP communities attribute.

Following commands can be used in Route Map.

match community WORD

match community WORD exact-match
: This command perform match to BGP updates using community list WORD. When the one of BGP communities value match to the one of communities value in community list, it is match. When *exact-match* keyword is specified, match happen only when BGP updates have completely same communities value specified in the community list.

set community none

set community COMMUNITY

set community COMMUNITY additive
: This command manipulate communities value in BGP updates. When *none* is specified as communities value, it removes entire communities attribute from BGP updates. When *community* is not *none*, specified communities value is set to BGP updates. If BGP updates already has BGP communities value, the existing BGP communities value is replaced with specified *community* value. When *additive* keyword is specified, *community* is appended to the existing communities value.

set comm-list WORD delete
: This command remove communities value from BGP communities attribute. The *word* is community list name. When BGP route's communities value matches to the community list *word*, the communities value is removed.

When all of communities value is removed eventually, the BGP update's communities attribute is completely removed.

11.9.4 Display BGP Routes by Community

To show BGP routes which has specific BGP communities attribute, *show bgp {ipv4|ipv6}* command can be used. The *community* and *community-list* subcommand can be used.

show bgp ipv4|ipv6 community

show bgp ipv4|ipv6 community COMMUNITY

show bgp ipv4|ipv6 community COMMUNITY exact-match
show bgp {ipv4|ipv6} community displays BGP routes which has communities attribute. Where the address family can be IPv4 or IPv6 among others. When *community* is specified, BGP routes that matches *community* value is displayed. For this command, *internet* keyword can't be used for *community* value. When *exact-match* is specified, it display only routes that have an exact match.

show bgp ipv4|ipv6 community-list WORD

show bgp ipv4|ipv6 community-list WORD exact-match
This commands display BGP routes for the address family specified that matches community list *word*. When *exact-match* is specified, display only routes that have an exact match.

11.9.5 Using BGP Communities Attribute

Following configuration is the most typical usage of BGP communities attribute. AS 7675 provides upstream Internet connection to AS 100. When following configuration exists in AS 7675, AS 100 networks operator can set local preference in AS 7675 network by setting BGP communities attribute to the updates.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
ip community-list 70 permit 7675:70
ip community-list 70 deny
ip community-list 80 permit 7675:80
ip community-list 80 deny
ip community-list 90 permit 7675:90
ip community-list 90 deny
!
route-map RMAP permit 10
 match community 70
 set local-preference 70
!
route-map RMAP permit 20
 match community 80
 set local-preference 80
!
route-map RMAP permit 30
 match community 90
 set local-preference 90
```

Following configuration announce 10.0.0.0/8 from AS 100 to AS 7675. The route has communities value 7675:80 so when above configuration exists in AS 7675, announced route's local preference will be set to value 80.

```
router bgp 100
 network 10.0.0.0/8
 neighbor 192.168.0.2 remote-as 7675
 address-family ipv4 unicast
  neighbor 192.168.0.2 route-map RMAP out
 exit-address-family
!
ip prefix-list PLIST permit 10.0.0.0/8
!
route-map RMAP permit 10
 match ip address prefix-list PLIST
 set community 7675:80
```

Following configuration is an example of BGP route filtering using communities attribute. This configuration only permit BGP routes which has BGP communities value 0:80 or 0:90. Network operator can put special internal communities value at BGP border router, then limit the BGP routes announcement into the internal network.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
ip community-list 1 permit 0:80 0:90
!
route-map RMAP permit in
 match community 1
```

Following example filter BGP routes which has communities value 1:1. When there is no match community-list returns deny. To avoid filtering all of routes, we need to define permit any at last.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
ip community-list standard FILTER deny 1:1
ip community-list standard FILTER permit
!
route-map RMAP permit 10
 match community FILTER
```

Communities value keyword *internet* has special meanings in standard community lists. In below example *internet* act as match any. It matches all of BGP routes even if the route does not have communities attribute at all. So community list `INTERNET` is same as above example's `FILTER`.

```
ip community-list standard INTERNET deny 1:1
ip community-list standard INTERNET permit internet
```

Following configuration is an example of communities value deletion. With this configuration communities value 100:1 and 100:2 is removed from BGP updates. For communities value deletion, only *permit* community-list is used. *deny* community-list is ignored.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
```

(continues on next page)

(continued from previous page)

```
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
ip community-list standard DEL permit 100:1 100:2
!
route-map RMAP permit 10
 set comm-list DEL delete
```

11.10 BGP Extended Communities Attribute

BGP extended communities attribute is introduced with MPLS VPN/BGP technology. MPLS VPN/BGP expands capability of network infrastructure to provide VPN functionality. At the same time it requires a new framework for policy routing. With BGP Extended Communities Attribute we can use Route Target or Site of Origin for implementing network policy for MPLS VPN/BGP.

BGP Extended Communities Attribute is similar to BGP Communities Attribute. It is an optional transitive attribute. BGP Extended Communities Attribute can carry multiple Extended Community value. Each Extended Community value is eight octet length.

BGP Extended Communities Attribute provides an extended range compared with BGP Communities Attribute. Adding to that there is a type field in each value to provides community space structure.

There are two format to define Extended Community value. One is AS based format the other is IP address based format.

AS:VAL This is a format to define AS based Extended Community value. *AS* part is 2 octets Global Administrator subfield in Extended Community value. *VAL* part is 4 octets Local Administrator subfield. *7675:100* represents AS 7675 policy value 100.

IP-Address:VAL This is a format to define IP address based Extended Community value. *IP-Address* part is 4 octets Global Administrator subfield. *VAL* part is 2 octets Local Administrator subfield. *10.0.0.1:100* represents

11.10.1 BGP Extended Community Lists

Expanded Community Lists is a user defined BGP Expanded Community Lists.

ip extcommunity-list standard NAME permit|deny EXTCOMMUNITY
This command defines a new standard extcommunity-list. *extcommunity* is extended communities value. The *extcommunity* is compiled into extended community structure. We can define multiple extcommunity-list under same name. In that case match will happen user defined order. Once the extcommunity-list matches to extended communities attribute in BGP updates it return permit or deny based upon the extcommunity-list definition. When there is no matched entry, deny will be returned. When *extcommunity* is empty it matches to any routes.

ip extcommunity-list expanded NAME permit|deny LINE
This command defines a new expanded extcommunity-list. *line* is a string expression of extended communities attribute. *line* can be a regular expression (*BGP Regular Expressions*) to match an extended communities attribute in BGP updates.

no ip extcommunity-list NAME

no ip extcommunity-list standard NAME

no ip extcommunity-list expanded NAME
These commands delete extended community lists specified by *name*. All of extended community lists shares a single name space. So extended community lists can be removed simply specifying the name.

show ip extcommunity-list

show ip extcommunity-list NAME
This command displays current extcommunity-list information. When *name* is specified the community list's information is shown.:

```
# show ip extcommunity-list
```

11.10.2 BGP Extended Communities in Route Map

match extcommunity WORD

set extcommunity rt EXTCOMMUNITY
This command set Route Target value.

set extcommunity soo EXTCOMMUNITY
This command set Site of Origin value.

11.11 BGP Large Communities Attribute

The BGP Large Communities attribute was introduced in Feb 2017 with **RFC 8092**.

The BGP Large Communities Attribute is similar to the BGP Communities Attribute except that it has 3 components instead of two and each of which are 4 octets in length. Large Communities bring additional functionality and convenience over traditional communities, specifically the fact that the *GLOBAL* part below is now 4 octets wide allowing AS4 operators seamless use.

GLOBAL:LOCAL1:LOCAL2 This is the format to define Large Community values. Referencing *RFC8195, Use of BGP Large Communities* the values are commonly referred to as follows. The *GLOBAL* part is a 4 octet Global Administrator field, common use of this field is the operators AS number. The *LOCAL1* part is a 4 octet Local Data Part 1 subfield referred to as a function. The *LOCAL2* part is a 4 octet Local Data Part 2 field and referred to as the parameter subfield. *65551:1:10* represents AS 65551 function 1 and parameter 10. The referenced RFC above gives some guidelines on recommended usage.

11.11.1 BGP Large Community Lists

Two types of large community lists are supported, namely *standard* and *expanded*.

ip large-community-list standard NAME permit|deny LARGE-COMMUNITY
This command defines a new standard large-community-list. *large-community* is the Large Community value. We can add multiple large communities under same name. In that case the match will happen in the user defined order. Once the large-community-list matches the Large Communities attribute in BGP updates it will return permit or deny based upon the large-community-list definition. When there is no matched entry, a deny will be returned. When *large-community* is empty it matches any routes.

ip large-community-list expanded NAME permit|deny LINE
This command defines a new expanded large-community-list. Where *line* is a string matching expression, it will be compared to the entire Large Communities attribute as a string, with each large-community in order from lowest to highest. *line* can also be a regular expression which matches this Large Community attribute.

no ip large-community-list NAME

no ip large-community-list standard NAME

no ip large-community-list expanded NAME
These commands delete Large Community lists specified by *name*. All Large Community lists share a single namespace. This means Large Community lists can be removed by simply specifying the name.

show ip large-community-list

show ip large-community-list NAME
This command display current large-community-list information. When *name* is specified the community list information is shown.

show ip bgp large-community-info
This command displays the current large communities in use.

11.11.2 BGP Large Communities in Route Map

match large-community LINE
Where *line* can be a simple string to match, or a regular expression. It is very important to note that this match occurs on the entire large-community string as a whole, where each large-community is ordered from lowest to highest.

set large-community LARGE-COMMUNITY

set large-community LARGE-COMMUNITY LARGE-COMMUNITY

set large-community LARGE-COMMUNITY additive
These commands are used for setting large-community values. The first command will overwrite any large-communities currently present. The second specifies two large-communities, which overwrites the current large-community list. The third will add a large-community value without overwriting other values. Multiple large-community values can be specified.

11.12 BGP VRFs

BPGD supports multiple VRF instances via the *router bgp* command:

router bgp ASN vrf VRFNAME

VRFNAME is matched against VRFs configured in the kernel. When no *vrf VRFNAME* is specified, the BGP protocol process belongs to the default VRF.

With VRF, you can isolate networking information. Having BGP VRF allows you to have several BGP instances on the same system process. This solution solves scalabiliy issues where the network administrator had previously to run separately several BGP processes on each namespace. Now, not only BGP VRF solves this, but also this method applies to both kind of VRFs backend: default VRF from Linux kernel or network namespaces. Also, having separate BGP instances does not imply that the AS number has to be different. For internal purposes, it is possible to do iBGP peering from two differents network namespaces.

BGP routes may be leaked (i.e., copied) between a unicast VRF RIB and the VPN safi RIB of the default VRF (leaking is also permitted between the unicast RIB of the default VRF and VPN). A shortcut syntax is also available for specifying leaking from one vrf to another vrf using the VPN RIB as the intemediary. A common application of the VPN-VRF feature is to connect a customer's private routing domain to a provider's VPN service. Leaking is configured from the point of view of an individual VRF: `import` refers to routes leaked from VPN to a unicast VRF, whereas `export` refers to routes leaked from a unicast VRF to VPN.

11.12.1 Required Parameters

Routes exported from a unicast VRF to the VPN RIB must be augmented by two parameters:

- an RD (Route Distinguisher)
- an RTLIST (Route-target List)

Configuration for these exported routes must, at a minimum, specify these two parameters.

Routes imported from the VPN RIB to a unicast VRF are selected according to their RTLISTs. Routes whose RTLIST contains at least one route-target in common with the configured import RTLIST are leaked. Configuration for these imported routes must specify an RTLIST to be matched.

The RD, which carries no semantic value, is intended to make the route unique in the VPN RIB among all routes of its prefix that originate from all the customers and sites that are attached to the provider's VPN service. Accordingly, each site of each customer is typically assigned an RD that is unique across the entire provider network.

The RTLIST is a set of route-target extended community values whose purpose is to specify route-leaking policy. Typically, a customer is assigned a single route-target value for import and export to be used at all customer sites. This configuration specifies a simple topology wherein a customer has a single routing domain which is shared across all its sites. More complex routing topologies are possible through use of additional route-targets to augment the leaking of sets of routes in various ways.

When using the shortcut syntax for vrf-to-vrf leaking, the RD and RT are auto-derived.

11.12.2 Configuration

Configuration of route leaking between a unicast VRF RIB and the VPN safi RIB of the default VRF is accomplished via commands in the context of a VRF address-family:

rd vpn export AS:NN|IP:nn
Specifies the route distinguisher to be added to a route exported from the current unicast VRF to VPN.

no rd vpn export [AS:NN|IP:nn]
Deletes any previously-configured export route distinguisher.

rt vpn import|export|both RTLIST...
Specifies the route-target list to be attached to a route (export) or the route-target list to match against (import) when exporting/importing between the current unicast VRF and VPN.

The RTLIST is a space-separated list of route-targets, which are BGP extended community values as described in *BGP Extended Communities Attribute*.

no rt vpn import|export|both [RTLIST...]
Deletes any previously-configured import or export route-target list.

label vpn export (0..1048575)|auto
Specifies an optional MPLS label to be attached to a route exported from the current unicast VRF to VPN. If label is specified as `auto`, the label value is automatically assigned from a pool maintained by the zebra daemon. If zebra is not running, automatic label assignment will not complete, which will block corresponding route export.

no label vpn export [(0..1048575)|auto]
Deletes any previously-configured export label.

nexthop vpn export A.B.C.D|X:X::X:X
Specifies an optional nexthop value to be assigned to a route exported from the current unicast VRF to VPN. If left unspecified, the nexthop will be set to 0.0.0.0 or 0:0::0:0 (self).

no nexthop vpn export [A.B.C.D|X:X::X:X]
Deletes any previously-configured export nexthop.

route-map vpn import|export MAP
Specifies an optional route-map to be applied to routes imported or exported between the current unicast VRF and VPN.

no route-map vpn import|export [MAP]
Deletes any previously-configured import or export route-map.

import|export vpn
Enables import or export of routes between the current unicast VRF and VPN.

no import|export vpn
Disables import or export of routes between the current unicast VRF and VPN.

import vrf VRFNAME
Shortcut syntax for specifying automatic leaking from vrf VRFNAME to the current VRF using the VPN RIB as intermediary. The RD and RT are auto derived and should not be specified explicitly for either the source or destination VRF's.

This shortcut syntax mode is not compatible with the explicit *import vpn* and *export vpn* statements for the two VRF's involved. The CLI will disallow attempts to configure incompatible leaking modes.

no import vrf VRFNAME
Disables automatic leaking from vrf VRFNAME to the current VRF using the VPN RIB as intermediary.

11.13 Displaying BGP information

11.13.1 Showing BGP information

show ip bgp

show ip bgp A.B.C.D

show ip bgp X:X::X:X
This command displays BGP routes. When no route is specified it display all of IPv4 BGP routes.

```
BGP table version is 0, local router ID is 10.1.1.1
   Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
   Origin codes: i - IGP, e - EGP, ? - incomplete

Network    Next Hop      Metric LocPrf Weight Path
   \*> 1.1.1.1/32       0.0.0.0      0   32768 i

   Total number of prefixes 1
```

show ip bgp regexp LINE
This command displays BGP routes using AS path regular expression (*BGP Regular Expressions*).

show ip bgp community COMMUNITY

show ip bgp community COMMUNITY exact-match
This command displays BGP routes using *community* (*Display BGP Routes by Community*).

show ip bgp community-list WORD

show ip bgp community-list WORD exact-match
This command displays BGP routes using community list (*Display BGP Routes by Community*).

show bgp ipv4|ipv6 summary
Show a bgp peer summary for the specified address family.

show bgp ipv4|ipv6 neighbor [PEER]
This command shows information on a specific BGP *peer*.

show bgp ipv4|ipv6 dampening dampened-paths
Display paths suppressed due to dampening.

show bgp ipv4|ipv6 dampening flap-statistics
Display flap statistics of routes.

11.13.2 Other BGP commands

clear bgp ipv4|ipv6 *
Clear all address family peers.

clear bgp ipv4|ipv6 PEER
Clear peers which have addresses of X.X.X.X

clear bgp ipv4|ipv6 PEER soft in
Clear peer using soft reconfiguration.

show debug

debug event

debug update

debug keepalive

no debug event

no debug update

no debug keepalive

11.14 Capability Negotiation

When adding IPv6 routing information exchange feature to BGP. There were some proposals. IETF (Internet Engineering Task Force) IDR (Inter Domain Routing) adopted a proposal called Multiprotocol Extension for BGP. The specification is described in **RFC 2283**. The protocol does not define new protocols. It defines new attributes to existing BGP. When it is used exchanging IPv6 routing information it is called BGP-4+. When it is used for exchanging multicast routing information it is called MBGP.

bgpd supports Multiprotocol Extension for BGP. So if a remote peer supports the protocol, *bgpd* can exchange IPv6 and/or multicast routing information.

Traditional BGP did not have the feature to detect a remote peer's capabilities, e.g. whether it can handle prefix types other than IPv4 unicast routes. This was a big problem using Multiprotocol Extension for BGP in an operational network. **RFC 2842** adopted a feature called Capability Negotiation. *bgpd* use this Capability Negotiation to detect the remote peer's capabilities. If a peer is only configured as an IPv4 unicast neighbor, *bgpd* does not send these Capability Negotiation packets (at least not unless other optional BGP features require capability negotiation).

By default, FRR will bring up peering with minimal common capability for the both sides. For example, if the local router has unicast and multicast capabilities and the remote router only has unicast capability the local router will establish the connection with unicast only capability. When there are no common capabilities, FRR sends Unsupported Capability error and then resets the connection.

If you want to completely match capabilities with remote peer. Please use *strict-capability-match* command.

neighbor PEER strict-capability-match

no neighbor PEER strict-capability-match
Strictly compares remote capabilities and local capabilities. If capabilities are different, send Unsupported Capability error then reset connection.

You may want to disable sending Capability Negotiation OPEN message optional parameter to the peer when remote peer does not implement Capability Negotiation. Please use *dont-capability-negotiate* command to disable the feature.

neighbor PEER dont-capability-negotiate

no neighbor PEER dont-capability-negotiate
Suppress sending Capability Negotiation as OPEN message optional parameter to the peer. This command only affects the peer is configured other than IPv4 unicast configuration.

When remote peer does not have capability negotiation feature, remote peer will not send any capabilities at all. In that case, bgp configures the peer with configured capabilities.

You may prefer locally configured capabilities more than the negotiated capabilities even though remote peer sends capabilities. If the peer is configured by *override-capability*, *bgpd* ignores received capabilities then override negotiated capabilities with configured values.

neighbor PEER override-capability

no neighbor PEER override-capability
Override the result of Capability Negotiation with local configuration. Ignore remote peer's capability value.

11.15 Route Reflector

bgp cluster-id A.B.C.D

neighbor PEER route-reflector-client

no neighbor PEER route-reflector-client

11.16 Route Server

At an Internet Exchange point, many ISPs are connected to each other by the "full mesh method". As with internal BGP full mesh formation, this method has a scaling problem.

This scaling problem is well known. Route Server is a method to resolve the problem. Each ISP's BGP router only peers to Route Server. Route Server serves as BGP information exchange to other BGP routers. By applying this method, numbers of BGP connections is reduced from O(n*(n-1)/2) to O(n).

Unlike a normal BGP router, Route Server must have several routing tables for managing different routing policies for each BGP speaker. We call the routing tables as different "views". *bgpd* can work as normal BGP router or Route Server or both at the same time.

11.16.1 Multiple instance

To enable multiple view function of *bgpd*, you must turn on multiple instance feature beforehand.

bgp multiple-instance
Enable BGP multiple instance feature. After this feature is enabled, you can make multiple BGP instances or multiple BGP views.

no bgp multiple-instance
Disable BGP multiple instance feature. You can not disable this feature when BGP multiple instances or views exist.

When you want to make configuration more Cisco like one,

bgp config-type cisco
Cisco compatible BGP configuration output.

When bgp config-type cisco is specified,

`no synchronization` is displayed. `no auto-summary` is displayed.

The `network` and `aggregate-address` arguments are displayed as:

```
A.B.C.D M.M.M.M

FRR: network 10.0.0.0/8
Cisco: network 10.0.0.0

FRR: aggregate-address 192.168.0.0/24
Cisco: aggregate-address 192.168.0.0 255.255.255.0
```

Community attribute handling is also different. If no configuration is specified community attribute and extended community attribute are sent to the neighbor. If a user manually disables the feature, the community attribute is not sent to the neighbor. When `bgp config-type cisco` is specified, the community attribute is not sent to the neighbor by default. To send the community attribute user has to specify *neighbor A.B.C.D send-community* command.

```
!
router bgp 1
 neighbor 10.0.0.1 remote-as 1
 address-family ipv4 unicast
  no neighbor 10.0.0.1 send-community
 exit-address-family
!
router bgp 1
 neighbor 10.0.0.1 remote-as 1
 address-family ipv4 unicast
  neighbor 10.0.0.1 send-community
 exit-address-family
!
```

bgp config-type zebra
FRR style BGP configuration. This is default.

11.16.2 BGP instance and view

BGP instance is a normal BGP process. The result of route selection goes to the kernel routing table. You can setup different AS at the same time when BGP multiple instance feature is enabled.

router bgp AS-NUMBER
Make a new BGP instance. You can use an arbitrary word for the *name*.

```
bgp multiple-instance
!
router bgp 1
 neighbor 10.0.0.1 remote-as 2
 neighbor 10.0.0.2 remote-as 3
```

(continues on next page)

(continued from previous page)

```
!
router bgp 2
 neighbor 10.0.0.3 remote-as 4
 neighbor 10.0.0.4 remote-as 5
```

BGP view is almost same as normal BGP process. The result of route selection does not go to the kernel routing table. BGP view is only for exchanging BGP routing information.

router bgp AS-NUMBER view NAME
Make a new BGP view. You can use arbitrary word for the *name*. This view's route selection result does not go to the kernel routing table.

With this command, you can setup Route Server like below.

```
bgp multiple-instance
!
router bgp 1 view 1
 neighbor 10.0.0.1 remote-as 2
 neighbor 10.0.0.2 remote-as 3
!
router bgp 2 view 2
 neighbor 10.0.0.3 remote-as 4
 neighbor 10.0.0.4 remote-as 5
```

11.16.3 Routing policy

You can set different routing policy for a peer. For example, you can set different filter for a peer.

```
bgp multiple-instance
!
router bgp 1 view 1
 neighbor 10.0.0.1 remote-as 2
 address-family ipv4 unicast
  neighbor 10.0.0.1 distribute-list 1 in
 exit-address-family
!
router bgp 1 view 2
 neighbor 10.0.0.1 remote-as 2
 address-family ipv4 unicast
  neighbor 10.0.0.1 distribute-list 2 in
 exit-address-family
```

This means BGP update from a peer 10.0.0.1 goes to both BGP view 1 and view 2. When the update is inserted into view 1, distribute-list 1 is applied. On the other hand, when the update is inserted into view 2, distribute-list 2 is applied.

11.16.4 Viewing the view

To display routing table of BGP view, you must specify view name.

show ip bgp view NAME
Display routing table of BGP view `NAME`.

11.17 BGP Regular Expressions

BGP regular expressions are based on *POSIX 1003.2* regular expressions. The following description is just a quick subset of the *POSIX* regular expressions. Adding to that, the special character '_' is added.

.* Matches any single character.

- Matches 0 or more occurrences of pattern.
- Matches 1 or more occurrences of pattern.

? Match 0 or 1 occurrences of pattern.

^ Matches the beginning of the line.

$ Matches the end of the line.

_ Character _ has special meanings in BGP regular expressions. It matches to space and comma , and AS set delimiter { and } and AS confederation delimiter *(* and *)*. And it also matches to the beginning of the line and the end of the line. So _ can be used for AS value boundaries match. This character technically evaluates to *(^|[,{}()]|$)*.

11.18 How to set up a 6-Bone connection

```
! bgpd configuration
! ==================
!
! MP-BGP configuration
!
router bgp 7675
 bgp router-id 10.0.0.1
 neighbor 3ffe:1cfa:0:2:2a0:c9ff:fe9e:f56 remote-as `as-number`
!
 address-family ipv6
 network 3ffe:506::/32
 neighbor 3ffe:1cfa:0:2:2a0:c9ff:fe9e:f56 activate
 neighbor 3ffe:1cfa:0:2:2a0:c9ff:fe9e:f56 route-map set-nexthop out
 neighbor 3ffe:1cfa:0:2:2c0:4fff:fe68:a231 remote-as `as-number`
 neighbor 3ffe:1cfa:0:2:2c0:4fff:fe68:a231 route-map set-nexthop out
 exit-address-family
!
ipv6 access-list all permit any
!
! Set output nexthop address.
!
route-map set-nexthop permit 10
 match ipv6 address all
 set ipv6 nexthop global 3ffe:1cfa:0:2:2c0:4fff:fe68:a225
 set ipv6 nexthop local fe80::2c0:4fff:fe68:a225
!
log file bgpd.log
!
```

11.19 Dump BGP packets and table

dump bgp all PATH [INTERVAL]

dump bgp all-et PATH [INTERVAL]

no dump bgp all [PATH] [INTERVAL]
Dump all BGP packet and events to *path* file. If *interval* is set, a new file will be created for echo *interval* of seconds. The path *path* can be set with date and time formatting (strftime). The type 'all-et' enables support for Extended Timestamp Header (*Packet Binary Dump Format*).

dump bgp updates PATH [INTERVAL]

dump bgp updates-et PATH [INTERVAL]

no dump bgp updates [PATH] [INTERVAL]
Dump only BGP updates messages to *path* file. If *interval* is set, a new file will be created for echo *interval* of seconds. The path *path* can be set with date and time formatting (strftime). The type 'updates-et' enables support for Extended Timestamp Header (*Packet Binary Dump Format*).

dump bgp routes-mrt PATH

dump bgp routes-mrt PATH INTERVAL

no dump bgp route-mrt [PATH] [INTERVAL]
Dump whole BGP routing table to *path*. This is heavy process. The path *path* can be set with date and time formatting (strftime). If *interval* is set, a new file will be created for echo *interval* of seconds.

Note: the interval variable can also be set using hours and minutes: 04h20m00.

11.20 BGP Configuration Examples

Example of a session to an upstream, advertising only one prefix to it.

```
router bgp 64512
 bgp router-id 10.236.87.1
 neighbor upstream peer-group
 neighbor upstream remote-as 64515
 neighbor upstream capability dynamic
 neighbor 10.1.1.1 peer-group upstream
 neighbor 10.1.1.1 description ACME ISP

 address-family ipv4 unicast
  network 10.236.87.0/24
  neighbor upstream prefix-list pl-allowed-adv out
 exit-address-family
!
ip prefix-list pl-allowed-adv seq 5 permit 82.195.133.0/25
ip prefix-list pl-allowed-adv seq 10 deny any
```

A more complex example. With upstream, peer and customer sessions. Advertising global prefixes and NO_EXPORT prefixes and providing actions for customer routes based on community values. Extensive use of route-maps and the 'call' feature to support selective advertising of prefixes. This example is intended as guidance only, it has NOT been tested and almost certainly contains silly mistakes, if not serious flaws.

```
router bgp 64512
 bgp router-id 10.236.87.1
 neighbor upstream capability dynamic
 neighbor cust capability dynamic
 neighbor peer capability dynamic
 neighbor 10.1.1.1 remote-as 64515
 neighbor 10.1.1.1 peer-group upstream
```

(continues on next page)

(continued from previous page)

```
 neighbor 10.2.1.1 remote-as 64516
 neighbor 10.2.1.1 peer-group upstream
 neighbor 10.3.1.1 remote-as 64517
 neighbor 10.3.1.1 peer-group cust-default
 neighbor 10.3.1.1 description customer1
 neighbor 10.4.1.1 remote-as 64518
 neighbor 10.4.1.1 peer-group cust
 neighbor 10.4.1.1 description customer2
 neighbor 10.5.1.1 remote-as 64519
 neighbor 10.5.1.1 peer-group peer
 neighbor 10.5.1.1 description peer AS 1
 neighbor 10.6.1.1 remote-as 64520
 neighbor 10.6.1.1 peer-group peer
 neighbor 10.6.1.1 description peer AS 2

 address-family ipv4 unicast
  network 10.123.456.0/24
  network 10.123.456.128/25 route-map rm-no-export
  neighbor upstream route-map rm-upstream-out out
  neighbor cust route-map rm-cust-in in
  neighbor cust route-map rm-cust-out out
  neighbor cust send-community both
  neighbor peer route-map rm-peer-in in
  neighbor peer route-map rm-peer-out out
  neighbor peer send-community both
  neighbor 10.3.1.1 prefix-list pl-cust1-network in
  neighbor 10.4.1.1 prefix-list pl-cust2-network in
  neighbor 10.5.1.1 prefix-list pl-peer1-network in
  neighbor 10.6.1.1 prefix-list pl-peer2-network in
 exit-address-family
!
ip prefix-list pl-default permit 0.0.0.0/0
!
ip prefix-list pl-upstream-peers permit 10.1.1.1/32
ip prefix-list pl-upstream-peers permit 10.2.1.1/32
!
ip prefix-list pl-cust1-network permit 10.3.1.0/24
ip prefix-list pl-cust1-network permit 10.3.2.0/24
!
ip prefix-list pl-cust2-network permit 10.4.1.0/24
!
ip prefix-list pl-peer1-network permit 10.5.1.0/24
ip prefix-list pl-peer1-network permit 10.5.2.0/24
ip prefix-list pl-peer1-network permit 192.168.0.0/24
!
ip prefix-list pl-peer2-network permit 10.6.1.0/24
ip prefix-list pl-peer2-network permit 10.6.2.0/24
ip prefix-list pl-peer2-network permit 192.168.1.0/24
ip prefix-list pl-peer2-network permit 192.168.2.0/24
ip prefix-list pl-peer2-network permit 172.16.1/24
!
ip as-path access-list asp-own-as permit ^$
ip as-path access-list asp-own-as permit _64512_
!
! ###################################################################
! Match communities we provide actions for, on routes receives from
! customers. Communities values of <our-ASN>:X, with X, have actions:
```

(continues on next page)

(continued from previous page)

```
!
! 100 - blackhole the prefix
! 200 - set no_export
! 300 - advertise only to other customers
! 400 - advertise only to upstreams
! 500 - set no_export when advertising to upstreams
! 2X00 - set local_preference to X00
!
! blackhole the prefix of the route
ip community-list standard cm-blackhole permit 64512:100
!
! set no-export community before advertising
ip community-list standard cm-set-no-export permit 64512:200
!
! advertise only to other customers
ip community-list standard cm-cust-only permit 64512:300
!
! advertise only to upstreams
ip community-list standard cm-upstream-only permit 64512:400
!
! advertise to upstreams with no-export
ip community-list standard cm-upstream-noexport permit 64512:500
!
! set local-pref to least significant 3 digits of the community
ip community-list standard cm-prefmod-100 permit 64512:2100
ip community-list standard cm-prefmod-200 permit 64512:2200
ip community-list standard cm-prefmod-300 permit 64512:2300
ip community-list standard cm-prefmod-400 permit 64512:2400
ip community-list expanded cme-prefmod-range permit 64512:2...
!
! Informational communities
!
! 3000 - learned from upstream
! 3100 - learned from customer
! 3200 - learned from peer
!
ip community-list standard cm-learnt-upstream permit 64512:3000
ip community-list standard cm-learnt-cust permit 64512:3100
ip community-list standard cm-learnt-peer permit 64512:3200
!
! ###################################################################
! Utility route-maps
!
! These utility route-maps generally should not used to permit/deny
! routes, i.e. they do not have meaning as filters, and hence probably
! should be used with 'on-match next'. These all finish with an empty
! permit entry so as not interfere with processing in the caller.
!
route-map rm-no-export permit 10
 set community additive no-export
route-map rm-no-export permit 20
!
route-map rm-blackhole permit 10
 description blackhole, up-pref and ensure it cant escape this AS
 set ip next-hop 127.0.0.1
 set local-preference 10
 set community additive no-export
```

(continues on next page)

(continued from previous page)

```
route-map rm-blackhole permit 20
!
! Set local-pref as requested
route-map rm-prefmod permit 10
 match community cm-prefmod-100
 set local-preference 100
route-map rm-prefmod permit 20
 match community cm-prefmod-200
 set local-preference 200
route-map rm-prefmod permit 30
 match community cm-prefmod-300
 set local-preference 300
route-map rm-prefmod permit 40
 match community cm-prefmod-400
 set local-preference 400
route-map rm-prefmod permit 50
!
! Community actions to take on receipt of route.
route-map rm-community-in permit 10
 description check for blackholing, no point continuing if it matches.
 match community cm-blackhole
 call rm-blackhole
route-map rm-community-in permit 20
 match community cm-set-no-export
 call rm-no-export
 on-match next
route-map rm-community-in permit 30
 match community cme-prefmod-range
 call rm-prefmod
route-map rm-community-in permit 40
!
! ###################################################################
! Community actions to take when advertising a route.
! These are filtering route-maps,
!
! Deny customer routes to upstream with cust-only set.
route-map rm-community-filt-to-upstream deny 10
 match community cm-learnt-cust
 match community cm-cust-only
route-map rm-community-filt-to-upstream permit 20
!
! Deny customer routes to other customers with upstream-only set.
route-map rm-community-filt-to-cust deny 10
 match community cm-learnt-cust
 match community cm-upstream-only
route-map rm-community-filt-to-cust permit 20
!
! ###################################################################
! The top-level route-maps applied to sessions. Further entries could
! be added obviously..
!
! Customers
route-map rm-cust-in permit 10
 call rm-community-in
 on-match next
route-map rm-cust-in permit 20
 set community additive 64512:3100
```

(continues on next page)

(continued from previous page)

```
route-map rm-cust-in permit 30
!
route-map rm-cust-out permit 10
 call rm-community-filt-to-cust
 on-match next
route-map rm-cust-out permit 20
!
! Upstream transit ASes
route-map rm-upstream-out permit 10
 description filter customer prefixes which are marked cust-only
 call rm-community-filt-to-upstream
 on-match next
route-map rm-upstream-out permit 20
 description only customer routes are provided to upstreams/peers
 match community cm-learnt-cust
!
! Peer ASes
! outbound policy is same as for upstream
route-map rm-peer-out permit 10
 call rm-upstream-out
!
route-map rm-peer-in permit 10
 set community additive 64512:3200
```

11.21 Configuring FRR as a Route Server

The purpose of a Route Server is to centralize the peerings between BGP speakers. For example if we have an exchange point scenario with four BGP speakers, each of which maintaining a BGP peering with the other three (*Full Mesh*), we can convert it into a centralized scenario where each of the four establishes a single BGP peering against the Route Server (*Route server and clients*).

We will first describe briefly the Route Server model implemented by FRR. We will explain the commands that have been added for configuring that model. And finally we will show a full example of FRR configured as Route Server.

11.21.1 Description of the Route Server model

First we are going to describe the normal processing that BGP announcements suffer inside a standard BGP speaker, as shown in *Announcement processing inside a 'normal' BGP speaker*, it consists of three steps:

- When an announcement is received from some peer, the *In* filters configured for that peer are applied to the announcement. These filters can reject the announcement, accept it unmodified, or accept it with some of its attributes modified.
- The announcements that pass the *In* filters go into the Best Path Selection process, where they are compared to other announcements referred to the same destination that have been received from different peers (in case such other announcements exist). For each different destination, the announcement which is selected as the best is inserted into the BGP speaker's Loc-RIB.
- The routes which are inserted in the Loc-RIB are considered for announcement to all the peers (except the one from which the route came). This is done by passing the routes in the Loc-RIB through the *Out* filters corresponding to each peer. These filters can reject the route, accept it unmodified, or accept it with some of its attributes modified. Those routes which are accepted by the *Out* filters of a peer are announced to that peer.

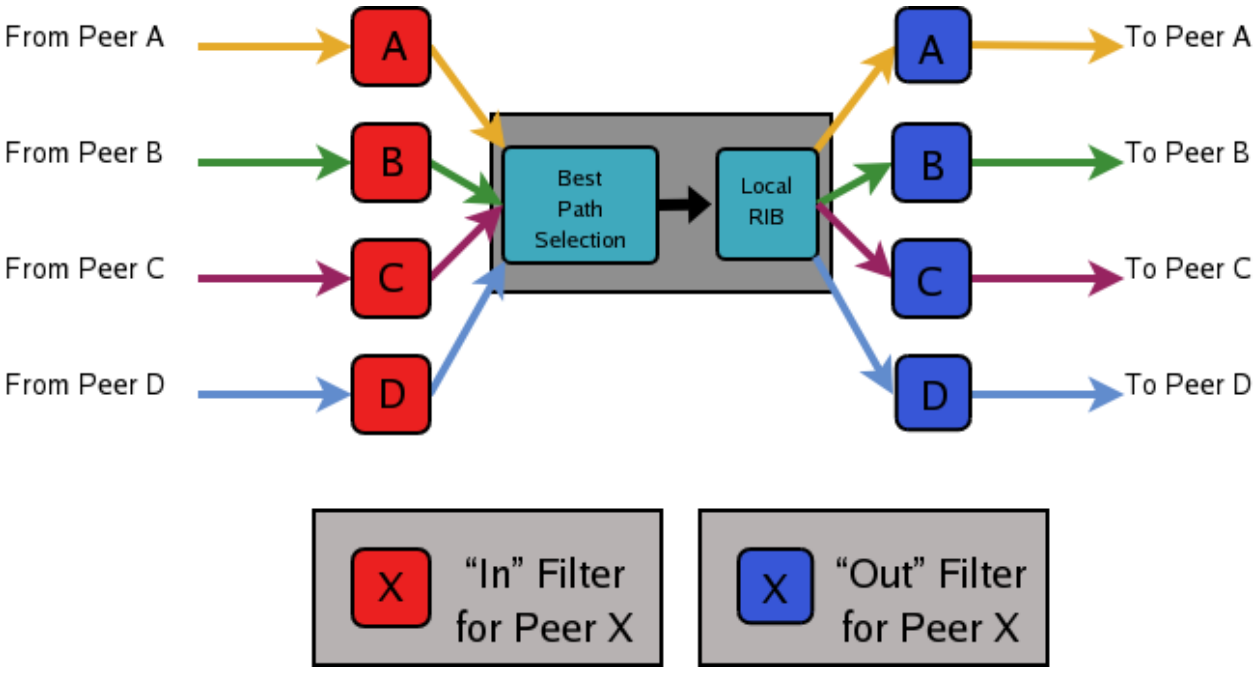


Fig. 1: Announcement processing inside a 'normal' BGP speaker

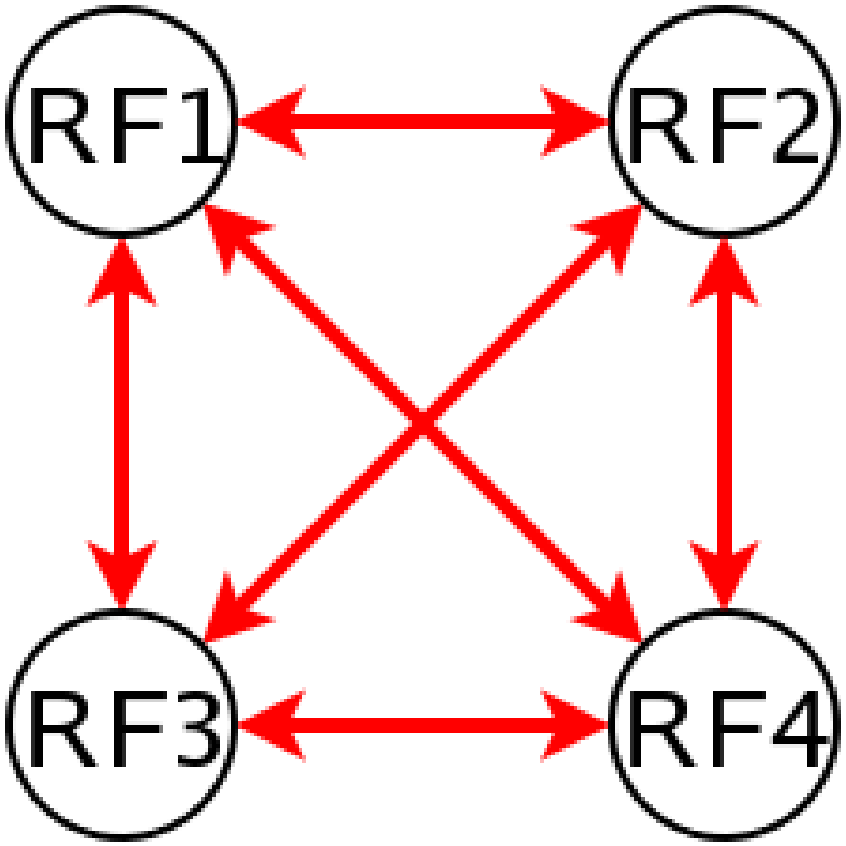


Fig. 2: Full Mesh

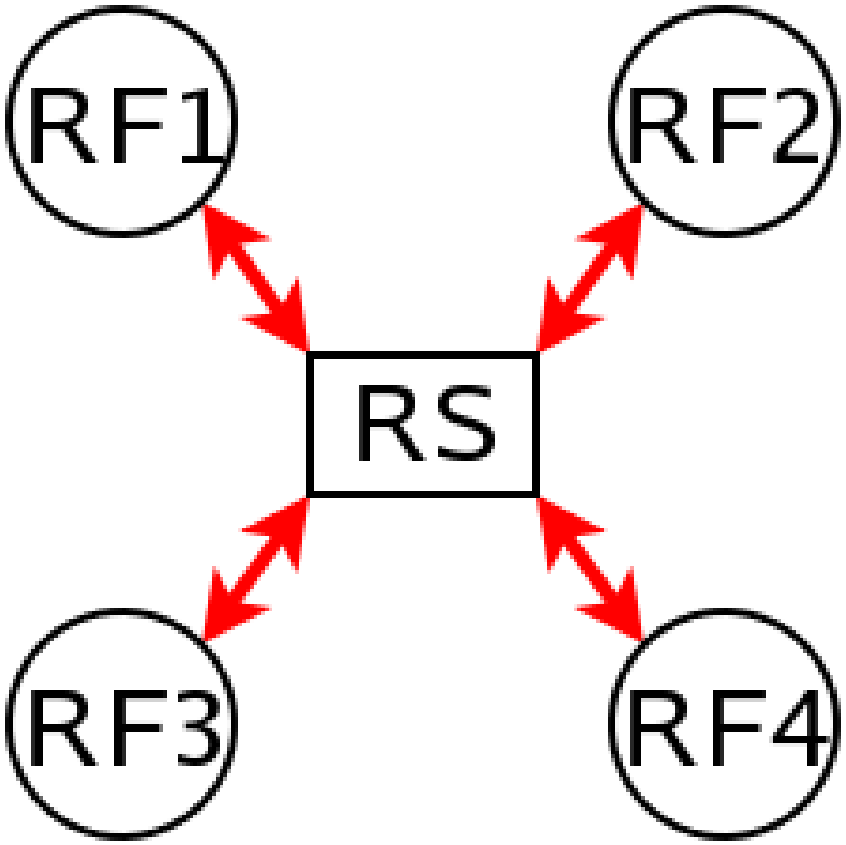


Fig. 3: Route server and clients

Of course we want that the routing tables obtained in each of the routers are the same when using the route server than when not. But as a consequence of having a single BGP peering (against the route server), the BGP speakers can no longer distinguish from/to which peer each announce comes/goes.

This means that the routers connected to the route server are not able to apply by themselves the same input/output filters as in the full mesh scenario, so they have to delegate those functions to the route server.

Even more, the 'best path' selection must be also performed inside the route server on behalf of its clients. The reason is that if, after applying the filters of the announcer and the (potential) receiver, the route server decides to send to some client two or more different announcements referred to the same destination, the client will only retain the last one, considering it as an implicit withdrawal of the previous announcements for the same destination. This is the expected behavior of a BGP speaker as defined in **RFC 1771**, and even though there are some proposals of mechanisms that permit multiple paths for the same destination to be sent through a single BGP peering, none are currently supported by most existing BGP implementations.

As a consequence a route server must maintain additional information and perform additional tasks for a RS-client that those necessary for common BGP peerings. Essentially a route server must:

- Maintain a separated Routing Information Base (Loc-RIB) for each peer configured as RS-client, containing the routes selected as a result of the 'Best Path Selection' process that is performed on behalf of that RS-client.
- Whenever it receives an announcement from a RS-client, it must consider it for the Loc-RIBs of the other RS-clients.
  - This means that for each of them the route server must pass the announcement through the appropriate *Out* filter of the announcer.
  - Then through the appropriate *In* filter of the potential receiver.
  - Only if the announcement is accepted by both filters it will be passed to the 'Best Path Selection' process.
  - Finally, it might go into the Loc-RIB of the receiver.

When we talk about the 'appropriate' filter, both the announcer and the receiver of the route must be taken into account. Suppose that the route server receives an announcement from client A, and the route server is considering it for the Loc-RIB of client B. The filters that should be applied are the same that would be used in the full mesh scenario, i.e., first the *Out* filter of router A for announcements going to router B, and then the *In* filter of router B for announcements coming from router A.

We call 'Export Policy' of a RS-client to the set of *Out* filters that the client would use if there was no route server. The same applies for the 'Import Policy' of a RS-client and the set of *In* filters of the client if there was no route server.

It is also common to demand from a route server that it does not modify some BGP attributes (next-hop, as-path and MED) that are usually modified by standard BGP speakers before announcing a route.

The announcement processing model implemented by FRR is shown in *Announcement processing model implemented by the Route Server*. The figure shows a mixture of RS-clients (B, C and D) with normal BGP peers (A). There are some details that worth additional comments:

- Announcements coming from a normal BGP peer are also considered for the Loc-RIBs of all the RS-clients. But logically they do not pass through any export policy.
- Those peers that are configured as RS-clients do not receive any announce from the *Main* Loc-RIB.
- Apart from import and export policies, *In* and *Out* filters can also be set for RS-clients. *In* filters might be useful when the route server has also normal BGP peers. On the other hand, *Out* filters for RS-clients are probably unnecessary, but we decided not to remove them as they do not hurt anybody (they can always be left empty).

11.21.2 Commands for configuring a Route Server

Now we will describe the commands that have been added to frr in order to support the route server features.

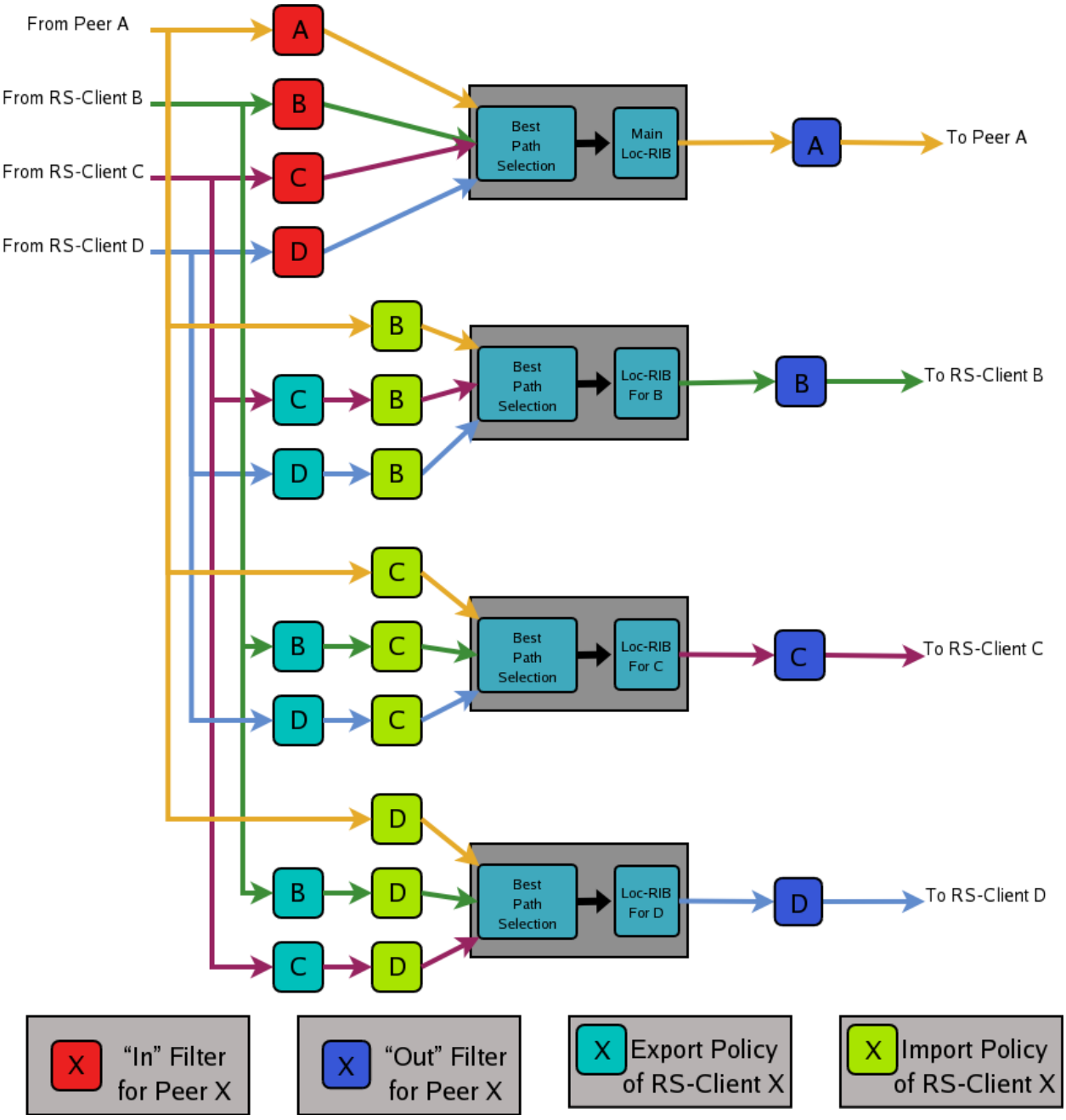


Fig. 4: Announcement processing model implemented by the Route Server

neighbor PEER-GROUP route-server-client

neighbor A.B.C.D route-server-client

neighbor X:X::X:X route-server-client
This command configures the peer given by *peer*, *A.B.C.D* or *X:X::X:X* as an RS-client.

Actually this command is not new, it already existed in standard FRR. It enables the transparent mode for the specified peer. This means that some BGP attributes (as-path, next-hop and MED) of the routes announced to that peer are not modified.

With the route server patch, this command, apart from setting the transparent mode, creates a new Loc-RIB dedicated to the specified peer (those named *Loc-RIB for X* in *Announcement processing model implemented by the Route Server*.). Starting from that moment, every announcement received by the route server will be also considered for the new Loc-RIB.

neigbor A.B.C.D|X.X::X.X|peer-group route-map WORD import|export
This set of commands can be used to specify the route-map that represents the Import or Export policy of a peer which is configured as a RS-client (with the previous command).

match peer A.B.C.D|X:X::X:X
This is a new *match* statement for use in route-maps, enabling them to describe import/export policies. As we said before, an import/export policy represents a set of input/output filters of the RS-client. This statement makes possible that a single route-map represents the full set of filters that a BGP speaker would use for its different peers in a non-RS scenario.

The *match peer* statement has different semantics whether it is used inside an import or an export route-map. In the first case the statement matches if the address of the peer who sends the announce is the same that the address specified by {A.B.C.D|X:X::X:X}. For export route-maps it matches when {A.B.C.D|X:X::X:X} is the address of the RS-Client into whose Loc-RIB the announce is going to be inserted (how the same export policy is applied before different Loc-RIBs is shown in *Announcement processing model implemented by the Route Server*.).

call WORD
This command (also used inside a route-map) jumps into a different route-map, whose name is specified by *WORD*. When the called route-map finishes, depending on its result the original route-map continues or not. Apart from being useful for making import/export route-maps easier to write, this command can also be used inside any normal (in or out) route-map.

11.21.3 Example of Route Server Configuration

Finally we are going to show how to configure a FRR daemon to act as a Route Server. For this purpose we are going to present a scenario without route server, and then we will show how to use the configurations of the BGP routers to generate the configuration of the route server.

All the configuration files shown in this section have been taken from scenarios which were tested using the VNUML tool http://www.dit.upm.es/vnuml,VNUML.

11.21.4 Configuration of the BGP routers without Route Server

We will suppose that our initial scenario is an exchange point with three BGP capable routers, named RA, RB and RC. Each of the BGP speakers generates some routes (with the *network* command), and establishes BGP peerings against the other two routers. These peerings have In and Out route-maps configured, named like 'PEER-X-IN' or 'PEER-X-OUT'. For example the configuration file for router RA could be the following:

```
#Configuration for router 'RA'
!
hostname RA
password ****
!
router bgp 65001
  no bgp default ipv4-unicast
  neighbor 2001:0DB8::B remote-as 65002
  neighbor 2001:0DB8::C remote-as 65003
!
  address-family ipv6
    network 2001:0DB8:AAAA:1::/64
    network 2001:0DB8:AAAA:2::/64
    network 2001:0DB8:0000:1::/64
    network 2001:0DB8:0000:2::/64
    neighbor 2001:0DB8::B activate
    neighbor 2001:0DB8::B soft-reconfiguration inbound
    neighbor 2001:0DB8::B route-map PEER-B-IN in
    neighbor 2001:0DB8::B route-map PEER-B-OUT out
    neighbor 2001:0DB8::C activate
    neighbor 2001:0DB8::C soft-reconfiguration inbound
    neighbor 2001:0DB8::C route-map PEER-C-IN in
    neighbor 2001:0DB8::C route-map PEER-C-OUT out
  exit-address-family
!
ipv6 prefix-list COMMON-PREFIXES seq  5 permit 2001:0DB8:0000::/48 ge 64 le 64
ipv6 prefix-list COMMON-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-A-PREFIXES seq  5 permit 2001:0DB8:AAAA::/48 ge 64 le 64
ipv6 prefix-list PEER-A-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-B-PREFIXES seq  5 permit 2001:0DB8:BBBB::/48 ge 64 le 64
ipv6 prefix-list PEER-B-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-C-PREFIXES seq  5 permit 2001:0DB8:CCCC::/48 ge 64 le 64
ipv6 prefix-list PEER-C-PREFIXES seq 10 deny any
!
route-map PEER-B-IN permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set metric 100
route-map PEER-B-IN permit 20
  match ipv6 address prefix-list PEER-B-PREFIXES
  set community 65001:11111
!
route-map PEER-C-IN permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set metric 200
route-map PEER-C-IN permit 20
  match ipv6 address prefix-list PEER-C-PREFIXES
  set community 65001:22222
!
route-map PEER-B-OUT permit 10
  match ipv6 address prefix-list PEER-A-PREFIXES
!
route-map PEER-C-OUT permit 10
  match ipv6 address prefix-list PEER-A-PREFIXES
!
```

(continues on next page)

(continued from previous page)

```
line vty
!
```

11.21.5 Configuration of the BGP routers with Route Server

To convert the initial scenario into one with route server, first we must modify the configuration of routers RA, RB and RC. Now they must not peer between them, but only with the route server. For example, RA's configuration would turn into:

```
# Configuration for router 'RA'
!
hostname RA
password ****
!
router bgp 65001
  no bgp default ipv4-unicast
  neighbor 2001:0DB8::FFFF remote-as 65000
!
  address-family ipv6
    network 2001:0DB8:AAAA:1::/64
    network 2001:0DB8:AAAA:2::/64
    network 2001:0DB8:0000:1::/64
    network 2001:0DB8:0000:2::/64

    neighbor 2001:0DB8::FFFF activate
    neighbor 2001:0DB8::FFFF soft-reconfiguration inbound
  exit-address-family
!
line vty
!
```

Which is logically much simpler than its initial configuration, as it now maintains only one BGP peering and all the filters (route-maps) have disappeared.

11.21.6 Configuration of the Route Server itself

As we said when we described the functions of a route server (*Description of the Route Server model*), it is in charge of all the route filtering. To achieve that, the In and Out filters from the RA, RB and RC configurations must be converted into Import and Export policies in the route server.

This is a fragment of the route server configuration (we only show the policies for client RA):

```
# Configuration for Route Server ('RS')
!
hostname RS
password ix
!
bgp multiple-instance
!
router bgp 65000 view RS
  no bgp default ipv4-unicast
  neighbor 2001:0DB8::A  remote-as 65001
  neighbor 2001:0DB8::B  remote-as 65002
```

(continues on next page)

(continued from previous page)

```
  neighbor 2001:0DB8::C  remote-as 65003
!
  address-family ipv6
    neighbor 2001:0DB8::A activate
    neighbor 2001:0DB8::A route-server-client
    neighbor 2001:0DB8::A route-map RSCLIENT-A-IMPORT import
    neighbor 2001:0DB8::A route-map RSCLIENT-A-EXPORT export
    neighbor 2001:0DB8::A soft-reconfiguration inbound

    neighbor 2001:0DB8::B activate
    neighbor 2001:0DB8::B route-server-client
    neighbor 2001:0DB8::B route-map RSCLIENT-B-IMPORT import
    neighbor 2001:0DB8::B route-map RSCLIENT-B-EXPORT export
    neighbor 2001:0DB8::B soft-reconfiguration inbound

    neighbor 2001:0DB8::C activate
    neighbor 2001:0DB8::C route-server-client
    neighbor 2001:0DB8::C route-map RSCLIENT-C-IMPORT import
    neighbor 2001:0DB8::C route-map RSCLIENT-C-EXPORT export
    neighbor 2001:0DB8::C soft-reconfiguration inbound
  exit-address-family
!
ipv6 prefix-list COMMON-PREFIXES seq  5 permit 2001:0DB8:0000::/48 ge 64 le 64
ipv6 prefix-list COMMON-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-A-PREFIXES seq  5 permit 2001:0DB8:AAAA::/48 ge 64 le 64
ipv6 prefix-list PEER-A-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-B-PREFIXES seq  5 permit 2001:0DB8:BBBB::/48 ge 64 le 64
ipv6 prefix-list PEER-B-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-C-PREFIXES seq  5 permit 2001:0DB8:CCCC::/48 ge 64 le 64
ipv6 prefix-list PEER-C-PREFIXES seq 10 deny any
!
route-map RSCLIENT-A-IMPORT permit 10
  match peer 2001:0DB8::B
  call A-IMPORT-FROM-B
route-map RSCLIENT-A-IMPORT permit 20
  match peer 2001:0DB8::C
  call A-IMPORT-FROM-C
!
route-map A-IMPORT-FROM-B permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set metric 100
route-map A-IMPORT-FROM-B permit 20
  match ipv6 address prefix-list PEER-B-PREFIXES
  set community 65001:11111
!
route-map A-IMPORT-FROM-C permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set metric 200
route-map A-IMPORT-FROM-C permit 20
  match ipv6 address prefix-list PEER-C-PREFIXES
  set community 65001:22222
!
route-map RSCLIENT-A-EXPORT permit 10
  match peer 2001:0DB8::B
```

(continues on next page)

(continued from previous page)

```
  match ipv6 address prefix-list PEER-A-PREFIXES
route-map RSCLIENT-A-EXPORT permit 20
  match peer 2001:0DB8::C
  match ipv6 address prefix-list PEER-A-PREFIXES
!
...
...
...
```

If you compare the initial configuration of RA with the route server configuration above, you can see how easy it is to generate the Import and Export policies for RA from the In and Out route-maps of RA's original configuration.

When there was no route server, RA maintained two peerings, one with RB and another with RC. Each of this peerings had an In route-map configured. To build the Import route-map for client RA in the route server, simply add route-map entries following this scheme:

```
route-map <NAME> permit 10
    match peer <Peer Address>
    call <In Route-Map for this Peer>
route-map <NAME> permit 20
    match peer <Another Peer Address>
    call <In Route-Map for this Peer>
```

This is exactly the process that has been followed to generate the route-map RSCLIENT-A-IMPORT. The route-maps that are called inside it (A-IMPORT-FROM-B and A-IMPORT-FROM-C) are exactly the same than the In route-maps from the original configuration of RA (PEER-B-IN and PEER-C-IN), only the name is different.

The same could have been done to create the Export policy for RA (route-map RSCLIENT-A-EXPORT), but in this case the original Out route-maps where so simple that we decided not to use the *call WORD* commands, and we integrated all in a single route-map (RSCLIENT-A-EXPORT).

The Import and Export policies for RB and RC are not shown, but the process would be identical.

11.21.7 Further considerations about Import and Export route-maps

The current version of the route server patch only allows to specify a route-map for import and export policies, while in a standard BGP speaker apart from route-maps there are other tools for performing input and output filtering (access-lists, community-lists, ...). But this does not represent any limitation, as all kinds of filters can be included in import/export route-maps. For example suppose that in the non-route-server scenario peer RA had the following filters configured for input from peer B:

```
neighbor 2001:0DB8::B prefix-list LIST-1 in
neighbor 2001:0DB8::B filter-list LIST-2 in
neighbor 2001:0DB8::B route-map PEER-B-IN in
...
...
route-map PEER-B-IN permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set local-preference 100
route-map PEER-B-IN permit 20
  match ipv6 address prefix-list PEER-B-PREFIXES
  set community 65001:11111
```

It is possible to write a single route-map which is equivalent to the three filters (the community-list, the prefix-list and the route-map). That route-map can then be used inside the Import policy in the route server. Lets see how to do it:

```
neighbor 2001:0DB8::A route-map RSCLIENT-A-IMPORT import
...
!
...
route-map RSCLIENT-A-IMPORT permit 10
  match peer 2001:0DB8::B
  call A-IMPORT-FROM-B
...
...
!
route-map A-IMPORT-FROM-B permit 1
  match ipv6 address prefix-list LIST-1
  match as-path LIST-2
  on-match goto 10
route-map A-IMPORT-FROM-B deny 2
route-map A-IMPORT-FROM-B permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set local-preference 100
route-map A-IMPORT-FROM-B permit 20
  match ipv6 address prefix-list PEER-B-PREFIXES
  set community 65001:11111
!
...
...
```

The route-map A-IMPORT-FROM-B is equivalent to the three filters (LIST-1, LIST-2 and PEER-B-IN). The first entry of route-map A-IMPORT-FROM-B (sequence number 1) matches if and only if both the prefix-list LIST-1 and the filter-list LIST-2 match. If that happens, due to the 'on-match goto 10' statement the next route-map entry to be processed will be number 10, and as of that point route-map A-IMPORT-FROM-B is identical to PEER-B-IN. If the first entry does not match, *on-match goto 10*' will be ignored and the next processed entry will be number 2, which will deny the route.

Thus, the result is the same that with the three original filters, i.e., if either LIST-1 or LIST-2 rejects the route, it does not reach the route-map PEER-B-IN. In case both LIST-1 and LIST-2 accept the route, it passes to PEER-B-IN, which can reject, accept or modify the route.

11.22 Prefix Origin Validation Using RPKI

Prefix Origin Validation allows BGP routers to verify if the origin AS of an IP prefix is legitimate to announce this IP prefix. The required attestation objects are stored in the Resource Public Key Infrastructure (RPKI). However, RPKI-enabled routers do not store cryptographic data itself but only validation information. The validation of the cryptographic data (so called Route Origin Authorization, or short ROA, objects) will be performed by trusted cache servers. The RPKI/RTR protocol defines a standard mechanism to maintain the exchange of the prefix/origin AS mapping between the cache server and routers. In combination with a BGP Prefix Origin Validation scheme a router is able to verify received BGP updates without suffering from cryptographic complexity.

The RPKI/RTR protocol is defined in **RFC 6810** and the validation scheme in **RFC 6811**. The current version of Prefix Origin Validation in FRR implements both RFCs.

For a more detailed but still easy-to-read background, we suggest:

- [Securing-BGP]
- [Resource-Certification]

11.22.1 Features of the Current Implementation

In a nutshell, the current implementation provides the following features

- The BGP router can connect to one or more RPKI cache servers to receive validated prefix to origin AS mappings. Advanced failover can be implemented by server sockets with different preference values.
- If no connection to an RPKI cache server can be established after a pre-defined timeout, the router will process routes without prefix origin validation. It still will try to establish a connection to an RPKI cache server in the background.
- By default, enabling RPKI does not change best path selection. In particular, invalid prefixes will still be considered during best path selection. However, the router can be configured to ignore all invalid prefixes.
- Route maps can be configured to match a specific RPKI validation state. This allows the creation of local policies, which handle BGP routes based on the outcome of the Prefix Origin Validation.

11.22.2 Enabling RPKI

rpki
This command enables the RPKI configuration mode. Most commands that start with *rpki* can only be used in this mode.

When it is used in a telnet session, leaving of this mode cause rpki to be initialized.

Executing this command alone does not activate prefix validation. You need to configure at least one reachable cache server. See section *Configuring RPKI/RTR Cache Servers* for configuring a cache server.

11.22.3 Configuring RPKI/RTR Cache Servers

The following commands are independent of a specific cache server.

rpki polling_period (1-3600)

no rpki polling_period
Set the number of seconds the router waits until the router asks the cache again for updated data.

The default value is 300 seconds.

rpki timeout <1-4,294,967,296>

no rpki timeout
Set the number of seconds the router waits for the cache reply. If the cache server is not replying within this time period, the router deletes all received prefix records from the prefix table.

The default value is 600 seconds.

rpki initial-synchronisation-timeout <1-4,294,967,296>

no rpki initial-synchronisation-timeout
Set the number of seconds until the first synchronization with the cache server needs to be completed. If the timeout expires, BGP routing is started without RPKI. The router will try to establish the cache server connection in the background.

The default value is 30 seconds.

The following commands configure one or multiple cache servers.

rpki cache (A.B.C.D|WORD) PORT [SSH_USERNAME] [SSH_PRIVKEY_PATH] [SSH_PUBKEY_PATH] [KNOWN_

no rpki cache (A.B.C.D|WORD) [PORT] PREFERENCE
Add a cache server to the socket. By default, the connection between router and cache server is based on plain TCP. Protecting the connection between router and cache server by SSH is optional. Deleting a socket removes the associated cache server and terminates the existing connection.

A.B.C.D|WORD Address of the cache server.

PORT Port number to connect to the cache server

SSH_USERNAME SSH username to establish an SSH connection to the cache server.

SSH_PRIVKEY_PATH Local path that includes the private key file of the router.

SSH_PUBKEY_PATH Local path that includes the public key file of the router.

KNOWN_HOSTS_PATH Local path that includes the known hosts file. The default value depends on the configuration of the operating system environment, usually `~/.ssh/known_hosts`.

11.22.4 Validating BGP Updates

match rpki notfound|invalid|valid

no match rpki notfound|invalid|valid
Create a clause for a route map to match prefixes with the specified RPKI state.

Note that the matching of invalid prefixes requires that invalid prefixes are considered for best path selection, i.e., `bgp bestpath prefix-validate disallow-invalid` is not enabled.

In the following example, the router prefers valid routes over invalid prefixes because invalid routes have a lower local preference.

```
! Allow for invalid routes in route selection process
route bgp 60001
!
! Set local preference of invalid prefixes to 10
route-map rpki permit 10
 match rpki invalid
 set local-preference 10
!
! Set local preference of valid prefixes to 500
route-map rpki permit 500
 match rpki valid
 set local-preference 500
```

11.22.5 Debugging

debug rpki

no debug rpki
Enable or disable debugging output for RPKI.

11.22.6 Displaying RPKI

show rpki prefix-table
Display all validated prefix to origin AS mappings/records which have been received from the cache servers and stored in the router. Based on this data, the router validates BGP Updates.

show rpki cache-connection
Display all configured cache servers, whether active or not.

11.22.7 RPKI Configuration Example

```
hostname bgpd1
password zebra
! log stdout
debug bgp updates
debug bgp keepalives
debug rpki
!
rpki
 rpki polling_period 1000
 rpki timeout 10
  ! SSH Example:
  rpki cache example.com 22 rtr-ssh ./ssh_key/id_rsa ./ssh_key/id_rsa.pub preference 1
  ! TCP Example:
  rpki cache rpki-validator.realmv6.org 8282 preference 2
  exit
!
router bgp 60001
 bgp router-id 141.22.28.223
 network 192.168.0.0/16
 neighbor 123.123.123.0 remote-as 60002
 neighbor 123.123.123.0 route-map rpki in
!
 address-family ipv6
  neighbor 123.123.123.0 activate
   neighbor 123.123.123.0 route-map rpki in
 exit-address-family
!
route-map rpki permit 10
 match rpki invalid
 set local-preference 10
!
route-map rpki permit 20
 match rpki notfound
 set local-preference 20
!
route-map rpki permit 30
 match rpki valid
 set local-preference 30
!
route-map rpki permit 40
!
```

11.23 Flowspec

11.23.1 Overview

Flowspec introduces a new NLRI (Network Layer Reachability Information) encoding format that is used to distribute traffic rule flow specifications. Basically, instead of simply relying on destination IP address for IP prefixes, the IP

prefix is replaced by a n-tuple consisting of a rule. That rule can be a more or less complex combination of the following:

- Network source/destination (can be one or the other, or both).
- Layer 4 information for UDP/TCP: source port, destination port, or any port.
- Layer 4 information for ICMP type and ICMP code.
- Layer 4 information for TCP Flags.
- Layer 3 information: DSCP value, Protocol type, packet length, fragmentation.
- Misc layer 4 TCP flags.

A combination of the above rules is applied for traffic filtering. This is encoded as part of specific BGP extended communities and the action can range from the obvious rerouting (to nexthop or to separate VRF) to shaping, or discard.

The following IETF drafts and RFCs have been used to implement FRR Flowspec:

- **RFC 5575**
- [Draft-IETF-IDR-Flowspec-redirect-IP]

11.23.2 Design Principles

FRR implements the Flowspec client side, that is to say that BGP is able to receive Flowspec entries, but is not able to act as manager and send Flowspec entries.

Linux provides the following mechanisms to implement policy based routing:

- Filtering the traffic with `Netfilter`. `Netfilter` provides a set of tools like `ipset` and `iptables` that are powerful enough to be able to filter such Flowspec filter rule.
- using non standard routing tables via `iproute2` (via the `ip rule` command provided by `iproute2`). `iproute2` is already used by FRR's *PBR* daemon which provides basic policy based routing based on IP source and destination criterion.

Below example is an illustration of what Flowspec will inject in the underlying system:

```
# linux shell
ipset create match0x102 hash:net,net counters
ipset add match0x102 32.0.0.0/16,40.0.0.0/16
iptables -N match0x102 -t mangle
iptables -A match0x102 -t mangle -j MARK --set-mark 102
iptables -A match0x102 -t mangle -j ACCEPT
iptables -i ntfp3 -t mangle -I PREROUTING -m set --match-set match0x102
             src,dst -g match0x102
ip rule add fwmark 102 lookup 102
ip route add 40.0.0.0/16 via 44.0.0.2 table 102
```

For handling an incoming Flowspec entry, the following workflow is applied:

- Incoming Flowspec entries are handled by *bgpd*, stored in the BGP RIB.
- Flowspec entry is installed according to its complexity.

It will be installed if one of the following filtering action is seen on the BGP extended community: either redirect IP, or redirect VRF, in conjunction with rate option, for redirecting traffic. Or rate option set to 0, for discarding traffic.

According to the degree of complexity of the Flowspec entry, it will be installed in *zebra* RIB. For more information about what is supported in the FRR implementation as rule, see *Limitations / Known Issues* chapter. Flowspec entry is split in several parts before being sent to *zebra*.

- *zebra* daemon receives the policy routing configuration

Policy Based Routing entities necessary to policy route the traffic in the underlying system, are received by *zebra*. Two filtering contexts will be created or appended in `Netfilter`: `ipset` and `iptable` context. The former is used to define an IP filter based on multiple criterium. For instance, an ipset `net:net` is based on two ip addresses, while `net,port,net` is based on two ip addresses and one port (for ICMP, UDP, or TCP). The way the filtering is used (for example, is src port or dst port used?) is defined by the latter filtering context. `iptable` command will reference the `ipset` context and will tell how to filter and what to do. In our case, a marker will be set to indicate `iproute2` where to forward the traffic to. Sometimes, for dropping action, there is no need to add a marker; the `iptable` will tell to drop all packets matching the `ipset` entry.

11.23.3 Configuration Guide

In order to configure an IPv4 Flowspec engine, use the following configuration. As of today, it is only possible to configure Flowspec on the default VRF.

```
router bgp <AS>
  neighbor <A.B.C.D> remote-as <remoteAS>
  address-family ipv4 flowspec
   neighbor <A.B.C.D> activate
 exit
exit
```

You can see Flowspec entries, by using one of the following show commands:

`show bgp ipv4 flowspec [detail | A.B.C.D]`

Per-interface configuration

One nice feature to use is the ability to apply Flowspec to a specific interface, instead of applying it to the whole machine. Despite the following IETF draft [Draft-IETF-IDR-Flowspec-Interface-Set] is not implemented, it is possible to manually limit Flowspec application to some incoming interfaces. Actually, not using it can result to some unexpected behaviour like accounting twice the traffic, or slow down the traffic (filtering costs). To limit Flowspec to one specific interface, use the following command, under *flowspec address-family* node.

`[no] local-install <IFNAME | any>`

By default, Flowspec is activated on all interfaces. Installing it to a named interface will result in allowing only this interface. Conversely, enabling any interface will flush all previously configured interfaces.

VRF redirection

Another nice feature to configure is the ability to redirect traffic to a separate VRF. This feature does not go against the ability to configure Flowspec only on default VRF. Actually, when you receive incoming BGP flowspec entries on that default VRF, you can redirect traffic to an other VRF.

As a reminder, BGP flowspec entries have a BGP extended community that contains a Route Target. Finding out a local VRF based on Route Target consists in the following:

- A configuration of each VRF must be done, with its Route Target set Each VRF is being configured within a BGP VRF instance with its own Route Target list. Route Target accepted format matches the following: `A.B.C.D:U16`, or `U16:U32`, `U32:U16`.

- The first VRF with the matching Route Target will be selected to route traffic to. Use the following command under ipv4 unicast address-family node

[no] rt redirect import RTLIST...

In order to illustrate, if the Route Target configured in the Flowspec entry is `E.F.G.H:II`, then a BGP VRF instance with the same Route Target will be set set. That VRF will then be selected. The below full configuration example depicts how Route Targets are configured and how VRFs and cross VRF configuration is done. Note that the VRF are mapped on Linux Network Namespaces. For data traffic to cross VRF boundaries, virtual ethernet interfaces are created with private IP adressing scheme.

```
router bgp <ASx>
 neighbor <A.B.C.D> remote-as <ASz>
 address-family ipv4 flowspec
  neighbor A.B.C.D activate
 exit
exit
router bgp <ASy> vrf vrf2
 address-family ipv4 unicast
  rt redirect import <E.F.G.H:II>
 exit
exit
```

Flowspec monitoring & troubleshooting

You can monitor policy-routing objects by using one of the following commands. Those command rely on the filtering contexts configured from BGP, and get the statistics information retrieved from the underlying system. In other words, those statistics are retrieved from `Netfilter`.

show pbr ipset IPSETNAME | iptable

`IPSETNAME` is the policy routing object name created by `ipset`. About rule contexts, it is possible to know which rule has been configured to policy-route some specific traffic. The `show pbr iptable` command displays for forwarded traffic, which table is used. Then it is easy to use that table identifier to dump the routing table that the forwarded traffic will match.

```

```

show ip route table TABLEID
`TABLEID` is the table number identifier referencing the non standard routing table used in this example.

[no] debug bgp flowspec
You can troubleshoot Flowspec, or BGP policy based routing. For instance, if you encounter some issues when decoding a Flowspec entry, you should enable `debug bgp flowspec`.

[no] debug bgp pbr [error]
If you fail to apply the flowspec entry into *zebra*, there should be some relationship with policy routing mechanism. Here, `debug bgp pbr error` could help.

To get information about policy routing contexts created/removed, only use `debug bgp pbr` command.

Ensuring that a Flowspec entry has been correctly installed and that incoming traffic is policy-routed correctly can be checked as demonstrated below. First of all, you must check whether the Flowspec entry has been installed or not.

```
CLI# show bgp ipv4 flowspec 5.5.5.2/32
 BGP flowspec entry: (flags 0x418)
   Destination Address 5.5.5.2/32
   IP Protocol = 17
```

(continues on next page)

(continued from previous page)

```
   Destination Port >= 50 , <= 90
   FS:redirect VRF RT:255.255.255.255:255
   received for 18:41:37
   installed in PBR (match0x271ce00)
```

This means that the Flowspec entry has been installed in an `iptable` named `match0x271ce00`. Once you have confirmation it is installed, you can check whether you find the associate entry by executing following command. You can also check whether incoming traffic has been matched by looking at counter line.

```
CLI# show pbr ipset match0x271ce00
IPset match0x271ce00 type net,port
     to 5.5.5.0/24:proto 6:80-120 (8)
        pkts 1000, bytes 1000000
     to 5.5.5.2:proto 17:50-90 (5)
        pkts 1692918, bytes 157441374
```

As you can see, the entry is present. note that an `iptable` entry can be used to host several Flowspec entries. In order to know where the matching traffic is redirected to, you have to look at the policy routing rules. The policy-routing is done by forwarding traffic to a routing table number. That routing table number is reached by using a `iptable`. The relationship between the routing table number and the incoming traffic is a `MARKER` that is set by the IPtable referencing the IPSet. In Flowspec case, `iptable` referencing the `ipset` context have the same name. So it is easy to know which routing table is used by issuing following command:

```
CLI# show pbr iptable
   IPtable match0x271ce00 action redirect (5)
     pkts 1700000, bytes 158000000
     table 257, fwmark 257
...
```

As you can see, by using following Linux commands, the MARKER `0x101` is present in both `iptable` and `ip rule` contexts.

```
# iptables -t mangle --list match0x271ce00 -v
Chain match0x271ce00 (1 references)
pkts bytes target     prot opt in     out     source               destination
1700K  158M MARK       all  --  any    any     anywhere             anywhere
     MARK set 0x101
1700K  158M ACCEPT     all  --  any    any     anywhere             anywhere

# ip rule list
0:from all lookup local
0:from all fwmark 0x101 lookup 257
32766:from all lookup main
32767:from all lookup default
```

This allows us to see where the traffic is forwarded to.

11.23.4 Limitations / Known Issues

As you can see, Flowspec is rich and can be very complex. As of today, not all Flowspec rules will be able to be converted into Policy Based Routing actions.

- The `Netfilter` driver is not integrated into FRR yet. Not having this piece of code prevents from injecting flowspec entries into the underlying system.
- There are some limitations around filtering contexts

If I take example of UDP ports, or TCP ports in Flowspec, the information can be a range of ports, or a unique value. This case is handled. However, complexity can be increased, if the flow is a combination of a list of range of ports and an enumerate of unique values. Here this case is not handled. Similarly, it is not possible to create a filter for both src port and dst port. For instance, filter on src port from [1-1000] and dst port = 80. The same kind of complexity is not possible for packet length, ICMP type, ICMP code.

There are some other known issues:

- The validation procedure depicted in **RFC 5575** is not available.

  This validation procedure has not been implemented, as this feature was not used in the existing setups you shared wih us.

- The filtering action shaper value, if positive, is not used to apply shaping.

  If value is positive, the traffic is redirected to the wished destination, without any other action configured by Flowspec. It is recommended to configure Quality of Service if needed, more globally on a per interface basis.

- Upon an unexpected crash or other event, *zebra* may not have time to flush PBR contexts.

  That is to say `ipset`, `iptable` and `ip rule` contexts. This is also a consequence due to the fact that ip rule / ipset / iptables are not discovered at startup (not able to read appropriate contexts coming from Flowspec).

11.23.5 Appendix

More information with a public presentation that explains the design of Flowspec inside FRRouting.

[Presentation]

CHAPTER 12

Babel

Babel is an interior gateway protocol that is suitable both for wired networks and for wireless mesh networks. Babel has been described as 'RIP on speed' – it is based on the same principles as RIP, but includes a number of refinements that make it react much faster to topology changes without ever counting to infinity, and allow it to perform reliable link quality estimation on wireless links. Babel is a double-stack routing protocol, meaning that a single Babel instance is able to perform routing for both IPv4 and IPv6.

FRR implements Babel as described in **RFC 6126**.

12.1 Configuring babeld

The *babeld* daemon can be invoked with any of the common options (*Common Invocation Options*).

The *zebra* daemon must be running before *babeld* is invoked. Also, if *zebra* is restarted then *babeld* must be too.

Configuration of *babeld* is done in its configuration file `babeld.conf`.

12.2 Babel configuration

[no] router babel
Enable or disable Babel routing.

[no] babel resend-delay (20-655340)
Specifies the time after which important messages are resent when avoiding a black-hole. The default is 2000 ms.

[no] babel diversity
Enable or disable routing using radio frequency diversity. This is highly recommended in networks with many wireless nodes. If you enable this, you will probably want to set *babel diversity-factor* and *babel channel* below.

babel diversity-factor (1-256)
Sets the multiplicative factor used for diversity routing, in units of 1/256; lower values cause diversity to play

a more important role in route selection. The default it 256, which means that diversity plays no role in route selection; you will probably want to set that to 128 or less on nodes with multiple independent radios.

no network IFNAME
Enable or disable Babel on the given interface.

babel <wired|wireless>
Specifies whether this interface is wireless, which disables a number of optimisations that are only correct on wired interfaces. Specifying *wireless* (the default) is always correct, but may cause slower convergence and extra routing traffic.

[no] babel split-horizon
Specifies whether to perform split-horizon on the interface. Specifying `no babel split-horizon` is always correct, while `babel split-horizon` is an optimisation that should only be used on symmetric and transitive (wired) networks. The default is `babel split-horizon` on wired interfaces, and `no babel split-horizon` on wireless interfaces. This flag is reset when the wired/wireless status of an interface is changed.

babel hello-interval (20-655340)
Specifies the time in milliseconds between two scheduled hellos. On wired links, Babel notices a link failure within two hello intervals; on wireless links, the link quality value is reestimated at every hello interval. The default is 4000 ms.

babel update-interval (20-655340)
Specifies the time in milliseconds between two scheduled updates. Since Babel makes extensive use of triggered updates, this can be set to fairly high values on links with little packet loss. The default is 20000 ms.

babel channel (1-254)

babel channel interfering

babel channel noninterfering
Set the channel number that diversity routing uses for this interface (see *babel diversity* above). Noninterfering interfaces are assumed to only interfere with themselves, interfering interfaces are assumed to interfere with all other channels except noninterfering channels, and interfaces with a channel number interfere with interfering interfaces and interfaces with the same channel number. The default is `babel channel interfering` for wireless interfaces, and `babel channel noninterfering` for wired interfaces. This is reset when the wired/wireless status of an interface is changed.

babel rxcost (1-65534)
Specifies the base receive cost for this interface. For wireless interfaces, it specifies the multiplier used for computing the ETX reception cost (default 256); for wired interfaces, it specifies the cost that will be advertised to neighbours. This value is reset when the wired/wireless attribute of the interface is changed.

Note: Do not use this command unless you know what you are doing; in most networks, acting directly on the cost using route maps is a better technique.

---

babel rtt-decay (1-256)
This specifies the decay factor for the exponential moving average of RTT samples, in units of 1/256. Higher values discard old samples faster. The default is 42.

babel rtt-min (1-65535)
This specifies the minimum RTT, in milliseconds, starting from which we increase the cost to a neighbour. The additional cost is linear in (rtt - rtt-min). The default is 100 ms.

babel rtt-max (1-65535)
This specifies the maximum RTT, in milliseconds, above which we don't increase the cost to a neighbour. The default is 120 ms.

babel max-rtt-penalty (0-65535)
This specifies the maximum cost added to a neighbour because of RTT, i.e. when the RTT is higher or equal than rtt-max. The default is 0, which effectively disables the use of a RTT-based cost.

[no] babel enable-timestamps
Enable or disable sending timestamps with each Hello and IHU message in order to compute RTT values. The default is *no babel enable-timestamps*.

babel resend-delay (20-655340)
Specifies the time in milliseconds after which an 'important' request or update will be resent. The default is 2000 ms. You probably don't want to tweak this value.

babel smoothing-half-life (0-65534)
Specifies the time constant, in seconds, of the smoothing algorithm used for implementing hysteresis. Larger values reduce route oscillation at the cost of very slightly increasing convergence time. The value 0 disables hysteresis, and is suitable for wired networks. The default is 4 s.

12.3 Babel redistribution

[no] redistribute <ipv4|ipv6> KIND
Specify which kind of routes should be redistributed into Babel.

12.4 Show Babel information

These commands dump various parts of *babeld*'s internal state.

show babel route

show babel route A.B.C.D

show babel route X:X::X:X

show babel route A.B.C.D/M

show babel route X:X::X:X/M

show babel interface

show babel interface IFNAME

show babel neighbor

show babel parameters

12.5 Babel debugging commands

[no] debug babel KIND
Enable or disable debugging messages of a given kind. `KIND` can be one of:

- `common`
- `filter`
- `timeout`
- `interface`

- route
- all

Note: If you have compiled with the NO_DEBUG flag, then these commands aren't available.

CHAPTER 13

EIGRP

DUAL The *Diffusing Update ALgorithm*, a *Bellman-Ford* based routing algorithm used by EIGRP.

EIGRP – Routing Information Protocol is widely deployed interior gateway routing protocol. EIGRP was developed in the 1990's. EIGRP is a *distance-vector* protocol and is based on the *DUAL* algorithms. As a distance-vector protocol, the EIGRP router send updates to its neighbors as networks change, thus allowing the convergence to a known topology.

eigrpd supports EIGRP as described in RFC7868

13.1 Starting and Stopping eigrpd

The default configuration file name of *eigrpd*'s is `eigrpd.conf`. When invocation *eigrpd* searches directory /etc/frr. If `eigrpd.conf` is not there next search current directory. If an integrated config is specified configuration is written into `frr.conf`.

The EIGRP protocol requires interface information maintained by *zebra* daemon. So running *zebra* is mandatory to run *eigrpd*. Thus minimum sequence for running EIGRP is:

```
# zebra -d
# eigrpd -d
```

Please note that *zebra* must be invoked before *eigrpd*.

To stop *eigrpd*, please use :: kill *cat /var/run/eigrpd.pid*

Certain signals have special meanings to *eigrpd*.

Signal	Meaning
SIGHUP & SIGUSR1	Rotate the log file
SIGINT & SIGTERM	Sweep all installed EIGRP routes and gracefully terminate

eigrpd invocation options. Common options that can be specified (*Common Invocation Options*).

-r, --retain
When the program terminates, retain routes added by *eigrpd*.

13.2 EIGRP Configuration

router eigrp (1-65535)
The *router eigrp* command is necessary to enable EIGRP. To disable EIGRP, use the *no router eigrp (1-65535)* command. EIGRP must be enabled before carrying out any of the EIGRP commands.

no router eigrp (1-65535)
Disable EIGRP.

network NETWORK

no network NETWORK
Set the EIGRP enable interface by *network*. The interfaces which have addresses matching with *network* are enabled.

This group of commands either enables or disables EIGRP interfaces between certain numbers of a specified network address. For example, if the network for 10.0.0.0/24 is EIGRP enabled, this would result in all the addresses from 10.0.0.0 to 10.0.0.255 being enabled for EIGRP. The *no network* command will disable EIGRP for the specified network.

Below is very simple EIGRP configuration. Interface *eth0* and interface which address match to *10.0.0.0/8* are EIGRP enabled.

```
!
router eigrp 1
 network 10.0.0.0/8
!
```

passive-interface (IFNAME|default)

no passive-interface IFNAME
This command sets the specified interface to passive mode. On passive mode interface, all receiving packets are ignored and eigrpd does not send either multicast or unicast EIGRP packets except to EIGRP neighbors specified with *neighbor* command. The interface may be specified as *default* to make eigrpd default to passive on all interfaces.

The default is to be passive on all interfaces.

13.3 How to Announce EIGRP route

redistribute kernel

redistribute kernel metric (1-4294967295) (0-4294967295) (0-255) (1-255) (1-65535)

no redistribute kernel
redistribute kernel redistributes routing information from kernel route entries into the EIGRP tables. *no redistribute kernel* disables the routes.

redistribute static

redistribute static metric (1-4294967295) (0-4294967295) (0-255) (1-255) (1-65535)

no redistribute static
redistribute static redistributes routing information from static route entries into the EIGRP tables. *no redistribute static* disables the routes.

redistribute connected

redistribute connected metric (1-4294967295) (0-4294967295) (0-255) (1-255) (1-65535)

no redistribute connected
Redistribute connected routes into the EIGRP tables. *no redistribute connected* disables the connected routes in the EIGRP tables. This command redistribute connected of the interface which EIGRP disabled. The connected route on EIGRP enabled interface is announced by default.

redistribute ospf

redistribute ospf metric (1-4294967295) (0-4294967295) (0-255) (1-255) (1-65535)

no redistribute ospf
redistribute ospf redistributes routing information from ospf route entries into the EIGRP tables. *no redistribute ospf* disables the routes.

redistribute bgp

redistribute bgp metric (1-4294967295) (0-4294967295) (0-255) (1-255) (1-65535)

no redistribute bgp
redistribute bgp redistributes routing information from bgp route entries into the EIGRP tables. *no redistribute bgp* disables the routes.

13.4 Show EIGRP Information

show ip eigrp topology
Display current EIGRP status.

```
eigrpd> **show ip eigrp topology**
# show ip eigrp topo

EIGRP Topology Table for AS(4)/ID(0.0.0.0)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply
       r - reply Status, s - sia Status

P  10.0.2.0/24, 1 successors, FD is 256256, serno: 0
       via Connected, enp0s3
```

13.5 EIGRP Debug Commands

Debug for EIGRP protocol.

debug eigrp packets
Debug eigrp packets

`debug eigrp` will show EIGRP packets that are sent and received.

debug eigrp transmit
Debug eigrp transmit events

`debug eigrp transmit` will display detailed information about the EIGRP transmit events.

show debugging eigrp
Display *eigrpd*'s debugging option.

`show debugging eigrp` will show all information currently set for eigrpd debug.

CHAPTER 14

ISIS

ISIS (Intermediate System to Intermediate System) is a routing protocol which is described in *ISO10589*, **RFC 1195**, **RFC 5308**. ISIS is an IGP (Interior Gateway Protocol). Compared with RIP, ISIS can provide scalable network support and faster convergence times like OSPF. ISIS is widely used in large networks such as ISP (Internet Service Provider) and carrier backbone networks.

14.1 Configuring isisd

There are no *isisd* specific options. Common options can be specified (*Common Invocation Options*) to *isisd*. *isisd* needs to acquire interface information from *zebra* in order to function. Therefore *zebra* must be running before invoking *isisd*. Also, if *zebra* is restarted then *isisd* must be too.

Like other daemons, *isisd* configuration is done in ISIS specific configuration file `isisd.conf`.

14.2 ISIS router

To start ISIS process you have to specify the ISIS router. As of this writing, *isisd* does not support multiple ISIS processes.

router isis WORD

no router isis WORD
Enable or disable the ISIS process by specifying the ISIS domain with 'WORD'. *isisd* does not yet support multiple ISIS processes but you must specify the name of ISIS process. The ISIS process name 'WORD' is then used for interface (see command `ip router isis WORD`).

net XX.XXXX. ... .XXX.XX

no net XX.XXXX. ... .XXX.XX
Set/Unset network entity title (NET) provided in ISO format.

hostname dynamic

no hostname dynamic
Enable support for dynamic hostname.

area-password [clear | md5] <password>

domain-password [clear | md5] <password>

no area-password

no domain-password
Configure the authentication password for an area, respectively a domain, as clear text or md5 one.

log-adjacency-changes

no log-adjacency-changes
Log changes in adjacency state.

metric-style [narrow | transition | wide]

no metric-style
Set old-style (ISO 10589) or new-style packet formats:

- narrow Use old style of TLVs with narrow metric
- transition Send and accept both styles of TLVs during transition
- wide Use new style of TLVs to carry wider metric

set-overload-bit

no set-overload-bit
Set overload bit to avoid any transit traffic.

14.3 ISIS Timer

lsp-gen-interval (1-120)

lsp-gen-interval [level-1 | level-2] (1-120)

no lsp-gen-interval

no lsp-gen-interval [level-1 | level-2]
Set minimum interval in seconds between regenerating same LSP, globally, for an area (level-1) or a domain (level-2).

lsp-refresh-interval [level-1 | level-2] (1-65235)

no lsp-refresh-interval [level-1 | level-2]
Set LSP refresh interval in seconds, globally, for an area (level-1) or a domain (level-2).

max-lsp-lifetime (360-65535)

max-lsp-lifetime [level-1 | level-2] (360-65535)

no max-lsp-lifetime

no max-lsp-lifetime [level-1 | level-2]
Set LSP maximum LSP lifetime in seconds, globally, for an area (level-1) or a domain (level-2).

spf-interval (1-120)

spf-interval [level-1 | level-2] (1-120)

no spf-interval

no spf-interval [level-1 | level-2]
Set minimum interval between consecutive SPF calculations in seconds.

14.4 ISIS region

is-type [level-1 | level-1-2 | level-2-only]

no is-type
Define the ISIS router behavior:

- level-1 Act as a station router only
- level-1-2 Act as both a station router and an area router
- level-2-only Act as an area router only

14.5 ISIS interface

ip router isis WORD

no ip router isis WORD

Activate ISIS adjacency on this interface. Note that the name of ISIS instance must be the same as the one used to configure the ISIS process (see command `router isis WORD`).

isis circuit-type [level-1 | level-1-2 | level-2]

no isis circuit-type
Configure circuit type for interface:

- level-1 Level-1 only adjacencies are formed
- level-1-2 Level-1-2 adjacencies are formed
- level-2-only Level-2 only adjacencies are formed

isis csnp-interval (1-600)

isis csnp-interval (1-600) [level-1 | level-2]

no isis csnp-interval

no isis csnp-interval [level-1 | level-2]
Set CSNP interval in seconds globally, for an area (level-1) or a domain (level-2).

isis hello padding
Add padding to IS-IS hello packets.

isis hello-interval (1-600)

isis hello-interval (1-600) [level-1 | level-2]

no isis hello-interval

no isis hello-interval [level-1 | level-2]
Set Hello interval in seconds globally, for an area (level-1) or a domain (level-2).

isis hello-multiplier (2-100)

isis hello-multiplier (2-100) [level-1 | level-2]

no isis hello-multiplier

no isis hello-multiplier [level-1 | level-2]
Set multiplier for Hello holding time globally, for an area (level-1) or a domain (level-2).

isis metric [(0-255) | (0-16777215)]

isis metric [(0-255) | (0-16777215)] [level-1 | level-2]

no isis metric

no isis metric [level-1 | level-2]
Set default metric value globally, for an area (level-1) or a domain (level-2). Max value depend if metric support narrow or wide value (see command `metric-style [narrow | transition | wide]`).

isis network point-to-point

no isis network point-to-point
Set network type to 'Point-to-Point' (broadcast by default).

isis passive

no isis passive
Configure the passive mode for this interface.

isis password [clear | md5] <password>

no isis password
Configure the authentication password (clear or encoded text) for the interface.

isis priority (0-127)

isis priority (0-127) [level-1 | level-2]

no isis priority

no isis priority [level-1 | level-2]
Set priority for Designated Router election, globally, for the area (level-1) or the domain (level-2).

isis psnp-interval (1-120)

isis psnp-interval (1-120) [level-1 | level-2]

no isis psnp-interval

no isis psnp-interval [level-1 | level-2]
Set PSNP interval in seconds globally, for an area (level-1) or a domain (level-2).

isis three-way-handshake

no isis three-way-handshake
Enable or disable **RFC 5303** Three-Way Handshake for P2P adjacencies. Three-Way Handshake is enabled by default.

14.6 Showing ISIS information

show isis summary
Show summary information about ISIS.

show isis hostname
Show information about ISIS node.

show isis interface

show isis interface detail

show isis interface <interface name>
Show state and configuration of ISIS specified interface, or all interfaces if no interface is given with or without details.

show isis neighbor

show isis neighbor <System Id>

show isis neighbor detail
Show state and information of ISIS specified neighbor, or all neighbors if no system id is given with or without details.

show isis database

show isis database [detail]

show isis database <LSP id> [detail]

show isis database detail <LSP id>
Show the ISIS database globally, for a specific LSP id without or with details.

show isis topology

show isis topology [level-1|level-2]
Show topology IS-IS paths to Intermediate Systems, globally, in area (level-1) or domain (level-2).

show ip route isis
Show the ISIS routing table, as determined by the most recent SPF calculation.

14.7 Traffic Engineering

mpls-te on

no mpls-te
Enable Traffic Engineering LSP flooding.

mpls-te router-address <A.B.C.D>

no mpls-te router-address
Configure stable IP address for MPLS-TE.

show isis mpls-te interface

show isis mpls-te interface INTERFACE
Show MPLS Traffic Engineering parameters for all or specified interface.

show isis mpls-te router
Show Traffic Engineering router parameters.

See also:

Traffic Engineering

14.8 Debugging ISIS

debug isis adj-packets

no debug isis adj-packets
IS-IS Adjacency related packets.

debug isis checksum-errors

no debug isis checksum-errors
IS-IS LSP checksum errors.

debug isis events

no debug isis events
IS-IS Events.

debug isis local-updates

no debug isis local-updates
IS-IS local update packets.

debug isis packet-dump

no debug isis packet-dump
IS-IS packet dump.

debug isis protocol-errors

no debug isis protocol-errors
IS-IS LSP protocol errors.

debug isis route-events

no debug isis route-events
IS-IS Route related events.

debug isis snp-packets

no debug isis snp-packets
IS-IS CSNP/PSNP packets.

debug isis spf-events

debug isis spf-statistics

debug isis spf-triggers

no debug isis spf-events

no debug isis spf-statistics

no debug isis spf-triggers
IS-IS Shortest Path First Events, Timing and Statistic Data and triggering events.

debug isis update-packets

no debug isis update-packets
Update related packets.

show debugging isis
Print which ISIS debug level is activate.

14.9 ISIS Configuration Examples

A simple example, with MD5 authentication enabled:

```
!
interface eth0
 ip router isis FOO
 isis network point-to-point
```

(continues on next page)

(continued from previous page)

```
 isis circuit-type level-2-only
!
router isis FOO
net 47.0023.0000.0000.0000.0000.0000.0000.1900.0004.00
 metric-style wide
 is-type level-2-only
```

A Traffic Engineering configuration, with Inter-ASv2 support.

First, the `zebra.conf` part:

```
hostname HOSTNAME
password PASSWORD
log file /var/log/zebra.log
!
interface eth0
 ip address 10.2.2.2/24
 link-params
  max-bw 1.25e+07
  max-rsv-bw 1.25e+06
  unrsv-bw 0 1.25e+06
  unrsv-bw 1 1.25e+06
  unrsv-bw 2 1.25e+06
  unrsv-bw 3 1.25e+06
  unrsv-bw 4 1.25e+06
  unrsv-bw 5 1.25e+06
  unrsv-bw 6 1.25e+06
  unrsv-bw 7 1.25e+06
  admin-grp 0xab
!
interface eth1
 ip address 10.1.1.1/24
 link-params
  enable
  metric 100
  max-bw 1.25e+07
  max-rsv-bw 1.25e+06
  unrsv-bw 0 1.25e+06
  unrsv-bw 1 1.25e+06
  unrsv-bw 2 1.25e+06
  unrsv-bw 3 1.25e+06
  unrsv-bw 4 1.25e+06
  unrsv-bw 5 1.25e+06
  unrsv-bw 6 1.25e+06
  unrsv-bw 7 1.25e+06
  neighbor 10.1.1.2 as 65000
```

Then the `isisd.conf` itself:

```
hostname HOSTNAME
password PASSWORD
log file /var/log/isisd.log
!
!
interface eth0
 ip router isis FOO
!
```

(continues on next page)

(continued from previous page)

```
interface eth1
 ip router isis FOO
!
!
router isis FOO
 isis net 47.0023.0000.0000.0000.0000.0000.0000.1900.0004.00
  mpls-te on
  mpls-te router-address 10.1.1.1
!
line vty
```

CHAPTER 15

NHRP

nhrpd is an implementation of the :abbr:NHRP *(Next Hop Routing Protocol)*. NHRP is described in :rfc‘2332‘.

NHRP is used to improve the efficiency of routing computer network traffic over NBMA (Non-Broadcast, Multiple Access) networks. NHRP provides an ARP-like solution that allows a system to dynamically learn the NBMA address of the other systems that are part of that network, allowing these systems to directly communicate without requiring traffic to use an intermediate hop.

Cisco Dynamic Multipoint VPN (DMVPN) is based on NHRP, and frr nhrpd implements this scenario.

15.1 Routing Design

nhrpd never handles routing of prefixes itself. You need to run some real routing protocol (e.g. BGP) to advertise routes over the tunnels. What nhrpd does it establishes ‘shortcut routes’ that optimizes the routing protocol to avoid going through extra nodes in NBMA GRE mesh.

nhrpd does route NHRP domain addresses individually using per-host prefixes. This is similar to Cisco FlexVPN; but in contrast to opennhrp which uses a generic subnet route.

To create NBMA GRE tunnel you might use the following (Linux terminal commands)::

```
ip tunnel add gre1 mode gre key 42 ttl 64
ip addr add 10.255.255.2/32 dev gre1
ip link set gre1 up
```

Note that the IP-address is assigned as host prefix to gre1. nhrpd will automatically create additional host routes pointing to gre1 when a connection with these hosts is established.

The gre1 subnet prefix should be announced by routing protocol from the hub nodes (e.g. BGP ‘network’ announce). This allows the routing protocol to decide which is the closest hub and determine the relay hub on prefix basis when direct tunnel is not established.

nhrpd will redistribute directly connected neighbors to zebra. Within hub nodes, these routes should be internally redistributed using some routing protocol (e.g. iBGP) to allow hubs to be able to relay all traffic.

This can be achieved in hubs with the following bgp configuration (network command defines the GRE subnet):

```
router bgp 65555
 address-family ipv4 unicast
   network 172.16.0.0/16
   redistribute nhrp
 exit-address-family
```

15.2 Configuring NHRP

FIXME

15.3 Hub Functionality

In addition to routing nhrp redistributed host prefixes, the hub nodes are also responsible to send NHRP Traffic Indication messages that trigger creation of the shortcut tunnels.

nhrpd sends Traffic Indication messages based on network traffic captured using NFLOG. Typically you want to send Traffic Indications for network traffic that is routed from gre1 back to gre1 in rate limited manner. This can be achieved with the following iptables rule.

```
iptables -A FORWARD -i gre1 -o gre1 \\
    -m hashlimit --hashlimit-upto 4/minute --hashlimit-burst 1 \\
    --hashlimit-mode srcip,dstip --hashlimit-srcmask 24 --hashlimit-dstmask 24 \\
    --hashlimit-name loglimit-0 -j NFLOG --nflog-group 1 --nflog-range 128
```

You can fine tune the src/dstmask according to the prefix lengths you announce internal, add additional IP range matches, or rate limitation if needed. However, the above should be good in most cases.

This kernel NFLOG target's nflog-group is configured in global nhrp config with:

```
nhrp nflog-group 1
```

To start sending these traffic notices out from hubs, use the nhrp per-interface directive:

```
interface gre1
 ip nhrp redirect
```

15.4 Integration with IKE

nhrpd needs tight integration with IKE daemon for various reasons. Currently only strongSwan is supported as IKE daemon.

nhrpd connects to strongSwan using VICI protocol based on UNIX socket (hardcoded now as /var/run/charon.vici).

strongSwan currently needs few patches applied. Please check out the http://git.alpinelinux.org/cgit/user/tteras/strongswan/log/?h=tteras-release,release and http://git.alpinelinux.org/cgit/user/tteras/strongswan/log/?h=tteras,working tree git repositories for the patches.

15.5 NHRP Events

FIXME

15.6 Configuration Example

FIXME

CHAPTER 16

OSPFv2

OSPF (Open Shortest Path First) version 2 is a routing protocol which is described in **RFC 2328**. OSPF is an IGP. Compared with RIP, OSPF can provide scalable network support and faster convergence times. OSPF is widely used in large networks such as ISP backbone and enterprise networks.

16.1 OSPF Fundamentals

OSPF is, mostly, a link-state routing protocol. In contrast to *distance-vector* protocols, such as RIP or BGP, where routers describe available *paths* (i.e. routes) to each other, in *link-state* protocols routers instead describe the state of their links to their immediate neighbouring routers.

Each router describes their link-state information in a message known as an LSA (Link State Advertisement), which is then propagated through to all other routers in a link-state routing domain, by a process called *flooding*. Each router thus builds up an LSDB (Link State Database) of all the link-state messages. From this collection of LSAs in the LSDB, each router can then calculate the shortest path to any other router, based on some common metric, by using an algorithm such as Edgar Djikstra's SPF (Shortest Path First) algorithm.

By describing connectivity of a network in this way, in terms of routers and links rather than in terms of the paths through a network, a link-state protocol can use less bandwidth and converge more quickly than other protocols. A link-state protocol need distribute only one link-state message throughout the link-state domain when a link on any single given router changes state, in order for all routers to reconverge on the best paths through the network. In contrast, distance vector protocols can require a progression of different path update messages from a series of different routers in order to converge.

The disadvantage to a link-state protocol is that the process of computing the best paths can be relatively intensive when compared to distance-vector protocols, in which near to no computation need be done other than (potentially) select between multiple routes. This overhead is mostly negligible for modern embedded CPUs, even for networks with thousands of nodes. The primary scaling overhead lies more in coping with the ever greater frequency of LSA updates as the size of a link-state area increases, in managing the LSDB and required flooding.

This section aims to give a distilled, but accurate, description of the more important workings of OSPF which an administrator may need to know to be able best configure and trouble-shoot OSPF.

16.1.1 OSPF Mechanisms

OSPF defines a range of mechanisms, concerned with detecting, describing and propagating state through a network. These mechanisms will nearly all be covered in greater detail further on. They may be broadly classed as:

The Hello Protocol

The OSPF Hello protocol allows OSPF to quickly detect changes in two-way reachability between routers on a link. OSPF can additionally avail of other sources of reachability information, such as link-state information provided by hardware, or through dedicated reachability protocols such as BFD (Bidirectional Forwarding Detection).

OSPF also uses the Hello protocol to propagate certain state between routers sharing a link, for example:

- Hello protocol configured state, such as the dead-interval.
- Router priority, for DR/BDR election.
- DR/BDR election results.
- Any optional capabilities supported by each router.

The Hello protocol is comparatively trivial and will not be explored in greater detail than here.

LSAs

At the heart of OSPF are LSA messages. Despite the name, some LSA s do not, strictly speaking, describe link-state information. Common LSA s describe information such as:

- Routers, in terms of their links.
- Networks, in terms of attached routers.
- Routes, external to a link-state domain:

  External Routes Routes entirely external to OSPF. Routers originating such routes are known as ASBR (Autonomous-System Border Router) routers.

  Summary Routes Routes which summarise routing information relating to OSPF areas external to the OSPF link-state area at hand, originated by ABR (Area Boundary Router) routers.

LSA Flooding

OSPF defines several related mechanisms, used to manage synchronisation of LSDB s between neighbours as neighbours form adjacencies and the propagation, or *flooding* of new or updated LSA s.

Areas

OSPF provides for the protocol to be broken up into multiple smaller and independent link-state areas. Each area must be connected to a common backbone area by an ABR. These ABR routers are responsible for summarising the link-state routing information of an area into *Summary LSAs*, possibly in a condensed (i.e. aggregated) form, and then originating these summaries into all other areas the ABR is connected to.

Note that only summaries and external routes are passed between areas. As these describe *paths*, rather than any router link-states, routing between areas hence is by *distance-vector*, **not** link-state.

16.1.2 OSPF LSAs

The core objects in OSPF are LSA s. Everything else in OSPF revolves around detecting what to describe in LSAs, when to update them, how to flood them throughout a network and how to calculate routes from them.

There are a variety of different LSA s, for purposes such as describing actual link-state information, describing paths (i.e. routes), describing bandwidth usage of links for TE (Traffic Engineering) purposes, and even arbitrary data by way of *Opaque* LSA s.

LSA Header

All LSAs share a common header with the following information:

- Type

  Different types of LSA s describe different things in OSPF. Types include:

  - Router LSA
  - Network LSA
  - Network Summary LSA
  - Router Summary LSA
  - AS-External LSA

  The specifics of the different types of LSA are examined below.

- Advertising Router

  The Router ID of the router originating the LSA.

See also:

`ospf router-id A.B.C.D`.

- LSA ID

  The ID of the LSA, which is typically derived in some way from the information the LSA describes, e.g. a Router LSA uses the Router ID as the LSA ID, a Network LSA will have the IP address of the DR as its LSA ID.

  The combination of the Type, ID and Advertising Router ID must uniquely identify the LSA. There can however be multiple instances of an LSA with the same Type, LSA ID and Advertising Router ID, see *sequence number*.

- Age

  A number to allow stale LSA s to, eventually, be purged by routers from their LSDB s.

  The value nominally is one of seconds. An age of 3600, i.e. 1 hour, is called the *MaxAge*. MaxAge LSAs are ignored in routing calculations. LSAs must be periodically refreshed by their Advertising Router before reaching MaxAge if they are to remain valid.

  Routers may deliberately flood LSAs with the age artificially set to 3600 to indicate an LSA is no longer valid. This is called *flushing* of an LSA.

  It is not abnormal to see stale LSAs in the LSDB, this can occur where a router has shutdown without flushing its LSA(s), e.g. where it has become disconnected from the network. Such LSAs do little harm.

- Sequence Number

  A number used to distinguish newer instances of an LSA from older instances.

Link-State LSAs

Of all the various kinds of LSA s, just two types comprise the actual link-state part of OSPF, Router LSA s and Network LSA s. These LSA types are absolutely core to the protocol.

Instances of these LSAs are specific to the link-state area in which they are originated. Routes calculated from these two LSA types are called *intra-area routes*.

- Router LSA

  Each OSPF Router must originate a router LSA to describe itself. In it, the router lists each of its OSPF enabled interfaces, for the given link-state area, in terms of:

  Cost The output cost of that interface, scaled inversely to some commonly known reference value, `auto-cost reference-bandwidth (1-4294967`.

  Link Type Transit Network

  A link to a multi-access network, on which the router has at least one Full adjacency with another router.

  PTP (Point-to-Point) A link to a single remote router, with a Full adjacency. No DR (Designated Router) is elected on such links; no network LSA is originated for such a link.

  Stub A link with no adjacent neighbours, or a host route.

  – Link ID and Data

  These values depend on the Link Type:

Link Type	Link ID	Link Data
Transit	Link IP address of the DR	Interface IP address
Point-to-Point	Router ID of the remote router	Local interface IP address, or the IFINDEX (MIB-II interface index) for unnumbered links
Stub	IP address	Subnet Mask

  Links on a router may be listed multiple times in the Router LSA, e.g. a PTP interface on which OSPF is enabled must *always* be described by a Stub link in the Router LSA, in addition to being listed as PtP link in the Router LSA if the adjacency with the remote router is Full.

  Stub links may also be used as a way to describe links on which OSPF is *not* spoken, known as *passive interfaces*, see `passive-interface INTERFACE`.

- Network LSA

  On multi-access links (e.g. ethernets, certain kinds of ATM and X.25 configurations), routers elect a DR. The DR is responsible for originating a Network LSA, which helps reduce the information needed to describe multi-access networks with multiple routers attached. The DR also acts as a hub for the flooding of LSA s on that link, thus reducing flooding overheads.

  The contents of the Network LSA describes the:

  – Subnet Mask

  As the LSA ID of a Network LSA must be the IP address of the DR, the Subnet Mask together with the LSA ID gives you the network address.

  – Attached Routers

  Each router fully-adjacent with the DR is listed in the LSA, by their Router-ID. This allows the corresponding Router LSA s to be easily retrieved from the LSDB.

Summary of Link State LSAs:

LSA Type	LSA ID	LSA Data Describes
Router LSA	Router ID	The OSPF enabled links of the router, within a specific link-state area.
Network LSA	The IP address of the DR for the network	The subnet mask of the network and the Router IDs of all routers on the network

With an LSDB composed of just these two types of LSA, it is possible to construct a directed graph of the connectivity between all routers and networks in a given OSPF link-state area. So, not surprisingly, when OSPF routers build updated routing tables, the first stage of SPF calculation concerns itself only with these two LSA types.

Link-State LSA Examples

The example below shows two LSA s, both originated by the same router (Router ID 192.168.0.49) and with the same LSA ID (192.168.0.49), but of different LSA types.

The first LSA being the router LSA describing 192.168.0.49's links: 2 links to multi-access networks with fully-adjacent neighbours (i.e. Transit links) and 1 being a Stub link (no adjacent neighbours).

The second LSA being a Network LSA, for which 192.168.0.49 is the DR, listing the Router IDs of 4 routers on that network which are fully adjacent with 192.168.0.49.

```
# show ip ospf database router 192.168.0.49

       OSPF Router with ID (192.168.0.53)

                Router Link States (Area 0.0.0.0)

  LS age: 38
  Options: 0x2  : *|-|-|-|-|-|E|*
  LS Flags: 0x6
  Flags: 0x2 : ASBR
  LS Type: router-LSA
  Link State ID: 192.168.0.49
  Advertising Router: 192.168.0.49
  LS Seq Number: 80000f90
  Checksum: 0x518b
  Length: 60
   Number of Links: 3

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 192.168.1.3
     (Link Data) Router Interface address: 192.168.1.3
      Number of TOS metrics: 0
       TOS 0 Metric: 10

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 192.168.0.49
     (Link Data) Router Interface address: 192.168.0.49
      Number of TOS metrics: 0
       TOS 0 Metric: 10

    Link connected to: Stub Network
     (Link ID) Net: 192.168.3.190
     (Link Data) Network Mask: 255.255.255.255
```

(continues on next page)

(continued from previous page)

```
      Number of TOS metrics: 0
       TOS 0 Metric: 39063
# show ip ospf database network 192.168.0.49

       OSPF Router with ID (192.168.0.53)

                Net Link States (Area 0.0.0.0)

  LS age: 285
  Options: 0x2  : *|-|-|-|-|-|E|*
  LS Flags: 0x6
  LS Type: network-LSA
  Link State ID: 192.168.0.49 (address of Designated Router)
  Advertising Router: 192.168.0.49
  LS Seq Number: 80000074
  Checksum: 0x0103
  Length: 40
  Network Mask: /29
        Attached Router: 192.168.0.49
        Attached Router: 192.168.0.52
        Attached Router: 192.168.0.53
        Attached Router: 192.168.0.54
```

Note that from one LSA, you can find the other. E.g. Given the Network-LSA you have a list of Router IDs on that network, from which you can then look up, in the local LSDB, the matching Router LSA. From that Router-LSA you may (potentially) find links to other Transit networks and Routers IDs which can be used to lookup the corresponding Router or Network LSA. And in that fashion, one can find all the Routers and Networks reachable from that starting LSA.

Given the Router LSA instead, you have the IP address of the DR of any attached transit links. Network LSAs will have that IP as their LSA ID, so you can then look up that Network LSA and from that find all the attached routers on that link, leading potentially to more links and Network and Router LSAs, etc. etc.

From just the above two LSA s, one can already see the following partial topology:

```
------------------------ Network: ......
            |            Designated Router IP: 192.168.1.3
            |
      IP: 192.168.1.3
       (transit link)
        (cost: 10)
   Router ID: 192.168.0.49(stub)---------- IP: 192.168.3.190/32
        (cost: 10)         (cost: 39063)
       (transit link)
      IP: 192.168.0.49
            |
            |
------------------------------ Network: 192.168.0.48/29
  |         |           |       Designated Router IP: 192.168.0.49
  |         |           |
  |         |     Router ID: 192.168.0.54
  |         |
  |   Router ID: 192.168.0.53
  |
Router ID: 192.168.0.52
```

Note the Router IDs, though they look like IP addresses and often are IP addresses, are not strictly speaking IP

addresses, nor need they be reachable addresses (though, OSPF will calculate routes to Router IDs).

External LSAs

External, or "Type 5", LSA s describe routing information which is entirely external to OSPF, and is "injected" into OSPF. Such routing information may have come from another routing protocol, such as RIP or BGP, they may represent static routes or they may represent a default route.

An OSPF router which originates External LSA s is known as an ASBR. Unlike the link-state LSA s, and most other LSA s, which are flooded only within the area in which they originate, External LSA s are flooded through-out the OSPF network to all areas capable of carrying External LSA s (*Areas*).

Routes internal to OSPF (intra-area or inter-area) are always preferred over external routes.

The External LSA describes the following:

IP Network number The IP Network number of the route is described by the LSA ID field.

IP Network Mask The body of the External LSA describes the IP Network Mask of the route. This, together with the LSA ID, describes the prefix of the IP route concerned.

Metric The cost of the External Route. This cost may be an OSPF cost (also known as a "Type 1" metric), i.e. equivalent to the normal OSPF costs, or an externally derived cost ("Type 2" metric) which is not comparable to OSPF costs and always considered larger than any OSPF cost. Where there are both Type 1 and 2 External routes for a route, the Type 1 is always preferred.

Forwarding Address The address of the router to forward packets to for the route. This may be, and usually is, left as 0 to specify that the ASBR originating the External LSA should be used. There must be an internal OSPF route to the forwarding address, for the forwarding address to be usable.

Tag An arbitrary 4-bytes of data, not interpreted by OSPF, which may carry whatever information about the route which OSPF speakers desire.

AS External LSA Example

To illustrate, below is an example of an External LSA in the LSDB of an OSPF router. It describes a route to the IP prefix of 192.168.165.0/24, originated by the ASBR with Router-ID 192.168.0.49. The metric of 20 is external to OSPF. The forwarding address is 0, so the route should forward to the originating ASBR if selected.

```
# show ip ospf database external 192.168.165.0
  LS age: 995
  Options: 0x2  : *|-|-|-|-|-|E|*
  LS Flags: 0x9
  LS Type: AS-external-LSA
  Link State ID: 192.168.165.0 (External Network Number)
  Advertising Router: 192.168.0.49
  LS Seq Number: 800001d8
  Checksum: 0xea27
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 20
        Forward Address: 0.0.0.0
        External Route Tag: 0
```

We can add this to our partial topology from above, which now looks like::

```
---------------------- Network: ......
         |             Designated Router IP: 192.168.1.3
         |
   IP: 192.168.1.3       /---- External route: 192.168.165.0/24
    (transit link)      /                Cost: 20 (External metric)
     (cost: 10)        /
Router ID: 192.168.0.49(stub)---------- IP: 192.168.3.190/32
     (cost: 10)         (cost: 39063)
    (transit link)
   IP: 192.168.0.49
         |
         |
------------------------------ Network: 192.168.0.48/29
  |        |           |       Designated Router IP: 192.168.0.49
  |        |           |
  |        |     Router ID: 192.168.0.54
  |        |
  |   Router ID: 192.168.0.53
  |
Router ID: 192.168.0.52
```

Summary LSAs

Summary LSAs are created by ABR s to summarise the destinations available within one area to other areas. These LSAs may describe IP networks, potentially in aggregated form, or ASBR routers.

16.2 Configuring ospfd

There are no *ospfd* specific options. Common options can be specified (*Common Invocation Options*) to *ospfd*. *ospfd* needs to acquire interface information from *zebra* in order to function. Therefore *zebra* must be running before invoking *ospfd*. Also, if *zebra* is restarted then *ospfd* must be too.

Like other daemons, *ospfd* configuration is done in OSPF specific configuration file `ospfd.conf`.

16.3 OSPF router

To start OSPF process you have to specify the OSPF router. As of this writing, *ospfd* does not support multiple OSPF processes.

router ospf

no router ospf
: Enable or disable the OSPF process. *ospfd* does not yet support multiple OSPF processes. So you can not specify an OSPF process number.

ospf router-id A.B.C.D

no ospf router-id
: This sets the router-ID of the OSPF process. The router-ID may be an IP address of the router, but need not be - it can be any arbitrary 32bit number. However it MUST be unique within the entire OSPF domain to the OSPF speaker - bad things will happen if multiple OSPF speakers are configured with the same router-ID! If one is not specified then *ospfd* will obtain a router-ID automatically from *zebra*.

ospf abr-type TYPE

no ospf abr-type TYPE

type can be cisco|ibm|shortcut|standard. The "Cisco" and "IBM" types are equivalent.

The OSPF standard for ABR behaviour does not allow an ABR to consider routes through non-backbone areas when its links to the backbone are down, even when there are other ABRs in attached non-backbone areas which still can reach the backbone - this restriction exists primarily to ensure routing-loops are avoided.

With the "Cisco" or "IBM" ABR type, the default in this release of FRR, this restriction is lifted, allowing an ABR to consider summaries learned from other ABRs through non-backbone areas, and hence route via non-backbone areas as a last resort when, and only when, backbone links are down.

Note that areas with fully-adjacent virtual-links are considered to be "transit capable" and can always be used to route backbone traffic, and hence are unaffected by this setting (`area A.B.C.D virtual-link A.B.C.D`).

More information regarding the behaviour controlled by this command can be found in **RFC 3509**, and *draft-ietf-ospf-shortcut-abr-02.txt*.

Quote: "Though the definition of the ABR in the OSPF specification does not require a router with multiple attached areas to have a backbone connection, it is actually necessary to provide successful routing to the inter-area and external destinations. If this requirement is not met, all traffic destined for the areas not connected to such an ABR or out of the OSPF domain, is dropped. This document describes alternative ABR behaviors implemented in Cisco and IBM routers."

ospf rfc1583compatibility

no ospf rfc1583compatibility

RFC 2328, the successor to **RFC 1583**, suggests according to section G.2 (changes) in section 16.4 a change to the path preference algorithm that prevents possible routing loops that were possible in the old version of OSPFv2. More specifically it demands that inter-area paths and intra-area backbone path are now of equal preference but still both preferred to external paths.

This command should NOT be set normally.

log-adjacency-changes [detail]

no log-adjacency-changes [detail]

Configures ospfd to log changes in adjacency. With the optional detail argument, all changes in adjacency status are shown. Without detail, only changes to full or regressions are shown.

passive-interface INTERFACE

no passive-interface INTERFACE

Do not speak OSPF interface on the given interface, but do advertise the interface as a stub link in the router-LSA for this router. This allows one to advertise addresses on such connected interfaces without having to originate AS-External/Type-5 LSAs (which have global flooding scope) - as would occur if connected addresses were redistributed into OSPF (*Redistribute routes to OSPF*). This is the only way to advertise non-OSPF links into stub areas.

timers throttle spf DELAY INITIAL-HOLDTIME MAX-HOLDTIME

no timers throttle spf

This command sets the initial *delay*, the *initial-holdtime* and the *maximum-holdtime* between when SPF is calculated and the event which triggered the calculation. The times are specified in milliseconds and must be in the range of 0 to 600000 milliseconds.

The *delay* specifies the minimum amount of time to delay SPF calculation (hence it affects how long SPF calculation is delayed after an event which occurs outside of the holdtime of any previous SPF calculation, and also serves as a minimum holdtime).

Consecutive SPF calculations will always be separated by at least 'hold-time' milliseconds. The hold-time is adaptive and initially is set to the *initial-holdtime* configured with the above command. Events which occur within the holdtime of the previous SPF calculation will cause the holdtime to be increased by *initial-holdtime*, bounded by the *maximum-holdtime* configured with this command. If the adaptive hold-time elapses without any SPF-triggering event occurring then the current holdtime is reset to the *initial-holdtime*. The current holdtime can be viewed with `show ip ospf`, where it is expressed as a multiplier of the *initial-holdtime*.

```
router ospf
timers throttle spf 200 400 10000
```

In this example, the *delay* is set to 200ms, the initial holdtime is set to 400ms and the *maximum holdtime* to 10s. Hence there will always be at least 200ms between an event which requires SPF calculation and the actual SPF calculation. Further consecutive SPF calculations will always be separated by between 400ms to 10s, the hold-time increasing by 400ms each time an SPF-triggering event occurs within the hold-time of the previous SPF calculation.

This command supercedes the *timers spf* command in previous FRR releases.

max-metric router-lsa [on-startup|on-shutdown] (5-86400)

max-metric router-lsa administrative

no max-metric router-lsa [on-startup|on-shutdown|administrative]

This enables **RFC 3137** support, where the OSPF process describes its transit links in its router-LSA as having infinite distance so that other routers will avoid calculating transit paths through the router while still being able to reach networks through the router.

This support may be enabled administratively (and indefinitely) or conditionally. Conditional enabling of max-metric router-lsas can be for a period of seconds after startup and/or for a period of seconds prior to shutdown.

Enabling this for a period after startup allows OSPF to converge fully first without affecting any existing routes used by other routers, while still allowing any connected stub links and/or redistributed routes to be reachable. Enabling this for a period of time in advance of shutdown allows the router to gracefully excuse itself from the OSPF domain.

Enabling this feature administratively allows for administrative intervention for whatever reason, for an indefinite period of time. Note that if the configuration is written to file, this administrative form of the stub-router command will also be written to file. If *ospfd* is restarted later, the command will then take effect until manually deconfigured.

Configured state of this feature as well as current status, such as the number of second remaining till on-startup or on-shutdown ends, can be viewed with the `show ip ospf` command.

auto-cost reference-bandwidth (1-4294967)

no auto-cost reference-bandwidth

This sets the reference bandwidth for cost calculations, where this bandwidth is considered equivalent to an OSPF cost of 1, specified in Mbits/s. The default is 100Mbit/s (i.e. a link of bandwidth 100Mbit/s or higher will have a cost of 1. Cost of lower bandwidth links will be scaled with reference to this cost).

This configuration setting MUST be consistent across all routers within the OSPF domain.

network A.B.C.D/M area A.B.C.D

network A.B.C.D/M area (0-4294967295)

no network A.B.C.D/M area A.B.C.D

no network A.B.C.D/M area (0-4294967295)

This command specifies the OSPF enabled interface(s). If the interface has an address from range 192.168.1.0/24 then the command below enables ospf on this interface so router can provide network information to the other ospf routers via this interface.

```
router ospf
network 192.168.1.0/24 area 0.0.0.0
```

Prefix length in interface must be equal or bigger (i.e. smaller network) than prefix length in network statement. For example statement above doesn't enable ospf on interface with address 192.168.1.1/23, but it does on interface with address 192.168.1.129/25.

Note that the behavior when there is a peer address defined on an interface changed after release 0.99.7. Currently, if a peer prefix has been configured, then we test whether the prefix in the network command contains the destination prefix. Otherwise, we test whether the network command prefix contains the local address prefix of the interface.

In some cases it may be more convenient to enable OSPF on a per interface/subnet basis (`ip ospf area AREA [ADDR]`).

16.4 OSPF area

area A.B.C.D range A.B.C.D/M

area (0-4294967295) range A.B.C.D/M

no area A.B.C.D range A.B.C.D/M

no area (0-4294967295) range A.B.C.D/M

Summarize intra area paths from specified area into one Type-3 summary-LSA announced to other areas. This command can be used only in ABR and ONLY router-LSAs (Type-1) and network-LSAs (Type-2) (i.e. LSAs with scope area) can be summarized. Type-5 AS-external-LSAs can't be summarized - their scope is AS. Summarizing Type-7 AS-external-LSAs isn't supported yet by FRR.

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 range 10.0.0.0/8
```

With configuration above one Type-3 Summary-LSA with routing info 10.0.0.0/8 is announced into backbone area if area 0.0.0.10 contains at least one intra-area network (i.e. described with router or network LSA) from this range.

area A.B.C.D range IPV4_PREFIX not-advertise

no area A.B.C.D range IPV4_PREFIX not-advertise

Instead of summarizing intra area paths filter them - i.e. intra area paths from this range are not advertised into other areas. This command makes sense in ABR only.

area A.B.C.D range IPV4_PREFIX substitute IPV4_PREFIX

no area A.B.C.D range IPV4_PREFIX substitute IPV4_PREFIX

Substitute summarized prefix with another prefix.

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 range 10.0.0.0/8 substitute 11.0.0.0/8
```

One Type-3 summary-LSA with routing info 11.0.0.0/8 is announced into backbone area if area 0.0.0.10 contains at least one intra-area network (i.e. described with router-LSA or network-LSA) from range 10.0.0.0/8. This command makes sense in ABR only.

area A.B.C.D virtual-link A.B.C.D

area (0-4294967295) virtual-link A.B.C.D

no area A.B.C.D virtual-link A.B.C.D

no area (0-4294967295) virtual-link A.B.C.D

area A.B.C.D shortcut

area (0-4294967295) shortcut

no area A.B.C.D shortcut

no area (0-4294967295) shortcut
Configure the area as Shortcut capable. See RFC 3509. This requires that the 'abr-type' be set to 'shortcut'.

area A.B.C.D stub

area (0-4294967295) stub

no area A.B.C.D stub

no area (0-4294967295) stub
Configure the area to be a stub area. That is, an area where no router originates routes external to OSPF and hence an area where all external routes are via the ABR(s). Hence, ABRs for such an area do not need to pass AS-External LSAs (type-5s) or ASBR-Summary LSAs (type-4) into the area. They need only pass Network-Summary (type-3) LSAs into such an area, along with a default-route summary.

area A.B.C.D stub no-summary

area (0-4294967295) stub no-summary

no area A.B.C.D stub no-summary

no area (0-4294967295) stub no-summary
Prevents an *ospfd* ABR from injecting inter-area summaries into the specified stub area.

area A.B.C.D default-cost (0-16777215)

no area A.B.C.D default-cost (0-16777215)
Set the cost of default-summary LSAs announced to stubby areas.

area A.B.C.D export-list NAME

area (0-4294967295) export-list NAME

no area A.B.C.D export-list NAME

no area (0-4294967295) export-list NAME
Filter Type-3 summary-LSAs announced to other areas originated from intra- area paths from specified area.

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 export-list foo
!
access-list foo permit 10.10.0.0/16
access-list foo deny any
```

With example above any intra-area paths from area 0.0.0.10 and from range 10.10.0.0/16 (for example 10.10.1.0/24 and 10.10.2.128/30) are announced into other areas as Type-3 summary-LSA's, but any others (for example 10.11.0.0/16 or 10.128.30.16/30) aren't.

This command is only relevant if the router is an ABR for the specified area.

area A.B.C.D import-list NAME

area (0-4294967295) import-list NAME

no area A.B.C.D import-list NAME

no area (0-4294967295) import-list NAME
Same as export-list, but it applies to paths announced into specified area as Type-3 summary-LSAs.

area A.B.C.D filter-list prefix NAME in

area A.B.C.D filter-list prefix NAME out

area (0-4294967295) filter-list prefix NAME in

area (0-4294967295) filter-list prefix NAME out

no area A.B.C.D filter-list prefix NAME in

no area A.B.C.D filter-list prefix NAME out

no area (0-4294967295) filter-list prefix NAME in

no area (0-4294967295) filter-list prefix NAME out
Filtering Type-3 summary-LSAs to/from area using prefix lists. This command makes sense in ABR only.

area A.B.C.D authentication

area (0-4294967295) authentication

no area A.B.C.D authentication

no area (0-4294967295) authentication
Specify that simple password authentication should be used for the given area.

area A.B.C.D authentication message-digest

area (0-4294967295) authentication message-digest
Specify that OSPF packets must be authenticated with MD5 HMACs within the given area. Keying material must also be configured on a per-interface basis (`ip ospf message-digest-key`).

MD5 authentication may also be configured on a per-interface basis (`ip ospf authentication message-digest`). Such per-interface settings will override any per-area authentication setting.

16.5 OSPF interface

ip ospf area AREA [ADDR]

no ip ospf area [ADDR]
Enable OSPF on the interface, optionally restricted to just the IP address given by *ADDR*, putting it in the *AREA* area. Per interface area settings take precedence to network commands (`network A.B.C.D/M area A.B.C.D`).

If you have a lot of interfaces, and/or a lot of subnets, then enabling OSPF via this command may result in a slight performance improvement.

ip ospf authentication-key AUTH_KEY

no ip ospf authentication-key
Set OSPF authentication key to a simple password. After setting *AUTH_KEY*, all OSPF packets are authenticated. *AUTH_KEY* has length up to 8 chars.

Simple text password authentication is insecure and deprecated in favour of MD5 HMAC authentication.

ip ospf authentication message-digest
Specify that MD5 HMAC authentication must be used on this interface. MD5 keying material must also be configured. Overrides any authentication enabled on a per-area basis (`area A.B.C.D authentication message-digest`)

Note that OSPF MD5 authentication requires that time never go backwards (correct time is NOT important, only that it never goes backwards), even across resets, if ospfd is to be able to promptly reestablish adjacencies with its neighbours after restarts/reboots. The host should have system time be set at boot from an external or non-volatile source (e.g. battery backed clock, NTP, etc.) or else the system clock should be periodically saved to non-volatile storage and restored at boot if MD5 authentication is to be expected to work reliably.

ip ospf message-digest-key KEYID md5 KEY

no ip ospf message-digest-key
Set OSPF authentication key to a cryptographic password. The cryptographic algorithm is MD5.

KEYID identifies secret key used to create the message digest. This ID is part of the protocol and must be consistent across routers on a link.

KEY is the actual message digest key, of up to 16 chars (larger strings will be truncated), and is associated with the given KEYID.

ip ospf cost (1-65535)

no ip ospf cost
Set link cost for the specified interface. The cost value is set to router-LSA's metric field and used for SPF calculation.

ip ospf dead-interval (1-65535)

ip ospf dead-interval minimal hello-multiplier (2-20)

no ip ospf dead-interval
Set number of seconds for RouterDeadInterval timer value used for Wait Timer and Inactivity Timer. This value must be the same for all routers attached to a common network. The default value is 40 seconds.

If 'minimal' is specified instead, then the dead-interval is set to 1 second and one must specify a hello-multiplier. The hello-multiplier specifies how many Hellos to send per second, from 2 (every 500ms) to 20 (every 50ms). Thus one can have 1s convergence time for OSPF. If this form is specified, then the hello-interval advertised in Hello packets is set to 0 and the hello-interval on received Hello packets is not checked, thus the hello-multiplier need NOT be the same across multiple routers on a common link.

ip ospf hello-interval (1-65535)

no ip ospf hello-interval
Set number of seconds for HelloInterval timer value. Setting this value, Hello packet will be sent every timer value seconds on the specified interface. This value must be the same for all routers attached to a common network. The default value is 10 seconds.

This command has no effect if `ip ospf dead-interval minimal hello-multiplier (2-20)` is also specified for the interface.

ip ospf network (broadcast|non-broadcast|point-to-multipoint|point-to-point)

no ip ospf network
Set explicitly network type for specified interface.

ip ospf priority (0-255)

no ip ospf priority
Set RouterPriority integer value. The router with the highest priority will be more eligible to become Designated Router. Setting the value to 0, makes the router ineligible to become Designated Router. The default value is 1.

ip ospf retransmit-interval (1-65535)

no ip ospf retransmit interval
Set number of seconds for RxmtInterval timer value. This value is used when retransmitting Database Description and Link State Request packets. The default value is 5 seconds.

ip ospf transmit-delay

no ip ospf transmit-delay
Set number of seconds for InfTransDelay value. LSAs' age should be incremented by this value when transmitting. The default value is 1 second.

ip ospf area (A.B.C.D|(0-4294967295))

no ip ospf area
Enable ospf on an interface and set associated area.

16.6 Redistribute routes to OSPF

redistribute (kernel|connected|static|rip|bgp)

redistribute (kernel|connected|static|rip|bgp) ROUTE-MAP

redistribute (kernel|connected|static|rip|bgp) metric-type (1|2)

redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) route-map WORD

redistribute (kernel|connected|static|rip|bgp) metric (0-16777214)

redistribute (kernel|connected|static|rip|bgp) metric (0-16777214) route-map WORD

redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) metric (0-16777214)

redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) metric (0-16777214) route-

no redistribute (kernel|connected|static|rip|bgp)

Redistribute routes of the specified protocol or kind into OSPF, with the metric type and metric set if specified, filtering the routes using the given route-map if specified. Redistributed routes may also be filtered with distribute-lists, see *ospf distribute-list configuration*.

Redistributed routes are distributed as into OSPF as Type-5 External LSAs into links to areas that accept external routes, Type-7 External LSAs for NSSA areas and are not redistributed at all into Stub areas, where external routes are not permitted.

Note that for connected routes, one may instead use the *passive-interface* configuration.

See also:

clicmd:*passive-interface INTERFACE*.

default-information originate

default-information originate metric (0-16777214)

default-information originate metric (0-16777214) metric-type (1|2)

default-information originate metric (0-16777214) metric-type (1|2) route-map WORD

default-information originate always

default-information originate always metric (0-16777214)

default-information originate always metric (0-16777214) metric-type (1|2)

default-information originate always metric (0-16777214) metric-type (1|2) route-map WORD

no default-information originate
Originate an AS-External (type-5) LSA describing a default route into all external-routing capable areas, of the specified metric and metric type. If the 'always' keyword is given then the default is always advertised, even when there is no default present in the routing table.

distribute-list NAME out (kernel|connected|static|rip|ospf

no distribute-list NAME out (kernel|connected|static|rip|ospf

Apply the access-list filter, NAME, to redistributed routes of the given type before allowing the routes to redistributed into OSPF (*ospf redistribution*).

default-metric (0-16777214)

no default-metric

distance (1-255)

no distance (1-255)

distance ospf (intra-area|inter-area|external) (1-255)

no distance ospf

router zebra

no router zebra

16.7 Showing OSPF information

show ip ospf
Show information on a variety of general OSPF and area state and configuration information.

show ip ospf interface [INTERFACE]
Show state and configuration of OSPF the specified interface, or all interfaces if no interface is given.

show ip ospf neighbor

show ip ospf neighbor INTERFACE

show ip ospf neighbor detail

show ip ospf neighbor INTERFACE detail

show ip ospf database

show ip ospf database (asbr-summary|external|network|router|summary)

show ip ospf database (asbr-summary|external|network|router|summary) LINK-STATE-ID

show ip ospf database (asbr-summary|external|network|router|summary) LINK-STATE-ID adv-rout

show ip ospf database (asbr-summary|external|network|router|summary) adv-router ADV-ROUTER

show ip ospf database (asbr-summary|external|network|router|summary) LINK-STATE-ID self-ori

show ip ospf database (asbr-summary|external|network|router|summary) self-originate

show ip ospf database max-age

show ip ospf database self-originate

show ip ospf route
Show the OSPF routing table, as determined by the most recent SPF calculation.

16.8 Opaque LSA

ospf opaque-lsa

capability opaque

no ospf opaque-lsa

no capability opaque
ospfd supports Opaque LSA (**RFC 2370**) as fundamental for MPLS Traffic Engineering LSA. Prior to used MPLS TE, opaque-lsa must be enable in the configuration file. Alternate command could be "mpls-te on" (*Traffic Engineering*).

show ip ospf database (opaque-link|opaque-area|opaque-external)

show ip ospf database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID

show ip ospf database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID adv-router AI

show ip ospf database (opaque-link|opaque-area|opaque-external) adv-router ADV-ROUTER

show ip ospf database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID self-originat

show ip ospf database (opaque-link|opaque-area|opaque-external) self-originate
Show Opaque LSA from the database.

16.9 Traffic Engineering

mpls-te on

no mpls-te
Enable Traffic Engineering LSA flooding.

mpls-te router-address <A.B.C.D>
Configure stable IP address for MPLS-TE. This IP address is then advertise in Opaque LSA Type-10 TLV=1 (TE) option 1 (Router-Address).

mpls-te inter-as area <area-id>|as

no mpls-te inter-as
Enable **RFC 5392** support - Inter-AS TE v2 - to flood Traffic Engineering parameters of Inter-AS link. 2 modes are supported: AREA and AS; LSA are flood in AREA <area-id> with Opaque Type-10, respectively in AS with Opaque Type-11. In all case, Opaque-LSA TLV=6.

show ip ospf mpls-te interface

show ip ospf mpls-te interface INTERFACE
Show MPLS Traffic Engineering parameters for all or specified interface.

show ip ospf mpls-te router
Show Traffic Engineering router parameters.

16.10 Router Information

router-info [as | area <A.B.C.D>]

no router-info
Enable Router Information (**RFC 4970**) LSA advertisement with AS scope (default) or Area scope flooding when area is specified.

pce address <A.B.C.D>

no pce address

pce domain as (0-65535)

no pce domain as (0-65535)

pce neighbor as (0-65535)

no pce neighbor as (0-65535)

pce flag BITPATTERN

no pce flag

pce scope BITPATTERN

no pce scope
The commands are conform to **RFC 5088** and allow OSPF router announce Path Computation Element (PCE) capabilities through the Router Information (RI) LSA. Router Information must be enable prior to this. The command set/unset respectively the PCE IP address, Autonomous System (AS) numbers of controlled domains, neighbor ASs, flag and scope. For flag and scope, please refer to :rfc‘5088‘ for the BITPATTERN recognition. Multiple ‘pce neighbor’ command could be specified in order to specify all PCE neighbours.

show ip ospf router-info
Show Router Capabilities flag.

show ip ospf router-info pce
Show Router Capabilities PCE parameters.

16.11 Segment Routing

This is an EXPERIMENTAL support of Segment Routing as per draft *draft-ietf-ospf-segment-routing-extensions-24.txt* for MPLS dataplane.

[no] segment-routing on
Enable Segment Routing. Even if this also activate routing information support, it is preferable to also activate routing information, and set accordingly the Area or AS flooding.

[no] segment-routing global-block (0-1048575) (0-1048575)
Fix the Segment Routing Global Block i.e. the label range used by MPLS to store label in the MPLS FIB.

[no] segment-routing node-msd (1-16)
Fix the Maximum Stack Depth supported by the router. The value depend of the MPLS dataplane. E.g. for Linux kernel, since version 4.13 it is 32.

[no] segment-routing prefix A.B.C.D/M index (0-65535) [no-php-flag]
Set the Segment Routing index for the specified prefix. Note that, only prefix with /32 corresponding to a loopback interface are currently supported. The ‘no-php-flag’ means NO Penultimate Hop Popping that allows SR node to request to its neighbor to not pop the label.

show ip ospf database segment-routing <adv-router ADVROUTER|self-originate> [json]
Show Segment Routing Data Base, all SR nodes, specific advertised router or self router. Optional JSON output can be obtained by appending ‘json’ to the end of the command.

16.12 Debugging OSPF

debug ospf packet (hello|dd|ls-request|ls-update|ls-ack|all) (send|recv) [detail]

no debug ospf packet (hello|dd|ls-request|ls-update|ls-ack|all) (send|recv) [detail]
Dump Packet for debugging

debug ospf ism

debug ospf ism (status|events|timers)

no debug ospf ism

no debug ospf ism (status|events|timers)
Show debug information of Interface State Machine

debug ospf nsm

debug ospf nsm (status|events|timers)

no debug ospf nsm

no debug ospf nsm (status|events|timers)
Show debug information of Network State Machine

debug ospf event

no debug ospf event
Show debug information of OSPF event

debug ospf nssa

no debug ospf nssa
Show debug information about Not So Stub Area

debug ospf lsa

debug ospf lsa (generate|flooding|refresh)

no debug ospf lsa

no debug ospf lsa (generate|flooding|refresh)
Show debug detail of Link State messages

debug ospf te

no debug ospf te
Show debug information about Traffic Engineering LSA

debug ospf zebra

debug ospf zebra (interface|redistribute)

no debug ospf zebra

no debug ospf zebra (interface|redistribute)
Show debug information of ZEBRA API

show debugging ospf

16.13 OSPF Configuration Examples

A simple example, with MD5 authentication enabled:

```
!
interface bge0
 ip ospf authentication message-digest
```

(continues on next page)

(continued from previous page)

```
 ip ospf message-digest-key 1 md5 ABCDEFGHIJK
!
router ospf
 network 192.168.0.0/16 area 0.0.0.1
 area 0.0.0.1 authentication message-digest
```

An ABR router, with MD5 authentication and performing summarisation of networks between the areas:

```
!
password ABCDEF
log file /var/log/frr/ospfd.log
service advanced-vty
!
interface eth0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 ABCDEFGHIJK
!
interface ppp0
!
interface br0
 ip ospf authentication message-digest
 ip ospf message-digest-key 2 md5 XYZ12345
!
router ospf
 ospf router-id 192.168.0.1
 redistribute connected
 passive interface ppp0
 network 192.168.0.0/24 area 0.0.0.0
 network 10.0.0.0/16 area 0.0.0.0
 network 192.168.1.0/24 area 0.0.0.1
 area 0.0.0.0 authentication message-digest
 area 0.0.0.0 range 10.0.0.0/16
 area 0.0.0.0 range 192.168.0.0/24
 area 0.0.0.1 authentication message-digest
 area 0.0.0.1 range 10.2.0.0/16
!
```

A Traffic Engineering configuration, with Inter-ASv2 support.

First, the `zebra.conf` part:

```
interface eth0
 ip address 198.168.1.1/24
 link-params
  enable
  admin-grp 0xa1
  metric 100
  max-bw 1.25e+07
  max-rsv-bw 1.25e+06
  unrsv-bw 0 1.25e+06
  unrsv-bw 1 1.25e+06
  unrsv-bw 2 1.25e+06
  unrsv-bw 3 1.25e+06
  unrsv-bw 4 1.25e+06
  unrsv-bw 5 1.25e+06
  unrsv-bw 6 1.25e+06
  unrsv-bw 7 1.25e+06
```

(continues on next page)

(continued from previous page)

```
!
interface eth1
 ip address 192.168.2.1/24
 link-params
  enable
  metric 10
  max-bw 1.25e+07
  max-rsv-bw 1.25e+06
  unrsv-bw 0 1.25e+06
  unrsv-bw 1 1.25e+06
  unrsv-bw 2 1.25e+06
  unrsv-bw 3 1.25e+06
  unrsv-bw 4 1.25e+06
  unrsv-bw 5 1.25e+06
  unrsv-bw 6 1.25e+06
  unrsv-bw 7 1.25e+06
  neighbor 192.168.2.2 as 65000
   hostname HOSTNAME
   password PASSWORD
   log file /var/log/zebra.log
   !
   interface eth0
    ip address 198.168.1.1/24
    link-params
     enable
     admin-grp 0xa1
     metric 100
     max-bw 1.25e+07
     max-rsv-bw 1.25e+06
     unrsv-bw 0 1.25e+06
     unrsv-bw 1 1.25e+06
     unrsv-bw 2 1.25e+06
     unrsv-bw 3 1.25e+06
     unrsv-bw 4 1.25e+06
     unrsv-bw 5 1.25e+06
     unrsv-bw 6 1.25e+06
     unrsv-bw 7 1.25e+06
   !
   interface eth1
    ip address 192.168.2.1/24
    link-params
     enable
     metric 10
     max-bw 1.25e+07
     max-rsv-bw 1.25e+06
     unrsv-bw 0 1.25e+06
     unrsv-bw 1 1.25e+06
     unrsv-bw 2 1.25e+06
     unrsv-bw 3 1.25e+06
     unrsv-bw 4 1.25e+06
     unrsv-bw 5 1.25e+06
     unrsv-bw 6 1.25e+06
     unrsv-bw 7 1.25e+06
     neighbor 192.168.2.2 as 65000
```

Then the `ospfd.conf` itself:

```
hostname HOSTNAME
password PASSWORD
log file /var/log/ospfd.log
!
!
interface eth0
 ip ospf hello-interval 60
 ip ospf dead-interval 240
!
interface eth1
 ip ospf hello-interval 60
 ip ospf dead-interval 240
!
!
router ospf
 ospf router-id 192.168.1.1
 network 192.168.0.0/16 area 1
 ospf opaque-lsa
 mpls-te
 mpls-te router-address 192.168.1.1
 mpls-te inter-as area 1
!
line vty
```

A router information example with PCE advertisement:

```
!
router ospf
 ospf router-id 192.168.1.1
 network 192.168.0.0/16 area 1
 capability opaque
 mpls-te
 mpls-te router-address 192.168.1.1
 router-info area 0.0.0.1
 pce address 192.168.1.1
 pce flag 0x80
 pce domain as 65400
 pce neighbor as 65500
 pce neighbor as 65200
 pce scope 0x80
!
```

CHAPTER 17

OSPFv3

ospf6d is a daemon support OSPF version 3 for IPv6 network. OSPF for IPv6 is described in **RFC 2740**.

17.1 OSPF6 router

router ospf6

router-id A.B.C.D
Set router's Router-ID.

interface IFNAME area AREA
Bind interface to specified area, and start sending OSPF packets. *area* can be specified as 0.

timers throttle spf DELAY INITIAL-HOLDTIME MAX-HOLDTIME

no timers throttle spf
This command sets the initial *delay*, the *initial-holdtime* and the *maximum-holdtime* between when SPF is calculated and the event which triggered the calculation. The times are specified in milliseconds and must be in the range of 0 to 600000 milliseconds.

The *delay* specifies the minimum amount of time to delay SPF calculation (hence it affects how long SPF calculation is delayed after an event which occurs outside of the holdtime of any previous SPF calculation, and also serves as a minimum holdtime).

Consecutive SPF calculations will always be separated by at least 'hold-time' milliseconds. The hold-time is adaptive and initially is set to the *initial-holdtime* configured with the above command. Events which occur within the holdtime of the previous SPF calculation will cause the holdtime to be increased by *initial-holdtime*, bounded by the *maximum-holdtime* configured with this command. If the adaptive hold-time elapses without any SPF-triggering event occurring then the current holdtime is reset to the *initial-holdtime*.

```
router ospf6
 timers throttle spf 200 400 10000
```

In this example, the *delay* is set to 200ms, the initial holdtime is set to 400ms and the *maximum holdtime* to 10s. Hence there will always be at least 200ms between an event which requires SPF calculation and the actual

SPF calculation. Further consecutive SPF calculations will always be separated by between 400ms to 10s, the hold-time increasing by 400ms each time an SPF-triggering event occurs within the hold-time of the previous SPF calculation.

auto-cost reference-bandwidth COST

no auto-cost reference-bandwidth
This sets the reference bandwidth for cost calculations, where this bandwidth is considered equivalent to an OSPF cost of 1, specified in Mbits/s. The default is 100Mbit/s (i.e. a link of bandwidth 100Mbit/s or higher will have a cost of 1. Cost of lower bandwidth links will be scaled with reference to this cost).

This configuration setting MUST be consistent across all routers within the OSPF domain.

17.2 OSPF6 area

Area support for OSPFv3 is not yet implemented.

17.3 OSPF6 interface

ipv6 ospf6 cost COST
Sets interface's output cost. Default value depends on the interface bandwidth and on the auto-cost reference bandwidth.

ipv6 ospf6 hello-interval HELLOINTERVAL
Sets interface's Hello Interval. Default 40

ipv6 ospf6 dead-interval DEADINTERVAL
Sets interface's Router Dead Interval. Default value is 40.

ipv6 ospf6 retransmit-interval RETRANSMITINTERVAL
Sets interface's Rxmt Interval. Default value is 5.

ipv6 ospf6 priority PRIORITY
Sets interface's Router Priority. Default value is 1.

ipv6 ospf6 transmit-delay TRANSMITDELAY
Sets interface's Inf-Trans-Delay. Default value is 1.

ipv6 ospf6 network (broadcast|point-to-point)
Set explicitly network type for specified interface.

17.4 Redistribute routes to OSPF6

redistribute static

redistribute connected

redistribute ripng

17.5 Showing OSPF6 information

show ipv6 ospf6 [INSTANCE_ID]
INSTANCE_ID is an optional OSPF instance ID. To see router ID and OSPF instance ID, simply type "show ipv6 ospf6 <cr>".

show ipv6 ospf6 database
This command shows LSA database summary. You can specify the type of LSA.

show ipv6 ospf6 interface
To see OSPF interface configuration like costs.

show ipv6 ospf6 neighbor
Shows state and chosen (Backup) DR of neighbor.

show ipv6 ospf6 request-list A.B.C.D
Shows requestlist of neighbor.

show ipv6 route ospf6
This command shows internal routing table.

show ipv6 ospf6 zebra
Shows state about what is being redistributed between zebra and OSPF6

17.6 OSPF6 Configuration Examples

Example of ospf6d configured on one interface and area:

```
interface eth0
 ipv6 ospf6 instance-id 0
!
router ospf6
 router-id 212.17.55.53
 area 0.0.0.0 range 2001:770:105:2::/64
 interface eth0 area 0.0.0.0
!
```

CHAPTER 18

PIM

PIM – Protocol Independent Multicast

pimd supports pim-sm as well as igmp v2 and v3. pim is vrf aware and can work within the context of vrf's in order to do S,G mrouting.

18.1 Starting and Stopping pimd

The default configuration file name of *pimd*'s is `pimd.conf`. When invoked *pimd* searches directory /etc/frr. If `pimd.conf` is not there then next search current directory.

pimd requires zebra for proper operation. Additionally *pimd* depends on routing properly setup and working in the network that it is working on.

```
# zebra -d
# pimd -d
```

Please note that *zebra* must be invoked before *pimd*.

To stop *pimd* please use:

```
kill `cat /var/run/pimd.pid`
```

Certain signals have special meanings to *pimd*.

Signal	Meaning
SIGUSR1	Rotate the *pimd* logfile
SIGINT SIGTERM	*pimd* sweeps all installed PIM mroutes then terminates gracefully.

pimd invocation options. Common options that can be specified (*Common Invocation Options*).

ip pim rp A.B.C.D A.B.C.D/M
: In order to use pim, it is necessary to configure a RP for join messages to be sent to. Currently the only methodology to do this is via static rp commands. All routers in the pim network must agree on these values.

The first ip address is the RP's address and the second value is the matching prefix of group ranges covered. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim spt-switchover infinity-and-beyond
On the last hop router if it is desired to not switch over to the SPT tree. Configure this command. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim ecmp
If pim has the a choice of ECMP nexthops for a particular RPF, pim will cause S,G flows to be spread out amongst the nexthops. If this command is not specified then the first nexthop found will be used. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim ecmp rebalance
If pim is using ECMP and an interface goes down, cause pim to rebalance all S,G flows across the remaining nexthops. If this command is not configured pim only modifies those S,G flows that were using the interface that went down. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim join-prune-interval (60-600)
Modify the join/prune interval that pim uses to the new value. Time is specified in seconds. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim keep-alive-timer (31-60000)
Modify the time out value for a S,G flow from 31-60000 seconds. 31 seconds is chosen for a lower bound because some hardware platforms cannot see data flowing in better than 30 second chunks. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim packets (1-100)
When processing packets from a neighbor process the number of packets incoming at one time before moving on to the next task. The default value is 3 packets. This command is only useful at scale when you can possibly have a large number of pim control packets flowing. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim register-suppress-time (5-60000)
Modify the time that pim will register suppress a FHR will send register notifications to the kernel. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim send-v6-secondary
When sending pim hello packets tell pim to send any v6 secondary addresses on the interface. This information is used to allow pim to use v6 nexthops in it's decision for RPF lookup. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim ssm prefix-list WORD
Specify a range of group addresses via a prefix-list that forces pim to never do SM over. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip multicast rpf-lookup-mode WORD
Modify how PIM does RPF lookups in the zebra routing table. You can use these choices:

longer-prefix Lookup the RPF in both tables using the longer prefix as a match

lower-distance Lookup the RPF in both tables using the lower distance as a match

mrib-only Lookup in the Multicast RIB only

mrib-then-urib Lookup in the Multicast RIB then the Unicast Rib, returning first found. This is the default value for lookup if this command is not entered

urib-only Lookup in the Unicast Rib only.

18.2 PIM Interface Configuration

PIM interface commands allow you to configure an interface as either a Receiver or a interface that you would like to form pim neighbors on. If the interface is in a vrf, enter the interface command with the vrf keyword at the end.

ip pim bfd
Turns on BFD support for PIM for this interface.

ip pim drpriority (1-4294967295)
Set the DR Priority for the interface. This command is useful to allow the user to influence what node becomes the DR for a lan segment.

ip pim hello (1-180) (1-180)
Set the pim hello and hold interval for a interface.

ip pim sm
Tell pim that we would like to use this interface to form pim neighbors over. Please note we will *not* accept igmp reports over this interface with this command.

ip igmp
Tell pim to receive IGMP reports and Query on this interface. The default version is v3. This command is useful on the LHR.

ip igmp query-interval (1-1800)
Set the IGMP query interval that PIM will use.

ip igmp query-max-response-time (10-250)
Set the IGMP query response timeout value. If an report is not returned in the specified time we will assume the S,G or *,G has timed out.

ip igmp version (2-3)
Set the IGMP version used on this interface. The default value is 3.

ip multicat boundary oil WORD
Set a pim multicast boundary, based upon the WORD prefix-list. If a pim join or IGMP report is received on this interface and the Group is denied by the prefix-list, PIM will ignore the join or report.

18.3 PIM Multicast RIB insertion:

In order to influence Multicast RPF lookup, it is possible to insert into zebra routes for the Multicast RIB. These routes are only used for RPF lookup and will not be used by zebra for insertion into the kernel *or* for normal rib processing. As such it is possible to create weird states with these commands. Use with caution. Most of the time this will not be necessary.

ip mroute A.B.C.D/M A.B.C.D (1-255)
Insert into the Multicast Rib Route A.B.C.D/M with specified nexthop. The distance can be specified as well if desired.

ip mroute A.B.C.D/M INTERFACE (1-255)
Insert into the Multicast Rib Route A.B.C.D/M using the specified INTERFACE. The distance can be specified as well if desired.

18.4 Show PIM Information

All PIM show commands are vrf aware and typically allow you to insert a specified vrf command if information is desired about a specific vrf. If no vrf is specified then the default vrf is assumed. Finally the special keyword 'all'

allows you to look at all vrfs for the command. Naming a vrf 'all' will cause great confusion.

show ip multicast
Display various information about the interfaces used in this pim instance.

show ip mroute
Display information about installed into the kernel S,G mroutes.

show ip mroute count
Display information about installed into the kernel S,G mroutes and in addition display data about packet flow for the mroutes.

show ip pim assert
Display information about asserts in the PIM system for S,G mroutes.

show ip pim assert-internal
Display internal assert state for S,G mroutes

show ip pim assert-metric
Display metric information about assert state for S,G mroutes

show ip pim assert-winner-metric
Display winner metric for assert state for S,G mroutes

show ip pim group-type
Display SSM group ranges.

show ip pim interface
Display information about interfaces PIM is using.

show ip pim join
Display information about PIM joins received.

show ip pim local-membership
Display information about PIM interface local-membership.

show ip pim neighbor
Display information about PIM neighbors.

show ip pim nexthop
Display information about pim nexthops that are being used.

show ip pim nexthop-lookup
Display information about a S,G pair and how the RPF would be chosen. This is especially useful if there are ECMP's available from the RPF lookup.

show ip pim rp-info
Display information about RP's that are configured on this router.

show ip pim rpf
Display information about currently being used S,G's and their RPF lookup information. Additionally display some statistics about what has been happening on the router.

show ip pim secondary
Display information about an interface and all the secondary addresses associated with it.

show ip pim state
Display information about known S,G's and incoming interface as well as the OIL and how they were chosen.

show ip pim upstream
Display upstream information about a S,G mroute.

show ip pim upstream-join-desired
Display upstream information for S,G's and if we desire to join the multicast tree

show ip pim upstream-rpf
Display upstream information for S,G's and the RPF data associated with them.

show ip rpf
Display the multicast RIB created in zebra.

18.5 PIM Debug Commands

The debugging subsystem for PIM behaves in accordance with how FRR handles debugging. You can specify debugging at the enable CLI mode as well as the configure CLI mode. If you specify debug commands in the configuration cli mode, the debug commands can be persistent across restarts of the FRR pimd if the config was written out.

debug pim events
This turns on debugging for PIM system events. Especially timers.

debug pim nht
This turns on debugging for PIM nexthop tracking. It will display information about RPF lookups and information about when a nexthop changes.

debug pim packet-dump
This turns on an extraordinary amount of data. Each pim packet sent and received is dumped for debugging purposes. This should be considered a developer only command.

debug pim packets
This turns on information about packet generation for sending and about packet handling from a received packet.

debug pim trace
This traces pim code and how it is running.

debug pim zebra
This gathers data about events from zebra that come up through the ZAPI.

CHAPTER 19

PBR

PBR is Policy Based Routing. This implementation supports a very simple interface to allow admins to influence routing on their router. At this time you can only match on destination and source prefixes for an incoming interface. At this point in time, this implementation will only work on Linux.

19.1 Starting PBR

Default configuration file for *pbrd* is `pbrd.conf`. The typical location of `pbrd.conf` is /etc/frr/pbrd.conf.

If the user is using integrated config, then `pbrd.conf` need not be present and the `frr.conf` is read instead.

PBR supports all the common FRR daemon start options which are documented elsewhere.

19.2 Nexthop Groups

Nexthop groups are a way to encapsulate ECMP information together. It's a listing of ECMP nexthops used to forward packets for when a pbr-map is matched.

`nexthop-group NAME`
: Create a nexthop-group with an associated NAME. This will put you into a sub-mode where you can specify individual nexthops. To exit this mode type exit or end as per normal conventions for leaving a sub-mode.

`nexthop [A.B.C.D|X:X::X:XX] [interface] [nexthop-vrf NAME]`
: Create a v4 or v6 nexthop. All normal rules for creating nexthops that you are used to are allowed here. The syntax was intentionally kept the same as creating nexthops as you would for static routes.

19.3 PBR Maps

PBR maps are a way to group policies that we would like to apply to individual interfaces. These policies when applied are matched against incoming packets. If matched the nexthop-group or nexthop is used to forward the packets to the

end destination

pbr-map NAME seq (1-1000)
Create a pbr-map with NAME and sequence number specified. This command puts you into a new submode for pbr-map specification. To exit this mode type exit or end as per normal conventions for leaving a sub-mode.

match src-ip PREFIX
When a incoming packet matches the source prefix specified, take the packet and forward according to the nexthops specified. This command accepts both v4 and v6 prefixes. This command is used in conjunction of the `match dst-ip PREFIX` command for matching.

match dst-ip PREFIX
When a incoming packet matches the destination prefix specified, take the packet and forward according to the nexthops specified. This command accepts both v4 and v6 prefixes. This command is used in conjuction of the `match src-ip PREFIX` command for matching.

set nexthop-group NAME
Use the nexthop-group NAME as the place to forward packets when the match commands have matched a packet.

set nexthop [A.B.C.D|X:X::X:XX] [interface] [nexthop-vrf NAME]
Use this individual nexthop as the place to forward packets when the match commands have matched a packet.

19.4 PBR Policy

After you have specified a PBR map, in order for it to be turned on, you must apply the PBR map to an interface. This policy application to an interface causes the policy to be installed into the kernel.

pbr-policy NAME
This command is available under interface sub-mode. This turns on the PBR map NAME and allows it to work properly.

19.5 PBR Details

Under the covers a PBR map is translated into two separate constructs in the Linux kernel.

The PBR map specified creates a *ip rule* ... that is inserted into the Linux kernel that points to a table to use for forwarding once the rule matches.

The creation of a nexthop or nexthop-group is translated to a default route in a table with the nexthops specified as the nexthops for the default route.

CHAPTER 20

RIP

RIP – Routing Information Protocol is widely deployed interior gateway protocol. RIP was developed in the 1970s at Xerox Labs as part of the XNS routing protocol. RIP is a *distance-vector* protocol and is based on the *Bellman-Ford* algorithms. As a distance-vector protocol, RIP router send updates to its neighbors periodically, thus allowing the convergence to a known topology. In each update, the distance to any given network will be broadcast to its neighboring router.

ripd supports RIP version 2 as described in RFC2453 and RIP version 1 as described in RFC1058.

20.1 Starting and Stopping ripd

The default configuration file name of *ripd*'s is `ripd.conf`. When invocation *ripd* searches directory /etc/frr. If `ripd.conf` is not there next search current directory.

RIP uses UDP port 520 to send and receive RIP packets. So the user must have the capability to bind the port, generally this means that the user must have superuser privileges. RIP protocol requires interface information maintained by *zebra* daemon. So running *zebra* is mandatory to run *ripd*. Thus minimum sequence for running RIP is like below:

```
# zebra -d
# ripd -d
```

Please note that *zebra* must be invoked before *ripd*.

To stop *ripd*. Please use:: kill *cat /var/run/ripd.pid*

Certain signals have special meanings to *ripd*.

Signal	Action
`SIGHUP`	Reload configuration file `ripd.conf`. All configurations are reset. All routes learned so far are cleared and removed from routing table.
`SIGUSR1`	Rotate the *ripd* logfile.
`SIGINT` `SIGTERM`	Sweep all installed routes and gracefully terminate.

ripd invocation options. Common options that can be specified (*Common Invocation Options*).

-r, --retain
When the program terminates, retain routes added by *ripd*.

20.1.1 RIP netmask

The netmask features of *ripd* support both version 1 and version 2 of RIP. Version 1 of RIP originally contained no netmask information. In RIP version 1, network classes were originally used to determine the size of the netmask. Class A networks use 8 bits of mask, Class B networks use 16 bits of masks, while Class C networks use 24 bits of mask. Today, the most widely used method of a network mask is assigned to the packet on the basis of the interface that received the packet. Version 2 of RIP supports a variable length subnet mask (VLSM). By extending the subnet mask, the mask can be divided and reused. Each subnet can be used for different purposes such as large to middle size LANs and WAN links. FRR *ripd* does not support the non-sequential netmasks that are included in RIP Version 2.

In a case of similar information with the same prefix and metric, the old information will be suppressed. Ripd does not currently support equal cost multipath routing.

20.2 RIP Configuration

router rip
The *router rip* command is necessary to enable RIP. To disable RIP, use the *no router rip* command. RIP must be enabled before carrying out any of the RIP commands.

no router rip
Disable RIP.

network NETWORK

no network NETWORK
Set the RIP enable interface by NETWORK. The interfaces which have addresses matching with NETWORK are enabled.

This group of commands either enables or disables RIP interfaces between certain numbers of a specified network address. For example, if the network for 10.0.0.0/24 is RIP enabled, this would result in all the addresses from 10.0.0.0 to 10.0.0.255 being enabled for RIP. The *no network* command will disable RIP for the specified network.

network IFNAME

no network IFNAME
Set a RIP enabled interface by IFNAME. Both the sending and receiving of RIP packets will be enabled on the port specified in the *network ifname* command. The *no network ifname* command will disable RIP on the specified interface.

neighbor A.B.C.D

no neighbor A.B.C.D
Specify RIP neighbor. When a neighbor doesn't understand multicast, this command is used to specify neighbors. In some cases, not all routers will be able to understand multicasting, where packets are sent to a network or a group of addresses. In a situation where a neighbor cannot process multicast packets, it is necessary to establish a direct link between routers. The neighbor command allows the network administrator to specify a router as a RIP neighbor. The *no neighbor a.b.c.d* command will disable the RIP neighbor.

Below is very simple RIP configuration. Interface *eth0* and interface which address match to *10.0.0.0/8* are RIP enabled.

```
!
router rip
 network 10.0.0.0/8
 network eth0
!
```

passive-interface (IFNAME|default)

no passive-interface IFNAME
This command sets the specified interface to passive mode. On passive mode interface, all receiving packets are processed as normal and ripd does not send either multicast or unicast RIP packets except to RIP neighbors specified with *neighbor* command. The interface may be specified as *default* to make ripd default to passive on all interfaces.

The default is to be passive on all interfaces.

ip split-horizon

no ip split-horizon
Control split-horizon on the interface. Default is *ip split-horizon*. If you don't perform split-horizon on the interface, please specify *no ip split-horizon*.

20.3 RIP Version Control

RIP can be configured to send either Version 1 or Version 2 packets. The default is to send RIPv2 while accepting both RIPv1 and RIPv2 (and replying with packets of the appropriate version for REQUESTS / triggered updates). The version to receive and send can be specified globally, and further overridden on a per-interface basis if needs be for send and receive separately (see below).

It is important to note that RIPv1 cannot be authenticated. Further, if RIPv1 is enabled then RIP will reply to REQUEST packets, sending the state of its RIP routing table to any remote routers that ask on demand. For a more detailed discussion on the security implications of RIPv1 see *RIP Authentication*.

version VERSION
Set RIP version to accept for reads and send. `VERSION` can be either 1 or 1.

Disabling RIPv1 by specifying version 2 is STRONGLY encouraged, *RIP Authentication*. This may become the default in a future release.

Default: Send Version 2, and accept either version.

no version
Reset the global version setting back to the default.

ip rip send version VERSION
VERSION can be `1`, `2`, or `1 2`.

This interface command overrides the global rip version setting, and selects which version of RIP to send packets with, for this interface specifically. Choice of RIP Version 1, RIP Version 2, or both versions. In the latter case, where `1 2` is specified, packets will be both broadcast and multicast.

Default: Send packets according to the global version (version 2)

ip rip receive version VERSION
VERSION can be `1`, `2`, or `1 2`.

This interface command overrides the global rip version setting, and selects which versions of RIP packets will be accepted on this interface. Choice of RIP Version 1, RIP Version 2, or both.

Default: Accept packets according to the global setting (both 1 and 2).

20.4 How to Announce RIP route

redistribute kernel

redistribute kernel metric (0-16)

redistribute kernel route-map ROUTE-MAP

no redistribute kernel
redistribute kernel redistributes routing information from kernel route entries into the RIP tables. *no redistribute kernel* disables the routes.

redistribute static

redistribute static metric (0-16)

redistribute static route-map ROUTE-MAP

no redistribute static
redistribute static redistributes routing information from static route entries into the RIP tables. *no redistribute static* disables the routes.

redistribute connected

redistribute connected metric (0-16)

redistribute connected route-map ROUTE-MAP

no redistribute connected
Redistribute connected routes into the RIP tables. *no redistribute connected* disables the connected routes in the RIP tables. This command redistribute connected of the interface which RIP disabled. The connected route on RIP enabled interface is announced by default.

redistribute ospf

redistribute ospf metric (0-16)

redistribute ospf route-map ROUTE-MAP

no redistribute ospf
redistribute ospf redistributes routing information from ospf route entries into the RIP tables. *no redistribute ospf* disables the routes.

redistribute bgp

redistribute bgp metric (0-16)

redistribute bgp route-map ROUTE-MAP

no redistribute bgp
redistribute bgp redistributes routing information from bgp route entries into the RIP tables. *no redistribute bgp* disables the routes.

If you want to specify RIP only static routes:

default-information originate

route A.B.C.D/M

no route A.B.C.D/M
This command is specific to FRR. The *route* command makes a static route only inside RIP. This command should be used only by advanced users who are particularly knowledgeable about the RIP protocol. In most cases, we recommend creating a static route in FRR and redistributing it in RIP using *redistribute static*.

20.5 Filtering RIP Routes

RIP routes can be filtered by a distribute-list.

distribute-list ACCESS_LIST DIRECT IFNAME
You can apply access lists to the interface with a *distribute-list* command. ACCESS_LIST is the access list name. DIRECT is `in` or `out`. If DIRECT is `in` the access list is applied to input packets.

The *distribute-list* command can be used to filter the RIP path. *distribute-list* can apply access-lists to a chosen interface. First, one should specify the access-list. Next, the name of the access-list is used in the distribute-list command. For example, in the following configuration `eth0` will permit only the paths that match the route 10.0.0.0/8

```
!
router rip
 distribute-list private in eth0
!
access-list private permit 10 10.0.0.0/8
access-list private deny any
!
```

distribute-list can be applied to both incoming and outgoing data.

distribute-list prefix PREFIX_LIST (in|out) IFNAME
You can apply prefix lists to the interface with a *distribute-list* command. PREFIX_LIST is the prefix list name. Next is the direction of `in` or `out`. If DIRECT is `in` the access list is applied to input packets.

20.6 RIP Metric Manipulation

RIP metric is a value for distance for the network. Usually *ripd* increment the metric when the network information is received. Redistributed routes' metric is set to 1.

default-metric (1-16)

no default-metric (1-16)
This command modifies the default metric value for redistributed routes. The default value is 1. This command does not affect connected route even if it is redistributed by *redistribute connected*. To modify connected route's metric value, please use `redistribute connected metric` or *route-map*. *offset-list* also affects connected routes.

offset-list ACCESS-LIST (in|out)

offset-list ACCESS-LIST (in|out) IFNAME

20.7 RIP distance

Distance value is used in zebra daemon. Default RIP distance is 120.

distance (1-255)

no distance (1-255)
Set default RIP distance to specified value.

distance (1-255) A.B.C.D/M

no distance (1-255) A.B.C.D/M
Set default RIP distance to specified value when the route's source IP address matches the specified prefix.

distance (1-255) A.B.C.D/M ACCESS-LIST

no distance (1-255) A.B.C.D/M ACCESS-LIST
Set default RIP distance to specified value when the route's source IP address matches the specified prefix and the specified access-list.

20.8 RIP route-map

Usage of *ripd*'s route-map support.

Optional argument route-map MAP_NAME can be added to each *redistribute* statement.

```
redistribute static [route-map MAP_NAME]
redistribute connected [route-map MAP_NAME]
.....
```

Cisco applies route-map _before_ routes will exported to rip route table. In current FRR's test implementation, *ripd* applies route-map after routes are listed in the route table and before routes will be announced to an interface (something like output filter). I think it is not so clear, but it is draft and it may be changed at future.

Route-map statement (*Route Maps*) is needed to use route-map functionality.

match interface WORD
This command match to incoming interface. Notation of this match is different from Cisco. Cisco uses a list of interfaces - NAME1 NAME2 ... NAMEN. Ripd allows only one name (maybe will change in the future). Next - Cisco means interface which includes next-hop of routes (it is somewhat similar to "ip next-hop" statement). Ripd means interface where this route will be sent. This difference is because "next-hop" of same routes which sends to different interfaces must be different. Maybe it'd be better to made new matches - say "match interface-out NAME" or something like that.

match ip address WORD

match ip address prefix-list WORD
Match if route destination is permitted by access-list.

match ip next-hop WORD

match ip next-hop prefix-list WORD
Match if route next-hop (meaning next-hop listed in the rip route-table as displayed by "show ip rip") is permitted by access-list.

match metric (0-4294967295)
This command match to the metric value of RIP updates. For other protocol compatibility metric range is shown as (0-4294967295). But for RIP protocol only the value range (0-16) make sense.

set ip next-hop A.B.C.D
This command set next hop value in RIPv2 protocol. This command does not affect RIPv1 because there is no next hop field in the packet.

set metric (0-4294967295)
Set a metric for matched route when sending announcement. The metric value range is very large for compatibility with other protocols. For RIP, valid metric values are from 1 to 16.

20.9 RIP Authentication

RIPv2 allows packets to be authenticated via either an insecure plain text password, included with the packet, or via a more secure MD5 based HMAC (keyed-Hashing for Message AuthentiCation), RIPv1 can not be authenticated at all, thus when authentication is configured *ripd* will discard routing updates received via RIPv1 packets.

However, unless RIPv1 reception is disabled entirely, *RIP Version Control*, RIPv1 REQUEST packets which are received, which query the router for routing information, will still be honoured by *ripd*, and *ripd* WILL reply to such packets. This allows *ripd* to honour such REQUESTs (which sometimes is used by old equipment and very simple devices to bootstrap their default route), while still providing security for route updates which are received.

In short: Enabling authentication prevents routes being updated by unauthenticated remote routers, but still can allow routes (I.e. the entire RIP routing table) to be queried remotely, potentially by anyone on the internet, via RIPv1.

To prevent such unauthenticated querying of routes disable RIPv1, *RIP Version Control*.

ip rip authentication mode md5

no ip rip authentication mode md5
Set the interface with RIPv2 MD5 authentication.

ip rip authentication mode text

no ip rip authentication mode text
Set the interface with RIPv2 simple password authentication.

ip rip authentication string STRING

no ip rip authentication string STRING
RIP version 2 has simple text authentication. This command sets authentication string. The string must be shorter than 16 characters.

ip rip authentication key-chain KEY-CHAIN

no ip rip authentication key-chain KEY-CHAIN
Specify Keyed MD5 chain.

```
!
key chain test
 key 1
  key-string test
!
interface eth1
 ip rip authentication mode md5
 ip rip authentication key-chain test
!
```

20.10 RIP Timers

timers basic UPDATE TIMEOUT GARBAGE
RIP protocol has several timers. User can configure those timers' values by *timers basic* command.

The default settings for the timers are as follows:

- The update timer is 30 seconds. Every update timer seconds, the RIP process is awakened to send an unsolicited Response message containing the complete routing table to all neighboring RIP routers.

- The timeout timer is 180 seconds. Upon expiration of the timeout, the route is no longer valid; however, it is retained in the routing table for a short time so that neighbors can be notified that the route has been dropped.
- The garbage collect timer is 120 seconds. Upon expiration of the garbage-collection timer, the route is finally removed from the routing table.

The `timers basic` command allows the the default values of the timers listed above to be changed.

no timers basic
The *no timers basic* command will reset the timers to the default settings listed above.

20.11 Show RIP Information

To display RIP routes.

show ip rip
Show RIP routes.

The command displays all RIP routes. For routes that are received through RIP, this command will display the time the packet was sent and the tag information. This command will also display this information for routes redistributed into RIP.

show ip rip status
The command displays current RIP status. It includes RIP timer, filtering, version, RIP enabled interface and RIP peer information.

```
ripd> **show ip rip status**
Routing Protocol is "rip"
  Sending updates every 30 seconds with +/-50%, next due in 35 seconds
  Timeout after 180 seconds, garbage collect after 120 seconds
  Outgoing update filter list for all interface is not set
  Incoming update filter list for all interface is not set
  Default redistribution metric is 1
  Redistributing: kernel connected
  Default version control: send version 2, receive version 2
    Interface  Send  Recv
  Routing for Networks:
    eth0
    eth1
    1.1.1.1
    203.181.89.241
  Routing Information Sources:
    Gateway    BadPackets BadRoutes  Distance Last Update
```

20.12 RIP Debug Commands

Debug for RIP protocol.

debug rip events
Shows RIP events. Sending and receiving packets, timers, and changes in interfaces are events shown with *ripd*.

debug rip packet
Shows display detailed information about the RIP packets. The origin and port number of the packet as well as a packet dump is shown.

debug rip zebra
: This command will show the communication between *ripd* and *zebra*. The main information will include addition and deletion of paths to the kernel and the sending and receiving of interface information.

show debugging rip
: Shows all information currently set for ripd debug.

CHAPTER 21

RIPng

ripngd supports the RIPng protocol as described in **RFC 2080**. It's an IPv6 reincarnation of the RIP protocol.

21.1 Invoking ripngd

There are no *ripngd* specific invocation options. Common options can be specified (*Common Invocation Options*).

21.2 ripngd Configuration

Currently ripngd supports the following commands:

router ripng
: Enable RIPng.

flush_timer TIME
: Set flush timer.

network NETWORK
: Set RIPng enabled interface by NETWORK.

network IFNAME
: Set RIPng enabled interface by IFNAME.

route NETWORK
: Set RIPng static routing announcement of NETWORK.

router zebra
: This command is the default and does not appear in the configuration. With this statement, RIPng routes go to the *zebra* daemon.

21.3 ripngd Terminal Mode Commands

show ip ripng

show debugging ripng

debug ripng events

debug ripng packet

debug ripng zebra

21.4 ripngd Filtering Commands

distribute-list ACCESS_LIST (in|out) IFNAME
: You can apply an access-list to the interface using the *distribute-list* command. ACCESS_LIST is an access-list name. *direct* is `in` or `out`. If *direct* is `in`, the access-list is applied only to incoming packets.:

```
distribute-list local-only out sit1
```

*** SHARP ***

SHARP Super Happy Advanced Routing Process. This daemon is useful for the testing of FRR itself as well as useful for creation of Proof of Concept labs.

CHAPTER 22

Starting SHARP

Default configuration file for *sharpd* is `sharpd.conf`. The typical location of `sharpd.conf` is /etc/frr/sharpd.conf.

If the user is using integrated config, then `sharpd.conf` need not be present and the `frr.conf` is read instead.

SHARP supports all the common FRR daemon start options which are documented elsewhere.

CHAPTER 23

USING SHARP

All sharp commands are under the enable node and proceeded by the SHARP keyword. There are currently no permenent sharp commands for configuration.

..index:: sharp install ..clicmd:: sharp install routes A.B.C.D nexthop E.F.G.H (1-1000000)

Install up to a million /32 routes starting at A.B.C.D with specified nexthop E.F.G.H. The nexthop is a NEXTHOP_TYPE_IPV4 and must be reachable to be installed into the kernel. The routes are installed into zebra as ZEBRA_ROUTE_SHARP and can be used as part of a normal route redistribution. Route installation time is noted in the debug log and upon zebra successful installation into the kernel and sharp receiving the notification of all route installs the success will be noted in the debug log as well.

..index:: sharp remove ..clicmd:: sharp remove routes A.B.C.D (1-1000000)

Remove up 1000000 million /32 routes starting at A.B.C.D. The routes are removed from zebra. Route deletion start is noted in the debug log and when all routes have been successfully deleted the debug log will be updated with this information as well.

..index:: sharp label ..clicmd:: sharp label <ipv4|ipv6> vrf NAME label (0-1000000)

Install a label into the kernel that causes the specified vrf NAME table to be used for pop and forward operations when the specified label is seen.

..index:: sharp watch ..clicmd: sharp watch nexthop <A.B.C.D|X:X::X:X>

Instruct zebra to monitor and notify sharp when the specified nexthop is changed. The notification from zebra is written into the debug log.

CHAPTER 24

VNC and VNC-GW

This chapter describes how to use VNC (Virtual Network Control) services, including NVA (Network Virtualization Authority) and VNC-GW (VNC Gateway) functions. Background information on NVAs, NVE (Network Virtualization Edge) s, UN (Underlay Network) s, and VN (Virtual Network) is available from the IETF. VNC-GW s support the import/export of routing information between VNC and CE (customer edge) routers operating within a VN. Both IP/Layer 3 (L3) VNs, and IP with Ethernet/Layer 2 (L2) VNs are supported.

BGP, with IP VPNs and Tunnel Encapsulation, is used to distribute VN information between NVAs. BGP based IP VPN support is defined in **RFC 4364**, and **RFC 4659**. Encapsulation information is provided via the Tunnel Encapsulation Attribute, **RFC 5512**.

The protocol that is used to communicate routing and Ethernet / Layer 2 (L2) forwarding information between NVAs and NVEs is referred to as the Remote Forwarder Protocol (RFP). *OpenFlow* is an example RFP. Specific RFP implementations may choose to implement either a *hard-state* or *soft-state* prefix and address registration model. To support a *soft-state* refresh model, a *lifetime* in seconds is associated with all registrations and responses.

The chapter also provides sample configurations for basic example scenarios.

24.1 Configuring VNC

Virtual Network Control (VNC) service configuration commands appear in the *router bgp* section of the BGPD configuration file (*BGP Configuration Examples*). The commands are broken down into the following areas:

- *General VNC* configuration applies to general VNC operation and is primarily used to control the method used to advertise tunnel information.
- *Remote Forwarder Protocol (RFP)* configuration relates to the protocol used between NVAs and NVEs.
- *VNC Defaults* provides default parameters for registered NVEs.
- *VNC NVE Group* provides for configuration of a specific set of registered NVEs and overrides default parameters.
- *Redistribution* and *Export* control VNC-GW operation, i.e., the import/export of routing information between VNC and customer edge routers (CE s) operating within a VN.

24.1.1 General VNC Configuration

24.1.2 RFP Related Configuration

The protocol that is used to communicate routing and Ethernet / L2 forwarding information between NVAs and NVEs is referred to as the Remote Forwarder Protocol (RFP). Currently, only a simple example RFP is included in FRR. Developers may use this example as a starting point to integrate FRR with an RFP of their choosing, e.g., *OpenFlow*. The example code includes the following sample configuration:

rfp example-config-value VALUE

This is a simple example configuration parameter included as part of the RFP example code. VALUE must be in the range of 0 to 4294967295.

24.1.3 VNC Defaults Configuration

The VNC Defaults section allows the user to specify default values for configuration parameters for all registered NVEs. Default values are overridden by *VNC NVE Group Configuration*.

vnc defaults

Enter VNC configuration mode for specifying VNC default behaviors. Use *exit-vnc* to leave VNC configuration mode. *vnc defaults* is optional.

```
vnc defaults
... various VNC defaults
exit-vnc
```

These are the statements that can appear between `vnc defaults` and `exit-vnc`. Documentation for these statements is given in *VNC NVE Group Configuration*.

- `rt import RT-LIST`
- `rt export RT-LIST`
- `rt both RT-LIST`
- `rd ROUTE-DISTINGUISHER`
- `l2rd NVE-ID-VALUE`
- `response-lifetime LIFETIME|infinite`
- `export bgp|zebra route-map MAP-NAME`
- `export bgp|zebra no route-map`

exit-vnc
Exit VNC configuration mode.

24.1.4 VNC NVE Group Configuration

A NVE Group corresponds to a specific set of NVEs. A Client NVE is assigned to an NVE Group based on whether there is a match for either its virtual or underlay network address against the VN and/or UN address prefixes specified in the NVE Group definition. When an NVE Group definition specifies both VN and UN address prefixes, then an NVE must match both prefixes in order to be assigned to the NVE Group. In the event that multiple NVE Groups match based on VN and/or UN addresses, the NVE is assigned to the first NVE Group listed in the configuration. If an NVE is not assigned to an NVE Group, its messages will be ignored.

Configuration values specified for an NVE group apply to all member NVEs and override configuration values specified in the VNC Defaults section.

At least one 'nve-group' is mandatory for useful VNC operation.

vnc nve-group NAME
Enter VNC configuration mode for defining the NVE group *name*. Use *exit* or *exit-vnc* to exit group configuration mode.

```
vnc nve-group group1
... configuration commands
exit-vnc
```

no vnc nve-group NAME
Delete the NVE group named *name*.

The following statements are valid in an NVE group definition:

l2rd NVE-ID-VALUE
Set the value used to distinguish NVEs connected to the same physical Ethernet segment (i.e., at the same location)[1].

The nve-id subfield may be specified as either a literal value in the range 1-255, or it may be specified as *auto:vn*, which means to use the least-significant octet of the originating NVE's VN address.

prefix vn|un A.B.C.D/M|X:X::X:X/M
Specify the matching prefix for this NVE group by either virtual-network address (*vn*) or underlay-network address (*un*). Either or both virtual-network and underlay-network prefixes may be specified. Subsequent virtual-network or underlay-network values within a *vnc nve-group exit-vnc* block override their respective previous values.

These prefixes are used only for determining assignments of NVEs to NVE Groups.

rd ROUTE-DISTINGUISHER
Specify the route distinguisher for routes advertised via BGP VPNs. The route distinguisher must be in one of these forms:

- `IPv4-address:two-byte-integer`
- `four-byte-autonomous-system-number:two-byte-integer`
- `two-byte-autonomous-system-number:four-byte-integer`
- `auto:vn:two-byte-integer`

Routes originated by NVEs in the NVE group will use the group's specified *route-distinguisher* when they are advertised via BGP. If the *auto* form is specified, it means that a matching NVE has its RD set to `rd_type=IP=1:IPv4-address=VN-address:two-byte-integer`, for IPv4 VN addresses and `rd_type=IP=1:IPv4-address=Last-four-bytes-of-VN-address:two-byte-integer`, for IPv6 VN addresses.

If the NVE group definition does not specify a *route-distinguisher*, then the default *route-distinguisher* is used. If neither a group nor a default *route-distinguisher* is configured, then the advertised RD is set to `two-byte-autonomous-system-number=0:four-byte-integer=0`.

response-lifetime LIFETIME|infinite
Specify the response lifetime, in seconds, to be included in RFP response messages sent to NVEs. If the value 'infinite' is given, an infinite lifetime will be used.

[1] The nve-id is carried in the route distinguisher. It is the second octet of the eight-octet route distinguisher generated for Ethernet / L2 advertisements. The first octet is a constant 0xFF, and the third through eighth octets are set to the L2 ethernet address being advertised.

Note that this parameter is not the same as the lifetime supplied by NVEs in RFP registration messages. This parameter does not affect the lifetime value attached to routes sent by this server via BGP.

If the NVE group definition does not specify a *response-lifetime*, the default *response-lifetime* will be used. If neither a group nor a default *response-lifetime* is configured, the value 3600 will be used. The maximum response lifetime is 2147483647.

rt export RT-LIST

rt import RT-LIST

rt both RT-LIST

Specify route target import and export lists. *rt-list* is a space-separated list of route targets, each element of which is in one of the following forms:

- `IPv4-address:two-byte-integer`
- `four-byte-autonomous-system-number:two-byte-integer`
- `two-byte-autonomous-system-number:four-byte-integer`

The first form, *rt export*, specifies an *export rt-list*. The *export rt-list* will be attached to routes originated by NVEs in the NVE group when they are advertised via BGP. If the NVE group definition does not specify an *export rt-list*, then the default *export rt-list* is used. If neither a group nor a default *export rt-list* is configured, then no RT list will be sent; in turn, these routes will probably not be processed by receiving NVAs.

The second form, *rt import* specifies an *import rt-list*, which is a filter for incoming routes. In order to be made available to NVEs in the group, incoming BGP VPN routes must have RT lists that have at least one route target in common with the group's *import rt-list*.

If the NVE group definition does not specify an import filter, then the default *import rt-list* is used. If neither a group nor a default *import rt-list* is configured, there can be no RT intersections when receiving BGP routes and therefore no incoming BGP routes will be processed for the group.

The third, *rt both*, is a shorthand way of specifying both lists simultaneously, and is equivalent to *rt export 'rt-list'* followed by *rt import 'rt-list'*.

export bgp|zebra route-map MAP-NAME

Specify that the named route-map should be applied to routes being exported to bgp or zebra. This parameter is used in conjunction with *Configuring Export of Routes to Other Routing Protocols*. This item is optional.

export bgp|zebra no route-map

Specify that no route-map should be applied to routes being exported to bgp or zebra. This parameter is used in conjunction with *Configuring Export of Routes to Other Routing Protocols*. This item is optional.

export bgp|zebra ipv4|ipv6 prefix-list LIST-NAME

Specify that the named prefix-list filter should be applied to routes being exported to bgp or zebra. Prefix-lists for ipv4 and ipv6 are independent of each other. This parameter is used in conjunction with *Configuring Export of Routes to Other Routing Protocols*. This item is optional.

export bgp|zebra no ipv4|ipv6 prefix-list

Specify that no prefix-list filter should be applied to routes being exported to bgp or zebra. This parameter is used in conjunction with *Configuring Export of Routes to Other Routing Protocols*. This item is optional.

24.1.5 VNC L2 Group Configuration

The route targets advertised with prefixes and addresses registered by an NVE are determined based on the NVE's associated VNC NVE Group Configuration, *VNC NVE Group Configuration*. Layer 2 (L2) Groups are used to override the route targets for an NVE's Ethernet registrations based on the Logical Network Identifier and label value. A Logical Network Identifier is used to uniquely identify a logical Ethernet segment and is conceptually similar to the Ethernet

Segment Identifier defined in **RFC 7432**. Both the Logical Network Identifier and Label are passed to VNC via RFP prefix and address registration.

Note that a corresponding NVE group configuration must be present, and that other NVE associated configuration information, notably RD, is not impacted by L2 Group Configuration.

vnc l2-group NAME
Enter VNC configuration mode for defining the L2 group *name*. Use *exit* or *exit-vnc* to exit group configuration mode.

```
vnc l2-group group1
  ... configuration commands
exit-vnc
```

no vnc l2-group NAME
Delete the L2 group named *name*.

The following statements are valid in a L2 group definition:

logical-network-id VALUE
Define the Logical Network Identifier with a value in the range of 0-4294967295 that identifies the logical Ethernet segment.

labels LABEL-LIST

no labels LABEL-LIST
Add or remove labels associated with the group. *label-list* is a space separated list of label values in the range of 0-1048575.

rt import RT-TARGET

rt export RT-TARGET

rt both RT-TARGET
Specify the route target import and export value associated with the group. A complete definition of these parameters is given above, *VNC NVE Group Configuration*.

24.1.6 Configuring Redistribution of Routes from Other Routing Protocols

Routes from other protocols (including BGP) can be provided to VNC (both for RFP and for redistribution via BGP) from three sources: the zebra kernel routing process; directly from the main (default) unicast BGP RIB; or directly from a designated BGP unicast exterior routing RIB instance.

The protocol named in the *vnc redistribute* command indicates the route source: *bgp-direct* routes come directly from the main (default) unicast BGP RIB and are available for RFP and are redistributed via BGP; *bgp-direct-to-nve-groups* routes come directly from a designated BGP unicast routing RIB and are made available only to RFP; and routes from other protocols come from the zebra kernel routing process. Note that the zebra process does not need to be active if only *bgp-direct* or *bgp-direct-to-nve-groups* routes are used.

zebra routes

Routes originating from protocols other than BGP must be obtained via the zebra routing process. Redistribution of these routes into VNC does not support policy mechanisms such as prefix-lists or route-maps.

bgp-direct routes

bgp-direct redistribution supports policy via prefix lists and route-maps. This policy is applied to incoming original unicast routes before the redistribution translations (described below) are performed.

Redistribution of *bgp-direct* routes is performed in one of three possible modes: *plain*, *nve-group*, or *resolve-nve*. The default mode is *plain*. These modes indicate the kind of translations applied to routes before they are added to the VNC RIB.

In *plain* mode, the route's next hop is unchanged and the RD is set based on the next hop. For *bgp-direct* redistribution, the following translations are performed:

- The VN address is set to the original unicast route's next hop address.
- The UN address is NOT set. (VN->UN mapping will occur via ENCAP route or attribute, based on *vnc advertise-un-method* setting, generated by the RFP registration of the actual NVE)
- The RD is set to as if auto:vn:0 were specified (i.e., *rd_type=IP=1*:*IPv4-address=VN-address*:*two-byte-integer=0*)
- The RT list is included in the extended community list copied from the original unicast route (i.e., it must be set in the original unicast route).

In *nve-group* mode, routes are registered with VNC as if they came from an NVE in the nve-group designated in the *vnc redistribute nve-group* command. The following translations are performed:

- The next hop/VN address is set to the VN prefix configured for the redistribute nve-group.
- The UN address is set to the UN prefix configured for the redistribute nve-group.
- The RD is set to the RD configured for the redistribute nve-group.
- The RT list is set to the RT list configured for the redistribute nve-group. If *bgp-direct* routes are being redistributed, any extended communities present in the original unicast route will also be included.

In *resolve-nve* mode, the next hop of the original BGP route is typically the address of an NVE connected router (CE) connected by one or more NVEs. Each of the connected NVEs will register, via RFP, a VNC host route to the CE. This mode may be though of as a mechanism to proxy RFP registrations of BGP unicast routes on behalf of registering NVEs.

Multiple copies of the BGP route, one per matching NVE host route, will be added to VNC. In other words, for a given BGP unicast route, each instance of a RFP-registered host route to the unicast route's next hop will result in an instance of an imported VNC route. Each such imported VNC route will have a prefix equal to the original BGP unicast route's prefix, and a next hop equal to the next hop of the matching RFP-registered host route. If there is no RFP-registered host route to the next hop of the BGP unicast route, no corresponding VNC route will be imported.

The following translations are applied:

- The Next Hop is set to the next hop of the NVE route (i.e., the VN address of the NVE).
- The extended community list in the new route is set to the union of:
- Any extended communities in the original BGP route
  - Any extended communities in the NVE route
  - An added route-origin extended community with the next hop of the original BGP route is added to the new route. The value of the local administrator field defaults 5226 but may be configured by the user via the *roo-ec-local-admin* parameter.
- The Tunnel Encapsulation attribute is set to the value of the Tunnel Encapsulation attribute of the NVE route, if any.

bgp-direct-to-nve-groups routes

Unicast routes from the main or a designated instance of BGP may be redistributed to VNC as bgp-direct-to-nve-groups routes. These routes are NOT announced via BGP, but they are made available for local RFP lookup in response to queries from NVEs.

A non-main/default BGP instance is configured using the *bgp multiple-instance* and *router bgp AS view NAME* commands as described elsewhere in this document.

In order for a route in the unicast BGP RIB to be made available to a querying NVE, there must already be, available to that NVE, an (interior) VNC route matching the next hop address of the unicast route. When the unicast route is provided to the NVE, its next hop is replaced by the next hop of the corresponding NVE. If there are multiple longest-prefix-match VNC routes, the unicast route will be replicated for each.

There is currently no policy (prefix-list or route-map) support for *bgp-direct-to-nve-groups* routes.

Redistribution Command Syntax

vnc redistribute ipv4|ipv6 bgp|bgp-direct|ipv6 bgp-direct-to-nve-groups|connected|kernel|os

vnc redistribute ipv4|ipv6 bgp-direct-to-nve-groups view VIEWNAME

no vnc redistribute ipv4|ipv6 bgp|bgp-direct|bgp-direct-to-nve-groups|connected|kernel|ospf
Import (or do not import) prefixes from another routing protocols. Specify both the address family to import (*ipv4* or *ipv6*) and the protocol (*bgp*, *bgp-direct*, *bgp-direct-to-nve-groups*, *connected*, *kernel*, *ospf*, *rip*, or *static*). Repeat this statement as needed for each combination of address family and routing protocol. Prefixes from protocol *bgp-direct* are imported from unicast BGP in the same bgpd process. Prefixes from all other protocols (including *bgp*) are imported via the *zebra* kernel routing process.

vnc redistribute mode plain|nve-group|resolve-nve
Redistribute routes from other protocols into VNC using the specified mode. Not all combinations of modes and protocols are supported.

vnc redistribute nve-group GROUP-NAME

no vnc redistribute nve-group GROUP-NAME
When using *nve-group* mode, assign (or do not assign) the NVE group *group-name* to routes redistributed from another routing protocol. *group-name* must be configured using *vnc nve-group*.

The VN and UN prefixes of the nve-group must both be configured, and each prefix must be specified as a full-length (/32 for IPv4, /128 for IPv6) prefix.

vnc redistribute lifetime LIFETIME|infinite
Assign a registration lifetime, either *lifetime* seconds or *infinite*, to prefixes redistributed from other routing protocols as if they had been received via RFP registration messages from an NVE. *lifetime* can be any integer between 1 and 4294967295, inclusive.

vnc redistribute resolve-nve roo-ec-local-admin 0-65536
Assign a value to the local-administrator subfield used in the Route Origin extended community that is assigned to routes exported under the *resolve-nve* mode. The default value is *5226*.

The following four *prefix-list* and *route-map* commands may be specified in the context of an nve-group or not. If they are specified in the context of an nve-group, they apply only if the redistribution mode is *nve-group*, and then only for routes being redistributed from *bgp-direct*. If they are specified outside the context of an nve-group, then they apply only for redistribution modes *plain* and *resolve-nve*, and then only for routes being redistributed from *bgp-direct*.

vnc redistribute bgp-direct (ipv4|ipv6) prefix-list LIST-NAME
When redistributing *bgp-direct* routes, specifies that the named prefix-list should be applied.

vnc redistribute bgp-direct no (ipv4|ipv6) prefix-list
When redistributing *bgp-direct* routes, specifies that no prefix-list should be applied.

vnc redistribute bgp-direct route-map MAP-NAME
When redistributing *bgp-direct* routes, specifies that the named route-map should be applied.

vnc redistribute bgp-direct no route-map
When redistributing *bgp-direct* routes, specifies that no route-map should be applied.

24.1.7 Configuring Export of Routes to Other Routing Protocols

Routes from VNC (both for RFP and for redistribution via BGP) can be provided to other protocols, either via zebra or directly to BGP.

It is important to note that when exporting routes to other protocols, the downstream protocol must also be configured to import the routes. For example, when VNC routes are exported to unicast BGP, the BGP configuration must include a corresponding *redistribute vnc-direct* statement.

export bgp|zebra mode none|group-nve|registering-nve|ce
Specify how routes should be exported to bgp or zebra. If the mode is *none*, routes are not exported. If the mode is *group-nve*, routes are exported according to nve-group or vrf-policy group configuration (*VNC NVE Group Configuration*): if a group is configured to allow export, then each prefix visible to the group is exported with next hops set to the currently-registered NVEs. If the mode is *registering-nve*, then all VNC routes are exported with their original next hops. If the mode is *ce*, only VNC routes that have an NVE connected CE Router encoded in a Route Origin Extended Community are exported. This extended community must have an administrative value that matches the configured *roo-ec-local-admin* value. The next hop of the exported route is set to the encoded NVE connected CE Router.

The default for both bgp and zebra is mode *none*.

vnc export bgp|zebra group-nve group GROUP-NAME

vnc export bgp|zebra group-nve no group GROUP-NAME
When export mode is *group-nve*, export (or do not export) prefixes from the specified nve-group or vrf-policy group to unicast BGP or to zebra. Repeat this statement as needed for each nve-group to be exported. Each VNC prefix that is exported will result in N exported routes to the prefix, each with a next hop corresponding to one of the N NVEs currently associated with the nve-group.

Some commands have a special meaning under certain export modes.

export bgp|zebra ipv4|ipv6 prefix-list LIST-NAME When export mode is *ce* or *registering-nve*, specifies that the named prefix-list should be applied to routes being exported to bgp or zebra. Prefix-lists for ipv4 and ipv6 are independent of each other.

export bgp|zebra no ipv4|ipv6 prefix-list When export mode is *ce* or *registering-nve*, specifies that no prefix-list should be applied to routes being exported to bgp or zebra.

export bgp|zebra route-map MAP-NAME When export mode is *ce* or *registering-nve*, specifies that the named route-map should be applied to routes being exported to bgp or zebra.

export bgp|zebra no route-map When export mode is *ce* or *registering-nve*, specifies that no route-map should be applied to routes being exported to bgp or zebra.

When the export mode is *group-nve*, policy for exported routes is specified per-NVE-group or vrf-policy group inside a *nve-group RFG-NAME* block via the following commands(*VNC NVE Group Configuration*):

export bgp|zebra route-map MAP-NAME This command is valid inside a *nve-group RFG-NAME* block. It specifies that the named route-map should be applied to routes being exported to bgp or zebra.

export bgp|zebra no route-map This command is valid inside a *nve-group RFG-NAME* block. It specifies that no route-map should be applied to routes being exported to bgp or zebra.

export bgp|zebra ipv4|ipv6 prefix-list LIST-NAME This command is valid inside a *nve-group RFG-NAME* block. It specifies that the named prefix-list filter should be applied to routes being exported to bgp or zebra. Prefix-lists for ipv4 and ipv6 are independent of each other.

export bgp|zebra no ipv4|ipv6 prefix-list This command is valid inside a *nve-group RFG-NAME* block. It specifies that no prefix-list filter should be applied to routes being exported to bgp or zebra.

24.2 Manual Address Control

The commands in this section can be used to augment normal dynamic VNC. The *add vnc* commands can be used to manually add IP prefix or Ethernet MAC address forwarding information. The *clear vnc* commands can be used to remove manually and dynamically added information.

add vnc prefix (A.B.C.D/M|X:X::X:X/M) vn (A.B.C.D|X:X::X:X) un (A.B.C.D|X:X::X:X) [cost (0-
Register an IP prefix on behalf of the NVE identified by the VN and UN addresses. The *cost* parameter provides the administrative preference of the forwarding information for remote advertisement. If omitted, it defaults to 255 (lowest preference). The *lifetime* parameter identifies the period, in seconds, that the information remains valid. If omitted, it defaults to *infinite*. The optional *local-next-hop* parameter is used to configure a nexthop to be used by an NVE to reach the prefix via a locally connected CE router. This information remains local to the NVA, i.e., not passed to other NVAs, and is only passed to registered NVEs. When specified, it is also possible to provide a *local-cost* parameter to provide a forwarding preference. If omitted, it defaults to 255 (lowest preference).

add vnc mac xx:xx:xx:xx:xx:xx virtual-network-identifier (1-4294967295) vn (A.B.C.D|X:X::X
Register a MAC address for a logical Ethernet (L2VPN) on behalf of the NVE identified by the VN and UN addresses. The optional *prefix* parameter is to support enable IP address mediation for the given prefix. The *cost* parameter provides the administrative preference of the forwarding information. If omitted, it defaults to 255. The *lifetime* parameter identifies the period, in seconds, that the information remains valid. If omitted, it defaults to *infinite*.

clear vnc prefix (*|A.B.C.D/M|X:X::X:X/M) (*|[(vn|un) (A.B.C.D|X:X::X:X|*) [(un|vn) (A.B.C
Delete the information identified by prefix, VN address, and UN address. Any or all of these parameters may be wildcarded to (potentially) match more than one registration. The optional *mac* parameter specifies a layer-2 MAC address that must match the registration(s) to be deleted. The optional *local-next-hop* parameter is used to delete specific local nexthop information.

clear vnc mac (*|xx:xx:xx:xx:xx:xx) virtual-network-identifier (*|(1-4294967295)) (*|[(vn|u
Delete mac forwarding information. Any or all of these parameters may be wildcarded to (potentially) match more than one registration. The default value for the *prefix* parameter is the wildcard value *.

clear vnc nve (*|((vn|un) (A.B.C.D|X:X::X:X) [(un|vn) (A.B.C.D|X:X::X:X)]))
Delete prefixes associated with the NVE specified by the given VN and UN addresses. It is permissible to specify only one of VN or UN, in which case any matching registration will be deleted. It is also permissible to specify * in lieu of any VN or UN address, in which case all registrations will match.

24.3 Other VNC-Related Commands

Note: VNC-Related configuration can be obtained via the *show running-configuration* command when in *enable* mode.

The following commands are used to clear and display Virtual Network Control related information:

clear vnc counters
Reset the counter values stored by the NVA. Counter values can be seen using the *show vnc* commands listed above. This command is only available in *enable* mode.

show vnc summary
Print counter values and other general information about the NVA. Counter values can be reset using the *clear vnc counters* command listed below.

show vnc nves

show vnc nves vn|un ADDRESS
Display the NVA's current clients. Specifying *address* limits the output to the NVEs whose addresses match *address*. The time since the NVA last communicated with the NVE, per-NVE summary counters and each NVE's addresses will be displayed.

show vnc queries

show vnc queries PREFIX
Display active Query information. Queries remain valid for the default Response Lifetime (*VNC Defaults Configuration*) or NVE-group Response Lifetime (*VNC NVE Group Configuration*). Specifying *prefix* limits the output to Query Targets that fall within *prefix*.

Query information is provided for each querying NVE, and includes the Query Target and the time remaining before the information is removed.

show vnc registrations [all|local|remote|holddown|imported]

show vnc registrations [all|local|remote|holddown|imported] PREFIX
Display local, remote, holddown, and/or imported registration information. Local registrations are routes received via RFP, which are present in the NVA Registrations Cache. Remote registrations are routes received via BGP (VPN SAFIs), which are present in the NVE-group import tables. Holddown registrations are local and remote routes that have been withdrawn but whose holddown timeouts have not yet elapsed. Imported information represents routes that are imported into NVA and are made available to querying NVEs. Depending on configuration, imported routes may also be advertised via BGP. Specifying *prefix* limits the output to the registered prefixes that fall within *prefix*.

Registration information includes the registered prefix, the registering NVE addresses, the registered administrative cost, the registration lifetime and the time since the information was registered or, in the case of Holddown registrations, the amount of time remaining before the information is removed.

show vnc responses [active|removed]

show vnc responses [active|removed] PREFIX
Display all, active and/or removed response information which are present in the NVA Responses Cache. Responses remain valid for the default Response Lifetime (*VNC Defaults Configuration*) or NVE-group Response Lifetime (*VNC NVE Group Configuration*.) When Removal Responses are enabled (*General VNC Configuration*), such responses are listed for the Response Lifetime. Specifying *prefix* limits the output to the addresses that fall within *prefix*.

Response information is provided for each querying NVE, and includes the response prefix, the prefix-associated registering NVE addresses, the administrative cost, the provided response lifetime and the time remaining before the information is to be removed or will become inactive.

show memory vnc
Print the number of memory items allocated by the NVA.

24.4 Example VNC and VNC-GW Configurations

24.4.1 Mesh NVA Configuration

This example includes three NVAs, nine NVEs, and two NVE groups. Note that while not shown, a single physical device may support multiple logical NVEs. *A three-way full mesh with three NVEs per NVA.* shows `code`

NVA-1 (192.168.1.100), NVA 2 (192.168.1.101), and NVA 3 (192.168.1.102), which are connected in a full mesh. Each is a member of the autonomous system 64512. Each NVA provides VNC services to three NVE clients in the 172.16.0.0/16 virtual-network address range. The 172.16.0.0/16 address range is partitioned into two NVE groups, group1 (172.16.0.0/17) and group2 (172.16.128.0/17).

Each NVE belongs to either NVE group group1 or NVE group group2. The NVEs NVE 1, NVE 2, NVE 4, NVE 7, and NVE 8 are members of the NVE group group1. The NVEs NVE 3, NVE 5, NVE 6, and NVE 9 are members of the NVE group group2.

Each NVA advertises NVE underlay-network IP addresses using the Tunnel Encapsulation Attribute.

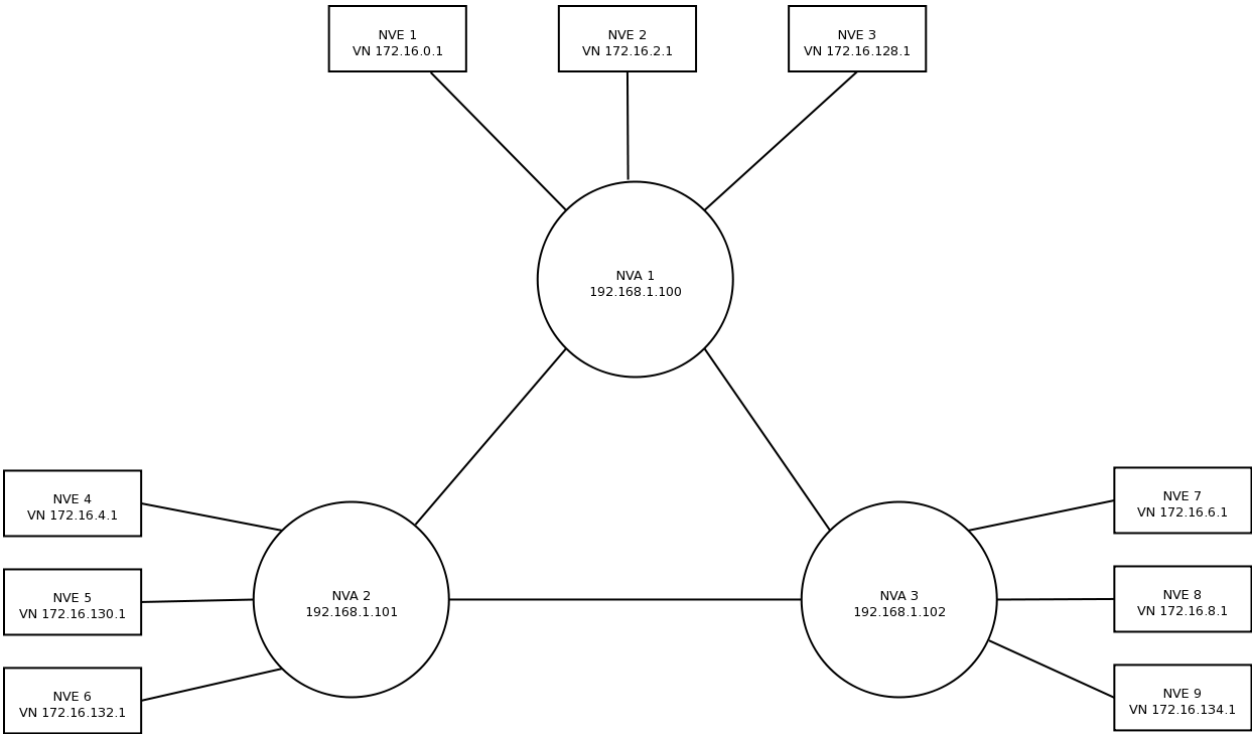


Fig. 1: A three-way full mesh with three NVEs per NVA.

bgpd.conf for NVA 1 (192.168.1.100):

```
router bgp 64512

    bgp router-id 192.168.1.100

    neighbor 192.168.1.101  remote-as 64512
    neighbor 192.168.1.102  remote-as 64512

    address-family ipv4 vpn
        neighbor 192.168.1.101 activate
        neighbor 192.168.1.102 activate
    exit-address-family

    vnc defaults
        rd 64512:1
        response-lifetime 200
        rt both 1000:1 1000:2
    exit-vnc
```

(continues on next page)

(continued from previous page)

```
    vnc nve-group group1
        prefix vn 172.16.0.0/17
        rt both 1000:1
    exit-vnc

    vnc nve-group group2
        prefix vn 172.16.128.0/17
        rt both 1000:2
    exit-vnc

exit
```

bgpd.conf for NVA 2 (192.168.1.101):

```
router bgp 64512

    bgp router-id 192.168.1.101

    neighbor 192.168.1.100  remote-as 64512
    neighbor 192.168.1.102  remote-as 64512

    address-family ipv4 vpn
        neighbor 192.168.1.100 activate
        neighbor 192.168.1.102 activate
    exit-address-family

    vnc nve-group group1
        prefix vn 172.16.0.0/17
        rd 64512:1
        response-lifetime 200
        rt both 1000:1 1000:2
    exit-vnc
exit
```

bgpd.conf for NVA 3 (192.168.1.102):

```
router bgp 64512

    bgp router-id 192.168.1.102

    neighbor 192.168.1.101  remote-as 64512
    neighbor 192.168.1.102  remote-as 64512

    address-family ipv4 vpn
        neighbor 192.168.1.100 activate
        neighbor 192.168.1.101 activate
    exit-address-family

    vnc defaults
        rd 64512:1
        response-lifetime 200
        rt both 1000:1 1000:2
    exit-vnc

    vnc nve-group group1
        prefix vn 172.16.128.0/17
```

(continues on next page)

(continued from previous page)

```
    exit-vnc
exit
```

24.4.2 Mesh NVA and VNC-GW Configuration

This example includes two NVAs, each with two associated NVEs, and two VNC-GWs, each supporting two CE routers physically attached to the four NVEs. Note that this example is showing a more complex configuration where VNC-GW is separated from normal NVA functions; it is equally possible to simplify the configuration and combine NVA and VNC-GW functions in a single FRR instance.

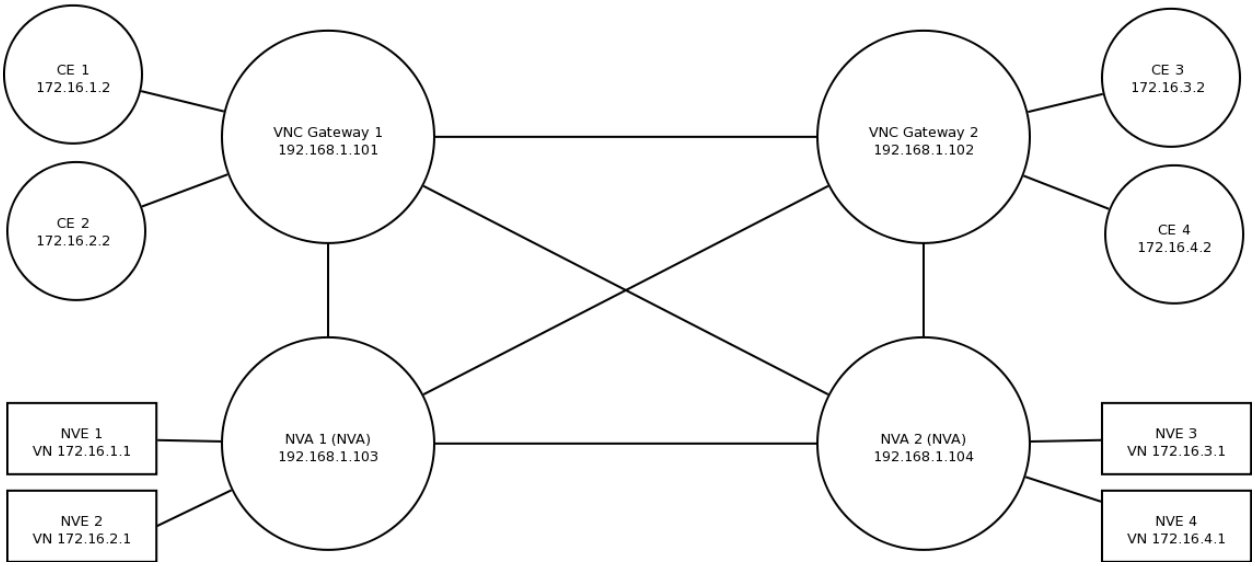


Fig. 2: Meshed NVEs and VNC-GWs

As shown in *Meshed NVEs and VNC-GWs*, NVAs and VNC-GWs are connected in a full iBGP mesh. The VNC-GWs each have two CEs configured as route-reflector clients. Each client provides BGP updates with unicast routes that the VNC-GW reflects to the other client. The VNC-GW also imports these unicast routes into VPN routes to be shared with the other VNC-GW and the two NVAs. This route importation is controlled with the `vnc redistribute` statements shown in the configuration. Similarly, registrations sent by NVEs via RFP to the NVAs are exported by the VNC-GWs to the route-reflector clients as unicast routes. RFP registrations exported this way have a next-hop address of the CE behind the connected (registering) NVE. Exporting VNC routes as IPv4 unicast is enabled with the `vnc export` command below.

The configuration for `VNC-GW 1` is shown below.

```
router bgp 64512
 bgp router-id 192.168.1.101
 bgp cluster-id 1.2.3.4
 neighbor 192.168.1.102 remote-as 64512
 neighbor 192.168.1.103 remote-as 64512
 neighbor 192.168.1.104 remote-as 64512
 neighbor 172.16.1.2 remote-as 64512
 neighbor 172.16.2.2 remote-as 64512
```

(continues on next page)

(continued from previous page)

```
 !
 address-family ipv4 unicast
  redistribute vnc-direct
  no neighbor 192.168.1.102 activate
  no neighbor 192.168.1.103 activate
  no neighbor 192.168.1.104 activate
  neighbor 172.16.1.2 route-reflector-client
  neighbor 172.16.2.2 route-reflector-client
 exit-address-family
 !
 address-family ipv4 vpn
   neighbor 192.168.1.102 activate
   neighbor 192.168.1.103 activate
   neighbor 192.168.1.104 activate
 exit-address-family
 vnc export bgp mode ce
 vnc redistribute mode resolve-nve
 vnc redistribute ipv4 bgp-direct
 exit
```

Note that in the VNC-GW configuration, the neighboring VNC-GW and NVAs each have a statement disabling the IPv4 unicast address family. IPv4 unicast is on by default and this prevents the other VNC-GW and NVAs from learning unicast routes advertised by the route-reflector clients.

Configuration for `NVA 2`:

```
router bgp 64512
 bgp router-id 192.168.1.104
 neighbor 192.168.1.101 remote-as 64512
 neighbor 192.168.1.102 remote-as 64512
 neighbor 192.168.1.103 remote-as 64512
 !
 address-family ipv4 unicast
  no neighbor 192.168.1.101 activate
  no neighbor 192.168.1.102 activate
  no neighbor 192.168.1.103 activate
 exit-address-family
 !
 address-family ipv4 vpn
   neighbor 192.168.1.101 activate
   neighbor 192.168.1.102 activate
   neighbor 192.168.1.103 activate
 exit-address-family
 !
 vnc defaults
  response-lifetime 3600
  exit-vnc
 vnc nve-group nve1
  prefix vn 172.16.1.1/32
  response-lifetime 3600
  rt both 1000:1 1000:2
  exit-vnc
 vnc nve-group nve2
  prefix vn 172.16.2.1/32
  response-lifetime 3600
  rt both 1000:1 1000:2
  exit-vnc
```

(continues on next page)

(continued from previous page)

```
exit
```

24.4.3 VNC with FRR Route Reflector Configuration

A route reflector eliminates the need for a fully meshed NVA network by acting as the hub between NVAs. *Two NVAs and a BGP Route Reflector* shows BGP route reflector `BGP Route Reflector 1` (192.168.1.100) as a route reflector for NVAs `NVA 2``(192.168.1.101) and ``NVA 3` (192.168.1.102).

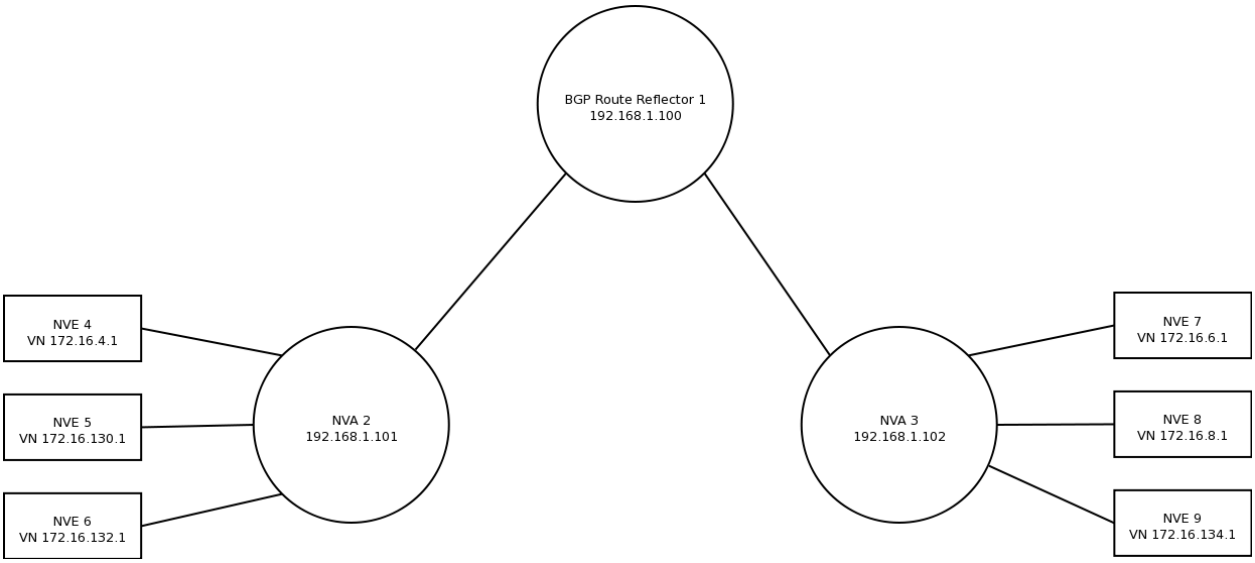


Fig. 3: Two NVAs and a BGP Route Reflector

`NVA 2` and `NVA 3` advertise NVE underlay-network IP addresses using the Tunnel Encapsulation Attribute. `BGP Route Reflector 1 reflects'' advertisements from ``NVA 2` to `NVA 3` and vice versa.

As in the example of *Mesh NVA Configuration*, there are two NVE groups. The 172.16.0.0/16 address range is partitioned into two NVE groups, `group1` (172.16.0.0/17) and `group2` (172.16.128.0/17). The NVE `NVE 4`, `NVE 7`, and `NVE 8` are members of the NVE group `group1`. The NVEs `NVE 5`, `NVE 6`, and `NVE 9` are members of the NVE group `group2`.

`bgpd.conf` for `BGP Route Reflector 1` on 192.168.1.100:

```
router bgp 64512

    bgp router-id 192.168.1.100

    neighbor 192.168.1.101 remote-as 64512
    neighbor 192.168.1.101 port 7179
    neighbor 192.168.1.101 description iBGP-client-192-168-1-101

    neighbor 192.168.1.102 remote-as 64512
    neighbor 192.168.1.102 port 7179
    neighbor 192.168.1.102 description iBGP-client-192-168-1-102

    address-family ipv4 unicast
        neighbor 192.168.1.101 route-reflector-client
        neighbor 192.168.1.102 route-reflector-client
```

(continues on next page)

(continued from previous page)

```
    exit-address-family

    address-family ipv4 vpn
        neighbor 192.168.1.101 activate
        neighbor 192.168.1.102 activate

        neighbor 192.168.1.101 route-reflector-client
        neighbor 192.168.1.102 route-reflector-client
    exit-address-family

exit
```

`bgpd.conf` for `NVA 2` on 192.168.1.101:

```
router bgp 64512

    bgp router-id 192.168.1.101

    neighbor 192.168.1.100  remote-as 64512

    address-family ipv4 vpn
        neighbor 192.168.1.100 activate
    exit-address-family

    vnc nve-group group1
        prefix vn 172.16.0.0/17
        rd 64512:1
        response-lifetime 200
        rt both 1000:1 1000:2
    exit-vnc
exit
```

`bgpd.conf` for `NVA 2` on 192.168.1.102:

```
router bgp 64512

    bgp router-id 192.168.1.102

    neighbor 192.168.1.100  remote-as 64512

    address-family ipv4 vpn
        neighbor 192.168.1.100 activate
    exit-address-family

    vnc defaults
        rd 64512:1
        response-lifetime 200
        rt both 1000:1 1000:2
    exit-vnc

    vnc nve-group group1
        prefix vn 172.16.128.0/17
    exit-vnc
exit
```

While not shown, an NVA can also be configured as a route reflector.

24.4.4 VNC with Commercial Route Reflector Configuration

This example is identical to *VNC with FRR Route Reflector Configuration* with the exception that the route reflector is a commercial router. Only the VNC-relevant configuration is provided.

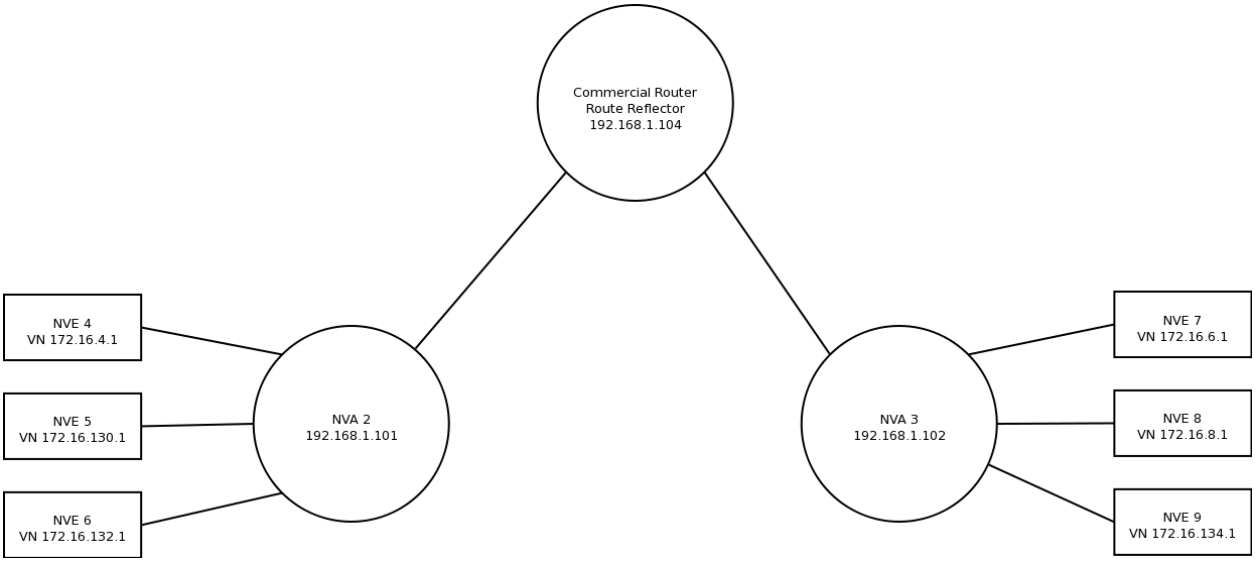


Fig. 4: Two NVAs with a commercial route reflector

`bgpd.conf` for BGP route reflector `Commercial Router` on 192.168.1.104::

```
version 8.5R1.13;
routing-options {
    rib inet.0 {
        static {
            route 172.16.0.0/16 next-hop 192.168.1.104;
        }
    }
    autonomous-system 64512;
    resolution {
        rib inet.3 {
            resolution-ribs inet.0;
        }
        rib bgp.l3vpn.0 {
            resolution-ribs inet.0;
        }
    }
}
protocols {
    bgp {
        advertise-inactive;
        family inet {
            labeled-unicast;
        }
       group 1 {
            type internal;
            advertise-inactive;
            advertise-peer-as;
            import h;
            family inet {
```

(continues on next page)

(continued from previous page)

```
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            cluster 192.168.1.104;
            neighbor 192.168.1.101;
            neighbor 192.168.1.102;
        }
    }
}
policy-options {
    policy-statement h {
        from protocol bgp;
        then {
            as-path-prepend 64512;
            accept;
        }
    }
}
```

bgpd.conf for NVA 2 on 192.168.1.101:

```
router bgp 64512

    bgp router-id 192.168.1.101

    neighbor 192.168.1.100  remote-as 64512

    address-family ipv4 vpn
        neighbor 192.168.1.100 activate
    exit-address-family

    vnc nve-group group1
        prefix vn 172.16.0.0/17
        rd 64512:1
        response-lifetime 200
        rt both 1000:1 1000:2
    exit-vnc
exit
```

bgpd.conf for NVA 3 on 192.168.1.102:

```
router bgp 64512

    bgp router-id 192.168.1.102

    neighbor 192.168.1.100  remote-as 64512

    address-family ipv4 vpn
        neighbor 192.168.1.100 activate
    exit-address-family

    vnc defaults
        rd 64512:1
        response-lifetime 200
        rt both 1000:1 1000:2
```

(continues on next page)

(continued from previous page)

```
    exit-vnc

    vnc nve-group group1
        prefix vn 172.16.128.0/17
    exit-vnc
exit
```

24.4.5 VNC with Redundant Route Reflectors Configuration

This example combines the previous two (*VNC with FRR Route Reflector Configuration* and *VNC with Commercial Route Reflector Configuration*) into a redundant route reflector configuration. BGP route reflectors `BGP Route Reflector 1` and `Commercial Router` are the route reflectors for NVAs `NVA 2` and `NVA 3`. The two NVAs have connections to both route reflectors.

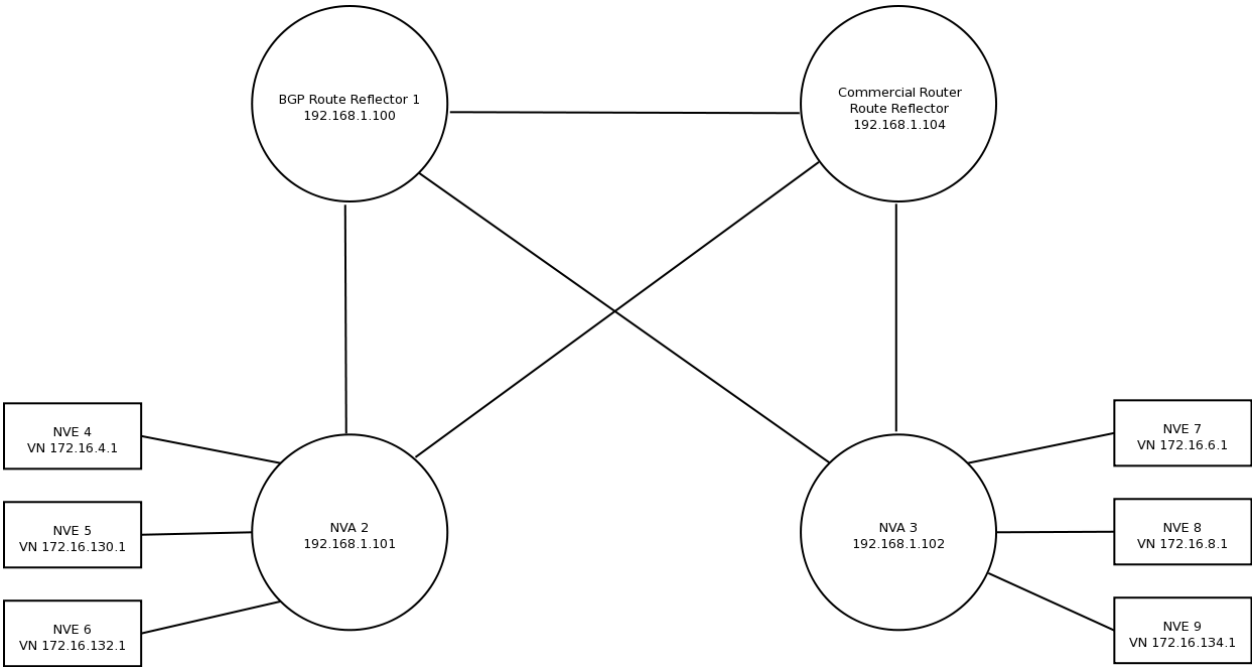


Fig. 5: FRR-based NVA with redundant route reflectors

`bgpd.conf` for `BPGD Route Reflector 1` on 192.168.1.100:

```
router bgp 64512

 bgp router-id 192.168.1.100
 bgp cluster-id 192.168.1.100

 neighbor 192.168.1.104 remote-as 64512

 neighbor 192.168.1.101 remote-as 64512
 neighbor 192.168.1.101 description iBGP-client-192-168-1-101
 neighbor 192.168.1.101 route-reflector-client

 neighbor 192.168.1.102 remote-as 64512
 neighbor 192.168.1.102 description iBGP-client-192-168-1-102
```

(continues on next page)

(continued from previous page)

```
 neighbor 192.168.1.102 route-reflector-client

 address-family ipv4 vpn
  neighbor 192.168.1.101 activate
  neighbor 192.168.1.102 activate
  neighbor 192.168.1.104 activate

  neighbor 192.168.1.101 route-reflector-client
  neighbor 192.168.1.102 route-reflector-client
 exit-address-family
exit
```

`bgpd.conf` for `NVA 2` on 192.168.1.101:

```
router bgp 64512

 bgp router-id 192.168.1.101

 neighbor 192.168.1.100  remote-as 64512
 neighbor 192.168.1.104  remote-as 64512

 address-family ipv4 vpn
  neighbor 192.168.1.100 activate
  neighbor 192.168.1.104 activate
 exit-address-family

 vnc nve-group group1
  prefix vn 172.16.0.0/17
  rd 64512:1
  response-lifetime 200
  rt both 1000:1 1000:2
 exit-vnc
exit
```

`bgpd.conf` for `NVA 3` on 192.168.1.102:

```
router bgp 64512

 bgp router-id 192.168.1.102

 neighbor 192.168.1.100  remote-as 64512
 neighbor 192.168.1.104  remote-as 64512

 address-family ipv4 vpn
  neighbor 192.168.1.100 activate
  neighbor 192.168.1.104 activate
 exit-address-family

 vnc defaults
  rd 64512:1
  response-lifetime 200
  rt both 1000:1 1000:2
 exit-vnc

 vnc nve-group group1
  prefix vn 172.16.128.0/17
 exit-vnc
```

(continues on next page)

(continued from previous page)

```
exit
```

bgpd.conf for the Commercial Router route reflector on 192.168.1.104::

```
routing-options {
    rib inet.0 {
        static {
            route 172.16.0.0/16 next-hop 192.168.1.104;
        }
    }
    autonomous-system 64512;
    resolution {
        rib inet.3 {
            resolution-ribs inet.0;
        }
        rib bgp.l3vpn.0 {
            resolution-ribs inet.0;
        }
    }
}
protocols {
    bgp {
        advertise-inactive;
        family inet {
            labeled-unicast;
        }
       group 1 {
            type internal;
            advertise-inactive;
            advertise-peer-as;
            import h;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            cluster 192.168.1.104;
            neighbor 192.168.1.101;
            neighbor 192.168.1.102;
        }

       group 2 {
            type internal;
            advertise-inactive;
            advertise-peer-as;
            import h;
            family inet {
                unicast;
            }
            family inet-vpn {
                unicast;
            }
            neighbor 192.168.1.100;
        }

    }
```

(continues on next page)

(continued from previous page)

```
}
policy-options {
    policy-statement h {
        from protocol bgp;
        then {
            as-path-prepend 64512;
            accept;
        }
    }
}
```

CHAPTER 25

Glossary

distance-vector A distance-vector routing protocol in data networks determines the best route for data packets based on distance. Distance-vector routing protocols measure the distance by the number of routers a packet has to pass. Some distance-vector protocols also take into account network latency and other factors that influence traffic on a given route. To determine the best route across a network, routers on which a distance-vector protocol is implemented exchange information with one another, usually routing tables plus hop counts for destination networks and possibly other traffic information. Distance-vector routing protocols also require that a router informs its neighbours of network topology changes periodically. [distance-vector-rp]

link-state Link-state algorithms (also known as shortest path first algorithms) flood routing information to all nodes in the internetwork. Each router, however, sends only the portion of the routing table that describes the state of its own links. In link-state algorithms, each router builds a picture of the entire network in its routing tables. Distance vector algorithms (also known as Bellman-Ford algorithms) call for each router to send all or some portion of its routing table, but only to its neighbors. In essence, link-state algorithms send small updates everywhere, while distance vector algorithms send larger updates only to neighboring routers. Distance vector algorithms know only about their neighbors. [link-state-rp]

Bellman-Ford The Bellman–Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph. [bellman-ford]

CHAPTER 26

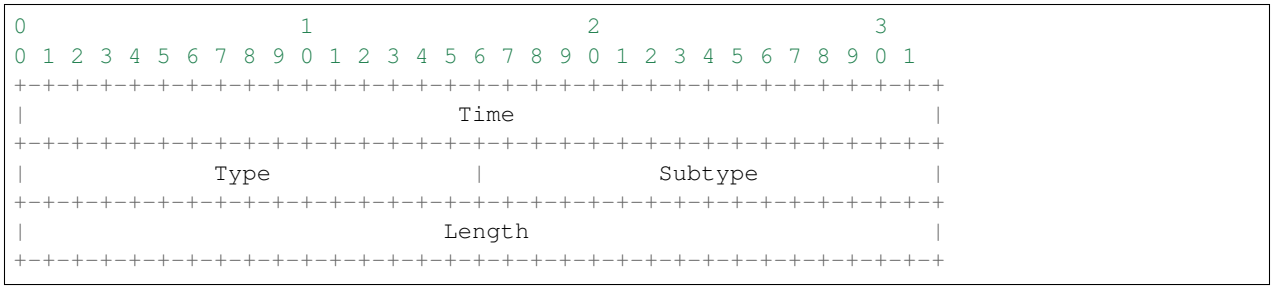
Packet Binary Dump Format

FRR can dump routing protocol packets into a file with a binary format.

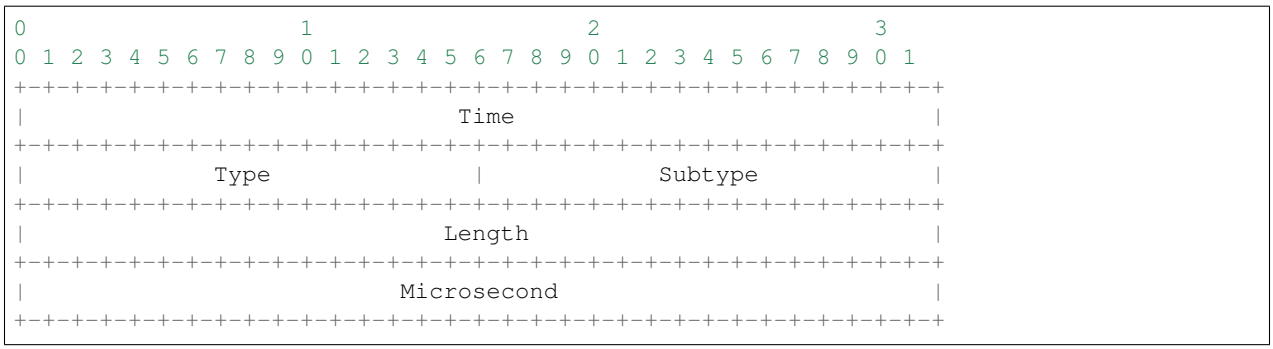
It seems to be better that we share the MRT's header format for backward compatibility with MRT's dump logs. We should also define the binary format excluding the header, because we must support both IP v4 and v6 addresses as socket addresses and / or routing entries.

In the last meeting, we discussed to have a version field in the header. But Masaki told us that we can define new 'type' value rather than having a 'version' field, and it seems to be better because we don't need to change header format.

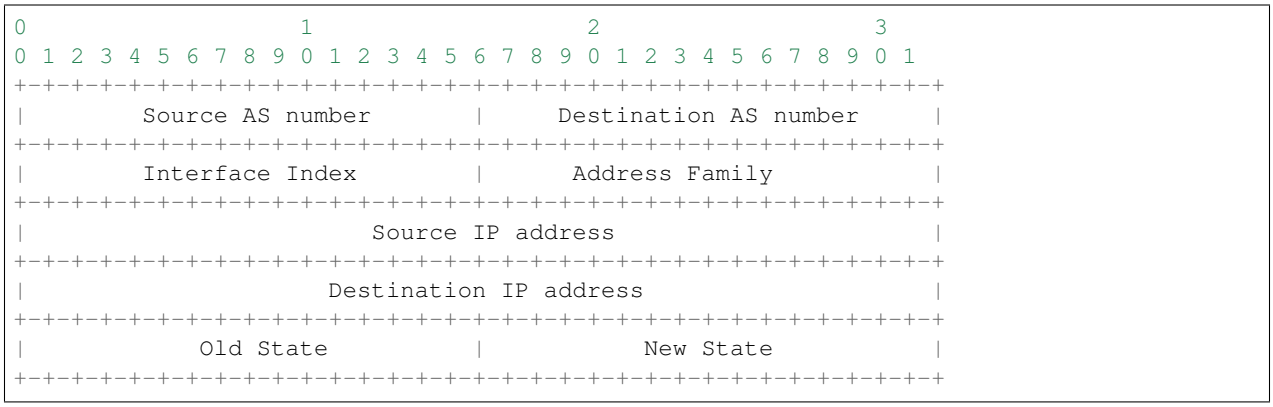
Here is the common header format. This is same as that of MRT.:



If 'type' is PROTOCOL_BGP4MP_ET, the common header format will contain an additional microsecond field (RFC6396 2011).:

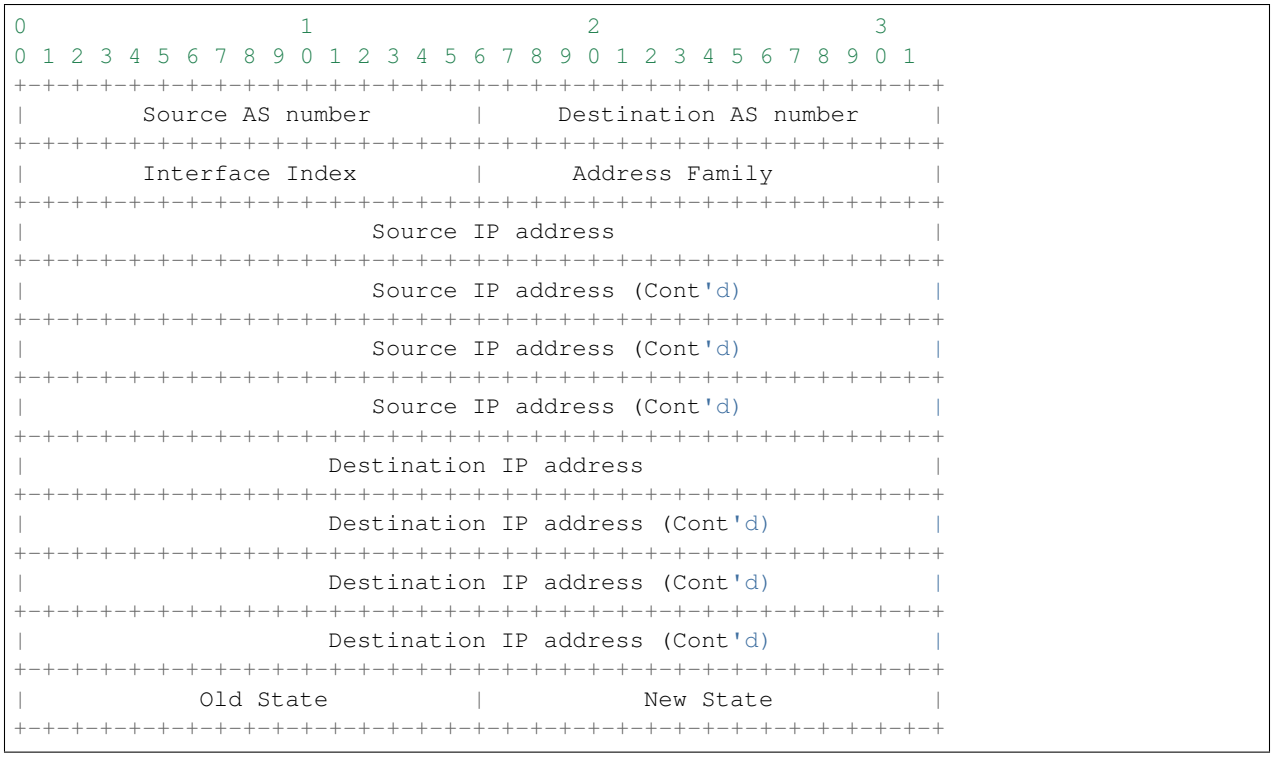


If 'type' is PROTOCOL_BGP4MP, 'subtype' is BGP4MP_STATE_CHANGE, and Address Family == IP (version 4):

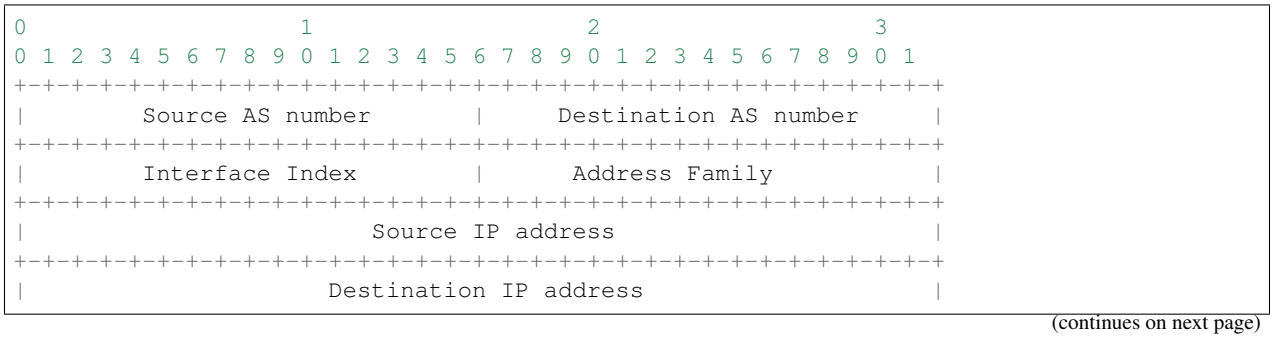


Where State is the value defined in RFC1771.

If 'type' is PROTOCOL_BGP4MP, 'subtype' is BGP4MP_STATE_CHANGE, and Address Family == IP version 6:

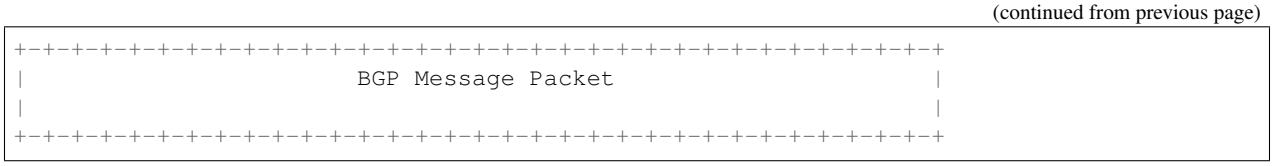


If 'type' is PROTOCOL_BGP4MP, 'subtype' is BGP4MP_MESSAGE, and Address Family == IP (version 4):



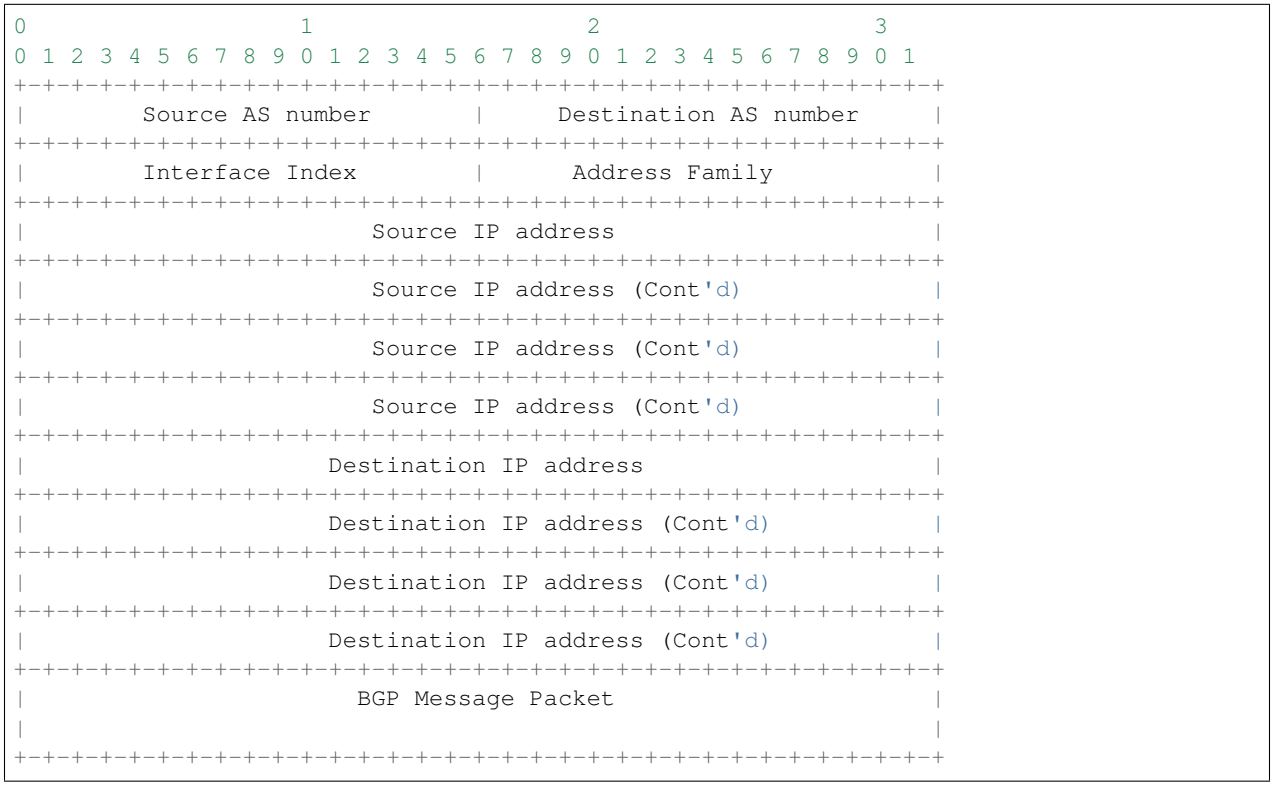
(continues on next page)

(continued from previous page)

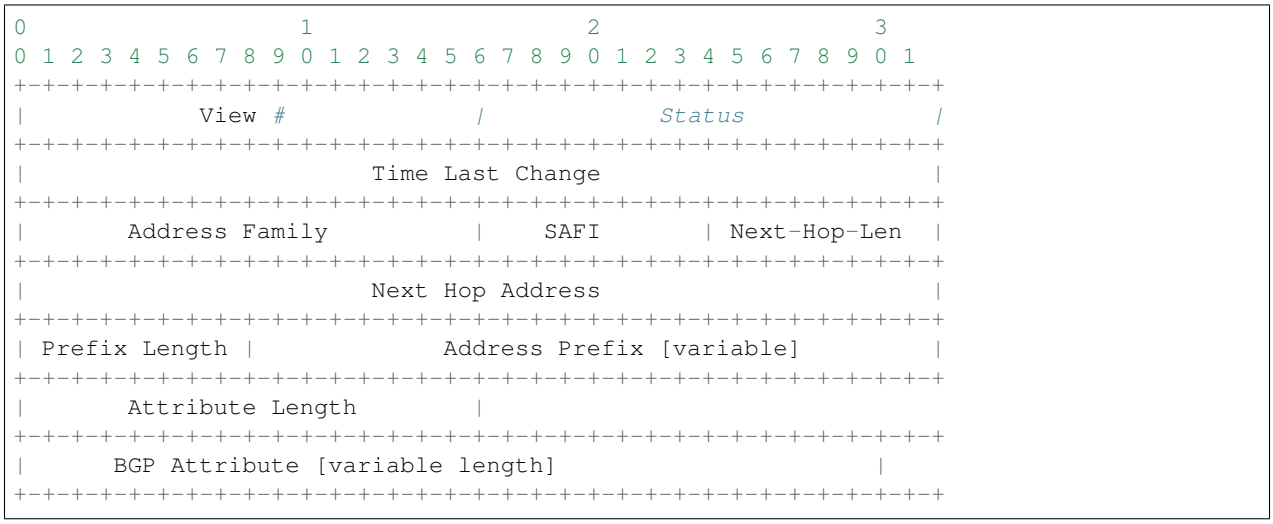


Where BGP Message Packet is the whole contents of the BGP4 message including header portion.

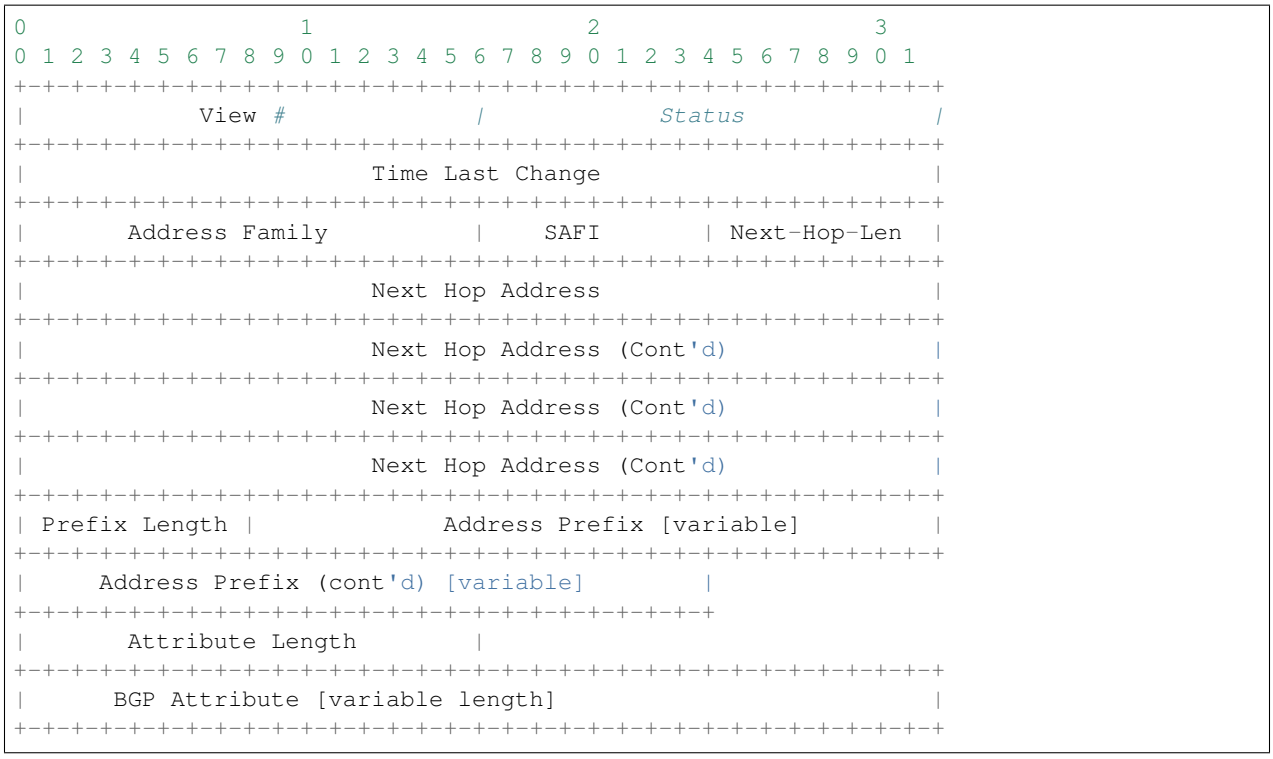
If 'type' is PROTOCOL_BGP4MP, 'subtype' is BGP4MP_MESSAGE, and Address Family == IP version 6:



If 'type' is PROTOCOL_BGP4MP, 'subtype' is BGP4MP_ENTRY, and Address Family == IP (version 4):

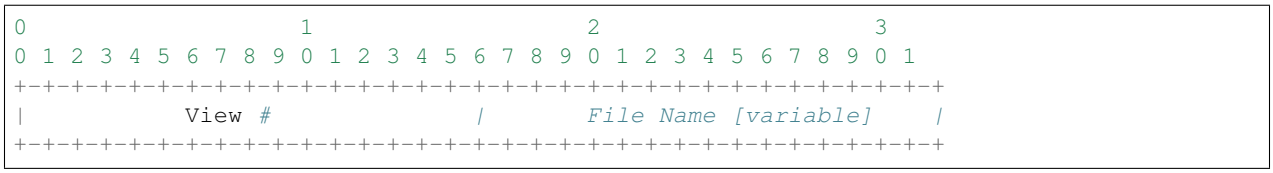


If 'type' is PROTOCOL_BGP4MP, 'subtype' is BGP4MP_ENTRY, and Address Family == IP version 6:



BGP4 Attribute must not contain MP_UNREACH_NLRI. If BGP Attribute has MP_REACH_NLRI field, it must has zero length NLRI, e.g., MP_REACH_NLRI has only Address Family, SAFI and next-hop values.

If 'type' is PROTOCOL_BGP4MP and 'subtype' is BGP4MP_SNAPSHOT:



The file specified in "File Name" contains all routing entries, which are in the format of `subtype == BGP4MP_ENTRY`.

```
Constants:

  /\* type value \*/
  #define MSG_PROTOCOL_BGP4MP    16
  #define MSG_PROTOCOL_BGP4MP_ET 17
  /\* subtype value \*/
  #define BGP4MP_STATE_CHANGE 0
  #define BGP4MP_MESSAGE 1
  #define BGP4MP_ENTRY 2
  #define BGP4MP_SNAPSHOT 3
```

Bibliography

[Securing-BGP] Geoff Huston, Randy Bush: Securing BGP, In: The Internet Protocol Journal, Volume 14, No. 2, 2011. <http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_14-2/142_bgp.html>

[Resource-Certification] Geoff Huston: Resource Certification, In: The Internet Protocol Journal, Volume 12, No.1, 2009. <http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_12-1/121_resource.html>

[Draft-IETF-IDR-Flowspec-redirect-IP] <https://tools.ietf.org/id/draft-ietf-idr-flowspec-redirect-ip-02.txt>

[Draft-IETF-IDR-Flowspec-Interface-Set] <https://tools.ietf.org/id/draft-ietf-idr-flowspec-interfaceset-03.txt>

[Presentation] <https://docs.google.com/presentation/d/1ekQygUAG5yvQ3wWUyrw4Wcag0LgmbW1kV02IWcU4iUg/edit#slide=id.g378f0e1b5e_1_44>

[bgp-route-osci-cond] McPherson, D. and Gill, V. and Walton, D., "Border Gateway Protocol (BGP) Persistent Route Oscillation Condition", IETF RFC3345

[stable-flexible-ibgp] Flavel, A. and M. Roughan, "Stable and flexible iBGP", ACM SIGCOMM 2009

[ibgp-correctness] Griffin, T. and G. Wilfong, "On the correctness of IBGP configuration", ACM SIGCOMM 2002

[distance-vector-rp] https://en.wikipedia.org/wiki/Distance-vector_routing_protocol

[link-state-rp] https://en.wikipedia.org/wiki/Link-state_routing_protocol

[bellman-ford] https://en.wikipedia.org/wiki/Bellman-Ford_algorithm

Index

Symbols

-disable-backtrace
    configure command line option, 6
-disable-bgp-announce
    configure command line option, 6
-disable-bgpd
    configure command line option, 5
-disable-isisd
    configure command line option, 6
-disable-ospf-ri
    configure command line option, 6
-disable-ospf6d
    configure command line option, 5
-disable-ospfapi
    configure command line option, 6
-disable-ospfclient
    configure command line option, 6
-disable-ospfd
    configure command line option, 5
-disable-ripd
    configure command line option, 5
-disable-ripngd
    configure command line option, 5
-disable-rtadv
    configure command line option, 6
-disable-snmp
    configure command line option, 6
-disable-vtysh
    configure command line option, 6
-disable-zebra
    configure command line option, 5
-enable-datacenter
    configure command line option, 6
-enable-dev-build
    configure command line option, 6
-enable-fpm
    configure command line option, 7
-enable-fuzzing
    configure command line option, 6
-enable-gcc-rdynamic
    configure command line option, 6
-enable-group <user>
    configure command line option, 7
-enable-isis-te
    configure command line option, 6
-enable-isis-topology
    configure command line option, 6
-enable-multipath=X
    configure command line option, 7
-enable-numeric-version
    configure command line option, 7
-enable-realms
    configure command line option, 6
-enable-snmp
    configure command line option, 6
-enable-user <user>
    configure command line option, 7
-enable-vty-group <group>
    configure command line option, 7
-localstatedir <dir>
    configure command line option, 7
-prefix <prefix>
    configure command line option, 7
-sysconfdir <dir>
    configure command line option, 7
-A, -vty_addr <address>
    command line option, 14
-M, -module <module:options>
    command line option, 15
-P, -vty_port <port>
    command line option, 14
-b, -batch
    zebra command line option, 43
-d, -daemon
    command line option, 14
-e X, -ecmp X
    zebra command line option, 43
-f, -config_file <file>
    command line option, 14

-h, -help
command line option, 14
-i, -pid_file <file>
command line option, 14
-k, -keep_kernel
zebra command line option, 43
-l, -listenon
bgpd command line option, 53
-n, -vrfwnetns
zebra command line option, 43
-p, -bgp_port <port>
bgpd command line option, 53
-r, -retain
bgpd command line option, 53
command line option, 156
eigrpd command line option, 105
zebra command line option, 43
-u <user>
command line option, 15
-v, -version
command line option, 15
[no] debug bgp flowspec, 98
[no] debug bgp pbr [error], 98
[no] local-install <IFNAME | any>, 97
[no] log timestamp precision (0-6), 12
[no] rt redirect import RTLIST..., 98
[no] segment-routing global-block (0-1048575) (0-1048575), 138
[no] segment-routing node-msd (1-16), 138
[no] segment-routing on, 138
[no] segment-routing prefix A.B.C.D/M index (0-65535) [no-php-flag], 138
296>, 93

Numbers

294, 93
967, 93

A

access-class ACCESS-LIST, 13
access-list NAME deny IPV4-NETWORK, 23
access-list NAME permit IPV4-NETWORK, 23
agentx, 39
aggregate-address A.B.C.D/M, 59
aggregate-address A.B.C.D/M as-set, 59
aggregate-address A.B.C.D/M summary-only, 59
area (0-4294967295) authentication, 133
area (0-4294967295) authentication message-digest, 133
area (0-4294967295) export-list NAME, 132
area (0-4294967295) filter-list prefix NAME in, 133
area (0-4294967295) filter-list prefix NAME out, 133
area (0-4294967295) import-list NAME, 133
area (0-4294967295) range A.B.C.D/M, 131
area (0-4294967295) shortcut, 132
area (0-4294967295) stub, 132
area (0-4294967295) stub no-summary, 132
area (0-4294967295) virtual-link A.B.C.D, 132
area A.B.C.D authentication, 133
area A.B.C.D authentication message-digest, 133
area A.B.C.D default-cost (0-16777215), 132
area A.B.C.D export-list NAME, 132
area A.B.C.D filter-list prefix NAME in, 133
area A.B.C.D filter-list prefix NAME out, 133
area A.B.C.D import-list NAME, 132
area A.B.C.D range A.B.C.D/M, 131
area A.B.C.D range IPV4_PREFIX not-advertise, 131
area A.B.C.D range IPV4_PREFIX substitute IPV4_PREFIX, 131
area A.B.C.D shortcut, 132
area A.B.C.D stub, 132
area A.B.C.D stub no-summary, 132
area A.B.C.D virtual-link A.B.C.D, 131
area-password [clear | md5] <password>, 110
auto-cost reference-bandwidth (1-4294967), 130
auto-cost reference-bandwidth COST, 144

B

babel <wired|wireless>, 102
babel channel (1-254), 102
babel channel interfering, 102
babel channel noninterfering, 102
babel diversity, 101
babel diversity-factor (1-256), 101
babel enable-timestamps, 103
babel hello-interval (20-655340), 102
babel max-rtt-penalty (0-65535), 102
babel resend-delay (20-655340), 101, 103
babel rtt-decay (1-256), 102
babel rtt-max (1-65535), 102
babel rtt-min (1-65535), 102
babel rxcost (1-65534), 102
babel smoothing-half-life (0-65534), 103
babel split-horizon, 102

babel update-interval (*20-655340*), 102
bandwidth (*1-10000000*), 44
banner motd default, 13
Bellman-Ford, **193**
bgp always-compare-med, 58
bgp bestpath as-path confed, 54
bgp bestpath as-path multipath-relax, 55
bgp cluster-id A.B.C.D, 74
bgp config-type cisco, 75
bgp config-type zebra, 75
bgp deterministic-med, 58
bgp multiple-instance, 74
bgp route-reflector allow-outbound-policy, 62
bgp router-id A.B.C.D, 54
bgpd command line option
    -l, -listenon, 53
    -p, -bgp_port <port>, 53
    -r, -retain, 53
Bug hunting, 4
Bug Reports, 4
Build options, 5
Building on Linux boxes, 8
Building the system, 5

C

Call Action, **27**
call NAME, 30
call WORD, 87
capability opaque, 137
clear bgp ipv4|ipv6 *, 73
clear bgp ipv4|ipv6 PEER, 73
clear bgp ipv4|ipv6 PEER soft in, 73
clear ip prefix-list, 25
clear ip prefix-list NAME, 25
clear ip prefix-list NAME A.B.C.D/M, 25
clear vnc counters, 179
clear vnc mac (*|xx:xx:xx:xx:xx:xx) virtual-network-identifier (*|(1-4294967295)) (*|[(vn|un) (A.B.C.D|X:X::X:X|*) [(un|vn) (A.B.C.D|X:X::X:X|*)] [prefix (*|A.B.C.D/M|X:X::X:X/M)]), 179
clear vnc nve (*|((vn|un) (A.B.C.D|X:X::X:X) [(un|vn) (A.B.C.D|X:X::X:X)])), 179
clear zebra fpm stats, 52
command line option
    -A, -vty_addr <address>, 14
    -M, -module <module:options>, 15
    -P, -vty_port <port>, 14
    -d, -daemon, 14
    -f, -config_file <file>, 14
    -h, -help, 14
    -i, -pid_file <file>, 14
    -r, -retain, 156
    -u <user>, 15
    -v, -version, 15
Compatibility with other systems, 2
Configuration files for running the software, 11
Configuration options, 5
configure command line option
    -disable-backtrace, 6
    -disable-bgp-announce, 6
    -disable-bgpd, 5
    -disable-isisd, 6
    -disable-ospf-ri, 6
    -disable-ospf6d, 5
    -disable-ospfapi, 6
    -disable-ospfclient, 6
    -disable-ospfd, 5
    -disable-ripd, 5
    -disable-ripngd, 5
    -disable-rtadv, 6
    -disable-snmp, 6
    -disable-vtysh, 6
    -disable-zebra, 5
    -enable-datacenter, 6
    -enable-dev-build, 6
    -enable-fpm, 7
    -enable-fuzzing, 6
    -enable-gcc-rdynamic, 6
    -enable-group <user>, 7
    -enable-isis-te, 6
    -enable-isis-topology, 6
    -enable-multipath=X, 7
    -enable-numeric-version, 7
    -enable-realms, 6
    -enable-snmp, 6
    -enable-user <user>, 7
    -enable-vty-group <group>, 7
    -localstatedir <dir>, 7
    -prefix <prefix>, 7
    -sysconfdir <dir>, 7
configure terminal, 14
Configuring FRR, 8
Contact information, 4
continue, 30
continue N, 30

D

debug eigrp packets, 107
debug eigrp transmit, 107
debug event, 73
debug isis adj-packets, 113

debug isis checksum-errors, 113
debug isis events, 114
debug isis local-updates, 114
debug isis packet-dump, 114
debug isis protocol-errors, 114
debug isis route-events, 114
debug isis snp-packets, 114
debug isis spf-events, 114
debug isis spf-statistics, 114
debug isis spf-triggers, 114
debug isis update-packets, 114
debug keepalive, 73
debug ospf event, 139
debug ospf ism, 139
debug ospf ism (*status|events|timers*), 139
debug ospf lsa, 139
debug ospf lsa (*generate|flooding|refresh*), 139
debug ospf nsm, 139
debug ospf nsm (*status|events|timers*), 139
debug ospf nssa, 139
debug ospf packet (hello|dd|ls-request|ls-update|ls-ack|all) (send|recv) [detail], 138
debug ospf te, 139
debug ospf zebra, 139
debug ospf zebra (*interface|redistribute*), 139
debug pim events, 151
debug pim nht, 151
debug pim packet-dump, 151
debug pim packets, 151
debug pim trace, 151
debug pim zebra, 151
debug rip events, 162
debug rip packet, 162
debug rip zebra, 162
debug ripng events, 166
debug ripng packet, 166
debug ripng zebra, 166
debug rpki, 94
debug update, 73
default-information originate, 135, 158
default-information originate always, 135
default-information originate always metric (*0-16777214*), 135
default-information originate always metric (0-16777214) metric-type (*1|2*), 135
default-information originate always metric (0-16777214) metric-type (1|2) route-map WORD, 135
default-information originate metric (*0-16777214*), 135
default-information originate metric (0-16777214) metric-type (*1|2*), 135
default-information originate metric (0-16777214) metric-type (1|2) route-map WORD, 135
default-metric (*0-16777214*), 136
default-metric (*1-16*), 159
description DESCRIPTION ..., 44
distance (*1-255*), 136, 159
distance (1-255) A.B.C.D/M, 54, 159
distance (1-255) A.B.C.D/M ACCESS-LIST, 160
distance (1-255) A.B.C.D/M word, 54
distance bgp (1-255) (1-255) (*1-255*), 54
distance ospf (intra-area|inter-area|external) (*1-255*), 136
distance-vector, **193**
Distance-vector routing protocol, 121
distribute-list ACCESS_LIST (in|out) IFNAME, 166
distribute-list ACCESS_LIST DIRECT IFNAME, 159
distribute-list NAME out (kernel|connected|static|rip|ospf, 136
distribute-list prefix PREFIX_LIST (in|out) IFNAME, 159
Distribution configuration, 5
domain-password [clear | md5] <password>, 110
DUAL, **105**
dump bgp all PATH [INTERVAL], 77
dump bgp all-et PATH [INTERVAL], 77
dump bgp routes-mrt PATH, 78
dump bgp routes-mrt PATH INTERVAL, 78
dump bgp updates PATH [INTERVAL], 78
dump bgp updates-et PATH [INTERVAL], 78

E

eigrpd command line option -r, -retain, 105
enable password PASSWORD, 12
Errors in the software, 4
exec-timeout MINUTE [SECOND], 13
Exit Policy, **27**
exit-vnc, 172
export bgp|zebra ipv4|ipv6 prefix-list LIST-NAME, 174
export bgp|zebra mode none|group-nve|registering-nve|ce, 178
export bgp|zebra no ipv4|ipv6 prefix-list, 174
export bgp|zebra no route-map, 174

export bgp|zebra route-map MAP-NAME, 174

F

Files for running configurations, 11
flush_timer TIME, 165
Found a bug?, 4
FRR Least-Privileges, 7
FRR on other systems, 2
FRR Privileges, 7

G

Getting the herd running, 11

H

hostname dynamic, 109
hostname HOSTNAME, 11
How to get in touch with FRR, 4
How to install FRR, 5

I

import vrf VRFNAME, 72
import|export vpn, 72
Installation, 5
Installing FRR, 5
interface IFNAME, 44
interface IFNAME area AREA, 143
interface IFNAME vrf VRF, 44
ip address ADDRESS/PREFIX, 44
ip address ADDRESS/PREFIX secondary, 44
ip address LOCAL-ADDR peer PEER-ADDR/PREFIX, 44
ip as-path access-list WORD permit|deny LINE, 63
ip community-list (1-99) permit|deny COMMUNITY, 65
ip community-list (100-199) permit|deny COMMUNITY, 65
ip community-list expanded NAME permit|deny LINE, 64
ip community-list NAME permit|deny COMMUNITY, 65
ip community-list standard NAME permit|deny COMMUNITY, 64
ip extcommunity-list expanded NAME permit|deny LINE, 68
ip extcommunity-list standard NAME permit|deny EXTCOMMUNITY, 68
ip igmp, 149
ip igmp query-interval (*1-1800*), 149
ip igmp query-max-response-time (*10-250*), 149
ip igmp version (*2-3*), 149
ip large-community-list expanded NAME permit|deny LINE, 69
ip large-community-list standard NAME permit|deny LARGE-COMMUNITY, 69
ip mroute A.B.C.D/M A.B.C.D (*1-255*), 149
ip mroute A.B.C.D/M INTERFACE (*1-255*), 149
ip mroute PREFIX NEXTHOP [DISTANCE], 49
ip multicast rpf-lookup-mode MODE, 49
ip multicast rpf-lookup-mode WORD, 148
ip multicat boundary oil WORD, 149
ip ospf area (A.B.C.D|(0-4294967295)), 135
ip ospf area AREA [ADDR], 133
ip ospf authentication message-digest, 133
ip ospf authentication-key AUTH_KEY, 133
ip ospf cost (*1-65535*), 134
ip ospf dead-interval (*1-65535*), 134
ip ospf dead-interval minimal hello-multiplier (*2-20*), 134
ip ospf hello-interval (*1-65535*), 134
ip ospf message-digest-key KEYID md5 KEY, 134
ip ospf network (*broadcast|non-broadcast|point-to-multipoint|point-to-point*), 134
ip ospf priority (*0-255*), 134
ip ospf retransmit-interval (*1-65535*), 134
ip ospf transmit-delay, 135
ip pim bfd, 149
ip pim drpriority (*1-4294967295*), 149
ip pim ecmp, 148
ip pim ecmp rebalance, 148
ip pim hello (1-180) (*1-180*), 149
ip pim join-prune-interval (*60-600*), 148
ip pim keep-alive-timer (*31-60000*), 148
ip pim packets (*1-100*), 148
ip pim register-suppress-time (*5-60000*), 148
ip pim rp A.B.C.D A.B.C.D/M, 147
ip pim send-v6-secondary, 148
ip pim sm, 149
ip pim spt-switchover infinity-and-beyond, 148
ip pim ssm prefix-list WORD, 148
ip prefix-list NAME (permit|deny) PREFIX [le LEN] [ge LEN], 23
ip prefix-list NAME description DESC, 24
ip prefix-list NAME seq NUMBER (permit|deny) PREFIX [le LEN] [ge LEN], 23
ip prefix-list sequence-number, 24
ip protocol PROTOCOL route-map ROUTEMAP, 50
ip rip authentication key-chain KEY-CHAIN, 161

ip rip authentication mode md5, 161
ip rip authentication mode text, 161
ip rip authentication string STRING, 161
ip rip receive version VERSION, 157
ip rip send version VERSION, 157
ip route NETWORK GATEWAY, 46
ip route NETWORK GATEWAY DISTANCE, 46
ip route NETWORK NETMASK GATEWAY, 46
ip route NETWORK NETMASK GATEWAY NEXTHOPVRF, 48
ip route NETWORK NETMASK GATEWAY table TABLENO, 48
ip router isis WORD, 111
ip split-horizon, 157
ipv6 address ADDRESS/PREFIX, 44
ipv6 nd adv-interval-option, 32
ipv6 nd home-agent-config-flag, 32
ipv6 nd home-agent-lifetime (*0-65520*), 32
ipv6 nd home-agent-preference (*0-65535*), 32
ipv6 nd managed-config-flag, 32
ipv6 nd mtu (*1-65535*), 32
ipv6 nd other-config-flag, 32
ipv6 nd prefix ipv6prefix [valid-lifetime] [preferred-lifetime] [off-link] [no-autoconfig] [router-address], 31
ipv6 nd ra-interval msec (*70-1800000*), 32
ipv6 nd ra-lifetime (*0-9000*), 32
ipv6 nd reachable-time (*1-3600000*), 32
ipv6 nd router-preference (*high|medium|low*), 32
ipv6 nd suppress-ra, 31
ipv6 ospf6 cost COST, 144
ipv6 ospf6 dead-interval DEADINTERVAL, 144
ipv6 ospf6 hello-interval HELLOINTERVAL, 144
ipv6 ospf6 network (*broadcast|point-to-point*), 144
ipv6 ospf6 priority PRIORITY, 144
ipv6 ospf6 retransmit-interval RETRANSMITINTERVAL, 144
ipv6 ospf6 transmit-delay TRANSMITDELAY, 144
ipv6 route NETWORK from SRCPREFIX GATEWAY, 47
ipv6 route NETWORK from SRCPREFIX GATEWAY DISTANCE, 47
ipv6 route NETWORK GATEWAY, 47
ipv6 route NETWORK GATEWAY DISTANCE, 47
is-type [level-1 | level-1-2 | level-2-only], 111
isis circuit-type [level-1 | level-1-2 | level-2], 111
isis csnp-interval (*1-600*), 111
isis csnp-interval (1-600) [level-1 | level-2], 111
isis hello padding, 111
isis hello-interval (*1-600*), 111
isis hello-interval (1-600) [level-1 | level-2], 111
isis hello-multiplier (*2-100*), 111
isis hello-multiplier (2-100) [level-1 | level-2], 111
isis metric [(0-255) | (0-16777215)], 112
isis metric [(0-255) | (0-16777215)] [level-1 | level-2], 112
isis network point-to-point, 112
isis passive, 112
isis password [clear | md5] <password>, 112
isis priority (*0-127*), 112
isis priority (0-127) [level-1 | level-2], 112
isis psnp-interval (*1-120*), 112
isis psnp-interval (1-120) [level-1 | level-2], 112
isis three-way-handshake, 112

L

l2rd NVE-ID-VALUE, 173
label vpn export (0..1048575)|auto, 71
labels LABEL-LIST, 175
line vty, 13
Link State Advertisement, 121
Link State Announcement, 121
Link State Database, 121
link-detect, 45
link-param ava-bw BANDWIDTH, 45
link-param delay (0-16777215) [min (0-16777215) | max (0-16777215)], 45
link-param delay-variation (*0-16777215*), 45
link-param neighbor <A.B.C.D> as (*0-65535*), 45
link-param no neighbor, 45
link-param packet-loss PERCENTAGE, 45
link-param res-bw BANDWIDTH, 45
link-param use-bw BANDWIDTH, 45
link-params, 45
link-params admin-grp BANDWIDTH, 45
link-params max-bw BANDWIDTH, 45
link-params max-rsv-bw BANDWIDTH, 45
link-params unrsv-bw (0-7) BANDWIDTH, 45

link-params [enable], 45
link-params [metric (0-4294967295)], 45
link-state, **193**
Link-state routing protocol, 121
Link-state routing protocol advantages, 121
Link-state routing protocol disadvantages, 121
Linux configurations, 8
list, 14
log commands, 13
log facility [FACILITY], 12
log file FILENAME [LEVEL], 12
log monitor [LEVEL], 12
log record-priority, 12
log stdout [LEVEL], 12
log syslog [LEVEL], 12
log timestamp precision (*0-6*), 12
log trap LEVEL, 12
log-adjacency-changes, 110
log-adjacency-changes [detail], 129
logical-network-id VALUE, 175
logmsg LEVEL MESSAGE, 14
LSA flooding, 121
lsp-gen-interval (*1-120*), 110
lsp-gen-interval [level-1 | level-2] (*1-120*), 110
lsp-refresh-interval [level-1 | level-2] (*1-65235*), 110

M

Mailing lists, 4
Making FRR, 5
match, 154
match as-path WORD, 63
match aspath AS_PATH, 28
match community COMMUNITY_LIST, 29
match community WORD, 65
match community WORD exact-match, 65
match extcommunity WORD, 69
match interface WORD, 160
match ip address ACCESS_LIST, 28
match ip address prefix-len 0-32, 28
match ip address PREFIX-LIST, 28
match ip address prefix-list WORD, 160
match ip address WORD, 160
match ip next-hop IPV4_ADDR, 28
match ip next-hop prefix-list WORD, 160
match ip next-hop WORD, 160
match ipv6 address ACCESS_LIST, 28
match ipv6 address prefix-len 0-128, 28
match ipv6 address PREFIX-LIST, 28
match large-community LINE, 70
match local-preference METRIC, 28
match metric (*0-4294967295*), 160
match metric METRIC, 28
match peer A.B.C.D|X:X::X:X, 87
match peer INTERFACE_NAME, 29
match peer IPV4_ADDR, 29
match peer IPV6_ADDR, 29
match rpki notfound|invalid|valid, 94
match source-instance NUMBER, 29
match source-protocol PROTOCOL_NAME, 29
match tag TAG, 28
Matching Conditions, **27**
Matching Policy, **27**
max-lsp-lifetime (*360-65535*), 110
max-lsp-lifetime [level-1 | level-2] (*360-65535*), 110
max-metric router-lsa administrative, 130
max-metric router-lsa [on-startup|on-shutdown] (*5-86400*), 130
metric-style [narrow | transition | wide], 110
Modifying the herd's behavior, 11
mpls-te inter-as area <area-id>|as, 137
mpls-te on, 113, 137
mpls-te router-address <A.B.C.D>, 113, 137
multicast, 44

N

neigbor A.B.C.D|X.X::X.X|peer-group route-map WORD import|export, 87
neighbor A.B.C.D, 156
neighbor A.B.C.D route-server-client, 87
neighbor PEER default-originate, 61
neighbor PEER description ..., 60
neighbor PEER distribute-list NAME [in|out], 62
neighbor PEER dont-capability-negotiate, 74
neighbor PEER ebgp-multihop, 60
neighbor PEER filter-list NAME [in|out], 62
neighbor PEER interface IFNAME, 61
neighbor PEER local-as AS-NUMBER, 61
neighbor PEER local-as AS-NUMBER no-prepend, 61
neighbor PEER local-as AS-NUMBER no-prepend replace-as, 61
neighbor PEER maximum-prefix NUMBER, 61
neighbor PEER next-hop-self [all], 61
neighbor PEER override-capability, 74
neighbor PEER peer-group WORD, 62

neighbor PEER port PORT, 61
neighbor PEER prefix-list NAME [in|out], 62
neighbor PEER remote-as ASN, 60
neighbor PEER remote-as external, 60
neighbor PEER remote-as internal, 60
neighbor PEER route-map NAME [in|out], 62
neighbor PEER route-reflector-client, 74
neighbor PEER send-community, 61
neighbor PEER shutdown, 60
neighbor PEER strict-capability-match, 73
neighbor PEER ttl-security hops NUMBER, 62
neighbor PEER update-source <IFNAME|ADDRESS>, 61
neighbor PEER version VERSION, 60
neighbor PEER weight WEIGHT, 61
neighbor PEER-GROUP route-server-client, 85
neighbor WORD peer-group, 62
neighbor X:X::X:X route-server-client, 87
net XX.XXXX. ... .XXX.XX, 109
netns NAMESPACE, 48
network A.B.C.D/M, 58
network A.B.C.D/M area (*0-4294967295*), 130
network A.B.C.D/M area A.B.C.D, 130
network IFNAME, 102, 156, 165
network NETWORK, 106, 156, 165
nexthop vpn export A.B.C.D|X:X::X:X, 71
nexthop-group, 153
no agentx, 39
no aggregate-address A.B.C.D/M, 59
no area (0-4294967295) authentication, 133
no area (0-4294967295) export-list NAME, 132
no area (0-4294967295) filter-list prefix NAME in, 133
no area (0-4294967295) filter-list prefix NAME out, 133
no area (0-4294967295) import-list NAME, 133
no area (0-4294967295) range A.B.C.D/M, 131
no area (0-4294967295) shortcut, 132
no area (0-4294967295) stub, 132
no area (0-4294967295) stub no-summary, 132
no area (0-4294967295) virtual-link A.B.C.D, 132
no area A.B.C.D authentication, 133
no area A.B.C.D default-cost (*0-16777215*), 132
no area A.B.C.D export-list NAME, 132
no area A.B.C.D filter-list prefix NAME in, 133
no area A.B.C.D filter-list prefix NAME out, 133
no area A.B.C.D import-list NAME, 133
no area A.B.C.D range A.B.C.D/M, 131
no area A.B.C.D range IPV4_PREFIX not-advertise, 131
no area A.B.C.D range IPV4_PREFIX substitute IPV4_PREFIX, 131
no area A.B.C.D shortcut, 132
no area A.B.C.D stub, 132
no area A.B.C.D stub no-summary, 132
no area A.B.C.D virtual-link A.B.C.D, 132
no area-password, 110
no auto-cost reference-bandwidth, 130, 144
no babel diversity, 101
no babel enable-timestamps, 103
no babel resend-delay [(20-655340)], 101
no babel split-horizon, 102
no bandwidth (*1-10000000*), 44
no banner motd, 13
no bgp multiple-instance, 74
no capability opaque, 137
no debug event, 73
no debug isis adj-packets, 113
no debug isis checksum-errors, 113
no debug isis events, 114
no debug isis local-updates, 114
no debug isis packet-dump, 114
no debug isis protocol-errors, 114
no debug isis route-events, 114
no debug isis snp-packets, 114
no debug isis spf-events, 114
no debug isis spf-statistics, 114
no debug isis spf-triggers, 114
no debug isis update-packets, 114
no debug keepalive, 73
no debug ospf event, 139
no debug ospf ism, 139
no debug ospf ism (*status|events|timers*), 139
no debug ospf lsa, 139
no debug ospf lsa (*generate|flooding|refresh*), 139
no debug ospf nsm, 139
no debug ospf nsm (*status|events|timers*), 139
no debug ospf nssa, 139
no debug ospf packet

(hello|dd|ls-request|ls-update|ls-ack|pdu)
(send|recv) [detail], 139
no debug ospf te, 139
no debug ospf zebra, 139
no debug ospf zebra (*interface|redistribute*), 139
no debug rpki, 94
no debug update, 73
no default-information originate, 136
no default-metric, 136
no default-metric (*1-16*), 159
no distance (*1-255*), 136, 159
no distance (1-255) A.B.C.D/M, 159
no distance (1-255) A.B.C.D/M ACCESS-LIST, 160
no distance ospf, 136
no distribute-list NAME out (kernel|connected|static|rip|ospf, 136
no domain-password, 110
no dump bgp all [PATH] [INTERVAL], 78
no dump bgp route-mrt [PATH] [INTERVAL], 78
no dump bgp updates [PATH] [INTERVAL], 78
no enable password PASSWORD, 12
no exec-timeout, 13
no hostname dynamic, 109
no import vrf VRFNAME, 72
no import|export vpn, 72
no ip address ADDRESS/PREFIX, 44
no ip address ADDRESS/PREFIX secondary, 44
no ip address LOCAL-ADDR peer PEER-ADDR/PREFIX, 44
no ip as-path access-list WORD, 63
no ip as-path access-list WORD permit|deny LINE, 63
no ip community-list expanded NAME, 64
no ip community-list NAME, 64
no ip community-list standard NAME, 64
no ip extcommunity-list expanded NAME, 68
no ip extcommunity-list NAME, 68
no ip extcommunity-list standard NAME, 68
no ip large-community-list expanded NAME, 69
no ip large-community-list NAME, 69
no ip large-community-list standard NAME, 69
no ip mroute PREFIX NEXTHOP [DISTANCE], 49
no ip multicast rpf-lookup-mode [MODE], 49
no ip nht resolve-via-default, 49
no ip ospf area, 135
no ip ospf area [ADDR], 133
no ip ospf authentication-key, 133
no ip ospf cost, 134
no ip ospf dead-interval, 134
no ip ospf hello-interval, 134
no ip ospf message-digest-key, 134
no ip ospf network, 134
no ip ospf priority, 134
no ip ospf retransmit interval, 135
no ip ospf transmit-delay, 135
no ip prefix-list NAME, 24
no ip prefix-list NAME description [DESC], 24
no ip prefix-list sequence-number, 24
no ip rip authentication key-chain KEY-CHAIN, 161
no ip rip authentication mode md5, 161
no ip rip authentication mode text, 161
no ip rip authentication string STRING, 161
no ip router isis WORD, 111
no ip split-horizon, 157
no ipv6 address ADDRESS/PREFIX, 44
no ipv6 nd adv-interval-option, 32
no ipv6 nd home-agent-config-flag, 32
no ipv6 nd home-agent-lifetime (*0-65520*), 32
no ipv6 nd home-agent-preference [(0-65535)], 32
no ipv6 nd managed-config-flag, 32
no ipv6 nd mtu [(1-65535)], 32
no ipv6 nd other-config-flag, 32
no ipv6 nd ra-interval [(1-1800)], 32
no ipv6 nd ra-interval [msec (70-1800000)], 32
no ipv6 nd ra-lifetime [(0-9000)], 32
no ipv6 nd reachable-time [(1-3600000)], 32
no ipv6 nd router-preference (*high|medium|low*), 32
no ipv6 nd suppress-ra, 31
no is-type, 111
no isis circuit-type, 111
no isis csnp-interval, 111
no isis csnp-interval [level-1 | level-2], 111
no isis hello-interval, 111
no isis hello-interval [level-1 | level-2], 111
no isis hello-multiplier, 111
no isis hello-multiplier [level-1 | level-2], 111
no isis metric, 112

no isis metric [level-1 | level-2], 112
no isis network point-to-point, 112
no isis passive, 112
no isis password, 112
no isis priority, 112
no isis priority [level-1 | level-2], 112
no isis psnp-interval, 112
no isis psnp-interval [level-1 | level-2], 112
no isis three-way-handshake, 112
no label vpn export [(0..1048575)|auto], 71
no labels LABEL-LIST, 175
no link-detect, 45
no link-param, 45
no log facility [FACILITY], 12
no log file [FILENAME [LEVEL]], 12
no log monitor [LEVEL], 12
no log record-priority, 12
no log stdout [LEVEL], 12
no log syslog [LEVEL], 12
no log trap [LEVEL], 12
no log-adjacency-changes, 110
no log-adjacency-changes [detail], 129
no lsp-gen-interval, 110
no lsp-gen-interval [level-1 | level-2], 110
no lsp-refresh-interval [level-1 | level-2], 110
no match rpki notfound|invalid|valid, 94
no max-lsp-lifetime, 110
no max-lsp-lifetime [level-1 | level-2], 110
no max-metric router-lsa [on-startup|on-shutdown|administrative], 130
no metric-style, 110
no mpls-te, 113, 137
no mpls-te inter-as, 137
no mpls-te router-address, 113
no multicast, 44
no neighbor A.B.C.D, 156
no neighbor PEER default-originate, 61
no neighbor PEER description ..., 60
no neighbor PEER dont-capability-negotiate, 74
no neighbor PEER ebgp-multihop, 60
no neighbor PEER interface IFNAME, 61
no neighbor PEER local-as, 61
no neighbor PEER maximum-prefix NUMBER, 61
no neighbor PEER next-hop-self [all], 61
no neighbor PEER override-capability, 74
no neighbor PEER route-reflector-client, 74
no neighbor PEER shutdown, 60
no neighbor PEER strict-capability-match, 73
no neighbor PEER ttl-security hops NUMBER, 62
no neighbor PEER update-source, 61
no neighbor PEER weight WEIGHT, 61
no net XX.XXXX. ... .XXX.XX, 109
no network A.B.C.D/M, 59
no network A.B.C.D/M area (*0-4294967295*), 130
no network A.B.C.D/M area A.B.C.D, 130
no network IFNAME, 102, 156
no network NETWORK, 106, 156
no nexthop vpn export [A.B.C.D|X:X::X:X], 71
no ospf abr-type TYPE, 129
no ospf opaque-lsa, 137
no ospf rfc1583compatibility, 129
no ospf router-id, 128
no passive-interface IFNAME, 106, 157
no passive-interface INTERFACE, 129
no password PASSWORD, 11
no pce address, 138
no pce domain as (*0-65535*), 138
no pce flag, 138
no pce neighbor as (*0-65535*), 138
no pce scope, 138
no rd vpn export [AS:NN|IP:nn], 71
no redistribute (*kernel|connected|static|rip|bgp*), 135
no redistribute <ipv4|ipv6> KIND, 103
no redistribute bgp, 107, 158
no redistribute connected, 107, 158
no redistribute kernel, 106, 158
no redistribute ospf, 107, 158
no redistribute static, 106, 158
no route A.B.C.D/M, 158
no route-map vpn import|export [MAP], 72
no router babel, 101
no router bgp ASN, 54
no router eigrp (*1-65535*), 106
no router isis WORD, 109
no router ospf, 128
no router rip, 156
no router zebra, 136
no router-info, 137

no rpki cache (A.B.C.D|WORD) [PORT] PREFERENCE, 93
no rpki initial-synchronisation-timeout, 93
no rpki polling_period, 93
no rpki timeout, 93
no rt vpn import|export|both [RTLIST...], 71
no service integrated-vtysh-config, 20
no set-overload-bit, 110
no shutdown, 44
no smux peer OID, 39
no smux peer OID PASSWORD, 39
no spf-interval, 110
no spf-interval [level-1 | level-2], 110
no timers basic, 162
no timers throttle spf, 129, 143
no version, 157
no vnc l2-group NAME, 175
no vnc nve-group NAME, 173
no vnc redistribute ipv4|ipv6 bgp|bgp-direct|bgp-direct-to-nve-groups|connected|kernel|ospf|rip|static, 177
no vnc redistribute nve-group GROUP-NAME, 177

O

offset-list ACCESS-LIST (*in|out*), 159
offset-list ACCESS-LIST (in|out) IFNAME, 159
on-match goto N, 30
on-match next, 30
Operating systems that support FRR, 2
Options for configuring, 5
Options to `./configure`, 5
ospf abr-type TYPE, 128
OSPF Areas overview, 122
OSPF Hello Protocol, 122
OSPF LSA overview, 122
ospf opaque-lsa, 137
ospf rfc1583compatibility, 129
ospf router-id A.B.C.D, 128

P

passive-interface (*IFNAME|default*), 106, 157
passive-interface INTERFACE, 129
password PASSWORD, 11
PBR Rules, 154
PBR Tables, 154
pbr-map, 154
pbr-policy, 154
pce address <A.B.C.D>, 138
pce domain as (*0-65535*), 138
pce flag BITPATTERN, 138
pce neighbor as (*0-65535*), 138
pce scope BITPATTERN, 138
prefix vn|un A.B.C.D/M|X:X::X:X/M, 173

R

rd ROUTE-DISTINGUISHER, 173
rd vpn export AS:NN|IP:nn, 71
redistribute (*kernel|connected|static|rip|bgp*), 135
redistribute (kernel|connected|static|rip|bgp) metric (*0-16777214*), 135
redistribute (kernel|connected|static|rip|bgp) metric (0-16777214) route-map WORD, 135
redistribute (kernel|connected|static|rip|bgp) metric-type (*1|2*), 135
redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) metric (*0-16777214*), 135
redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) metric (0-16777214) route-map WORD, 135
redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) route-map WORD, 135
redistribute (kernel|connected|static|rip|bgp) ROUTE-MAP, 135
redistribute <ipv4|ipv6> KIND, 103
redistribute bgp, 107, 158
redistribute bgp metric (1-4294967295) (0-4294967295) (0-255) (1-255) (*1-65535*), 107
redistribute bgp metric (*0-16*), 158
redistribute bgp route-map ROUTE-MAP, 158
redistribute connected, 59, 107, 144, 158
redistribute connected metric (*0-16*), 158
redistribute connected metric (1-4294967295) (0-4294967295) (0-255) (1-255) (*1-65535*), 107
redistribute connected route-map ROUTE-MAP, 158
redistribute kernel, 59, 106, 158
redistribute kernel metric (*0-16*), 158
redistribute kernel metric (1-4294967295) (0-4294967295) (0-255) (1-255) (*1-65535*), 106
redistribute kernel route-map ROUTE-MAP, 158
redistribute ospf, 59, 107, 158
redistribute ospf metric (*0-16*), 158
redistribute ospf metric (1-4294967295) (0-4294967295) (0-255) (1-255) (*1-65535*), 107

redistribute ospf route-map ROUTE-MAP, 158
redistribute rip, 59
redistribute ripng, 144
redistribute static, 59, 106, 144, 158
redistribute static metric (*0-16*), 158
redistribute static metric (1-4294967295) (0-4294967295) (0-255) (1-255) (*1-65535*), 106
redistribute static route-map ROUTE-MAP, 158
redistribute vpn, 59
Reporting bugs, 4
Reporting software errors, 4
response-lifetime LIFETIME|infinite, 173
RFC
    RFC 1058, 3
    RFC 1195, 109
    RFC 1227, 4, 37
    RFC 1583, 129
    RFC 1657, 4
    RFC 1724, 4
    RFC 1771, 3, 53, 85
    RFC 1850, 4
    RFC 1930, 63
    RFC 1965, 3
    RFC 1997, 3, 63
    RFC 1998, 63
    RFC 2080, 3, 165
    RFC 2082, 3
    RFC 2283, 73
    RFC 2328, 3, 121, 129
    RFC 2370, 3, 137
    RFC 2439, 55
    RFC 2453, 3
    RFC 2462, 33
    RFC 2545, 3
    RFC 2740, 3, 143
    RFC 2741, 4, 37
    RFC 2796, 3
    RFC 2842, 3, 73
    RFC 2858, 3, 53
    RFC 3101, 3
    RFC 3137, 3, 130
    RFC 3345, 57
    RFC 3509, 129, 132
    RFC 4191, 33
    RFC 4364, 62, 171
    RFC 4659, 62, 171
    RFC 4861, 33
    RFC 4970, 137
    RFC 5088, 138
    RFC 5303, 112
    RFC 5308, 109
    RFC 5392, 137
    RFC 5512, 62, 171
    RFC 5575, 96, 100
    RFC 6126, 101
    RFC 6275, 33
    RFC 6810, 92
    RFC 6811, 92
    RFC 7432, 175
    RFC 8092, 69
route A.B.C.D/M, 158
route NETWORK, 165
route-map ROUTE-MAP-NAME (permit|deny) ORDER, 28
route-map vpn import|export MAP, 71
router babel, 101
router bgp AS-NUMBER, 75
router bgp AS-NUMBER view NAME, 76
router bgp ASN, 53
router bgp ASN vrf VRFNAME, 70
router eigrp (*1-65535*), 106
router isis WORD, 109
router ospf, 128
router ospf6, 143
router rip, 156
router ripng, 165
router zebra, 136, 165
router-id A.B.C.D, 143
router-info [as | area <A.B.C.D>], 137
rpki, 93
rpki cache (A.B.C.D|WORD) PORT [SSH_USERNAME] [SSH_PRIVKEY_PATH] [SSH_PUBKEY_PATH] [KNOWN_HOSTS_PATH] PREFERENCE, 93
rpki initial-synchronisation-timeout <1-4, 93
rpki polling_period (*1-3600*), 93
rpki timeout <1-4, 93
rt both RT-LIST, 174
rt both RT-TARGET, 175
rt export RT-LIST, 174
rt export RT-TARGET, 175
rt import RT-LIST, 174
rt import RT-TARGET, 175
rt vpn import|export|both RTLIST..., 71

S

service advanced-vty, 13
service integrated-vtysh-config, 20
service password-encryption, 13
service terminal-length (*0-512*), 13
Set Actions, **27**
set as-path prepend AS-PATH, 63
set as-path prepend AS_PATH, 30

set as-path prepend last-as NUM, 63
set comm-list WORD delete, 65
set community COMMUNITY, 30, 65
set community COMMUNITY additive, 65
set community none, 65
set extcommunity rt EXTCOMMUNITY, 69
set extcommunity soo EXTCOMMUNITY, 69
set ip next-hop A.B.C.D, 160
set ip next-hop IPV4_ADDRESS, 29
set ip next-hop peer-address, 29
set ip next-hop unchanged, 29
set ipv6 next-hop global IPV6_ADDRESS, 29
set ipv6 next-hop local IPV6_ADDRESS, 30
set ipv6 next-hop peer-address, 29
set ipv6 next-hop prefer-global, 29
set large-community LARGE-COMMUNITY, 70
set large-community LARGE-COMMUNITY additive, 70
set large-community LARGE-COMMUNITY LARGE-COMMUNITY, 70
set local-preference LOCAL_PREF, 29
set metric (*0-4294967295*), 160
set metric METRIC, 29
set src ADDRESS, 50
set tag TAG, 29
set weight WEIGHT, 29
set-overload-bit, 110
show babel interface, 103
show babel interface IFNAME, 103
show babel neighbor, 103
show babel parameters, 103
show babel route, 103
show babel route A.B.C.D, 103
show babel route A.B.C.D/M, 103
show babel route X:X::X:X, 103
show babel route X:X::X:X/M, 103
show bgp ipv4 flowspec [detail | A.B.C.D], 97
show bgp ipv4 vpn summary, 62
show bgp ipv4|ipv6 community, 66
show bgp ipv4|ipv6 community COMMUNITY, 66
show bgp ipv4|ipv6 community COMMUNITY exact-match, 66
show bgp ipv4|ipv6 community-list WORD, 66
show bgp ipv4|ipv6 community-list WORD exact-match, 66
show bgp ipv4|ipv6 dampening dampened-paths, 73
show bgp ipv4|ipv6 dampening flap-statistics, 73
show bgp ipv4|ipv6 neighbor [PEER], 72
show bgp ipv4|ipv6 regexp LINE, 63
show bgp ipv4|ipv6 summary, 72
show bgp ipv6 vpn summary, 62
show debug, 73
show debugging eigrp, 108
show debugging isis, 114
show debugging ospf, 139
show debugging rip, 163
show debugging ripng, 166
show interface, 51
show ip bgp, 72
show ip bgp A.B.C.D, 72
show ip bgp community COMMUNITY, 72
show ip bgp community COMMUNITY exact-match, 72
show ip bgp community-list WORD, 72
show ip bgp community-list WORD exact-match, 72
show ip bgp ipv4 vpn, 62
show ip bgp large-community-info, 70
show ip bgp regexp LINE, 72
show ip bgp view NAME, 76
show ip bgp X:X::X:X, 72
show ip community-list, 64
show ip community-list NAME, 64
show ip eigrp topology, 107
show ip extcommunity-list, 68
show ip extcommunity-list NAME, 69
show ip large-community-list, 70
show ip large-community-list NAME, 70
show ip mroute, 150
show ip mroute count, 150
show ip multicast, 150
show ip ospf, 136
show ip ospf database, 136
show ip ospf database (*asbr-summary|external|network|router|summary*), 136
show ip ospf database (asbr-summary|external|network|router|summar
show ip ospf database (asbr-summary|external|network|router|summar adv-router ADV-ROUTER, 136
show ip ospf database (asbr-summary|external|network|router|summar LINK-STATE-ID, 136
show ip ospf database (asbr-summary|external|network|router|summar LINK-STATE-ID adv-router ADV-ROUTER, 136
show ip ospf database (asbr-summary|external|network|router|summar LINK-STATE-ID self-originate, 136
show ip ospf database (asbr-summary|external|network|router|summar

self-originate, 136
show ip ospf database (*opaque-link|opaque-area|opaque-external*), 137
show ip ospf database (opaque-link|opaque-area|opaque-external) adv-router ADV-ROUTER, 137
show ip ospf database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID, 137
show ip ospf database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID adv-router ADV-ROUTER, 137
show ip ospf database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID self-originate, 137
show ip ospf database (opaque-link|opaque-area|opaque-external) self-originate, 137
show ip ospf database max-age, 136
show ip ospf database segment-routing <adv-router ADVROUTER|self-originate> [json], 138
show ip ospf database self-originate, 136
show ip ospf interface [INTERFACE], 136
show ip ospf mpls-te interface, 137
show ip ospf mpls-te interface INTERFACE, 137
show ip ospf mpls-te router, 137
show ip ospf neighbor, 136
show ip ospf neighbor detail, 136
show ip ospf neighbor INTERFACE, 136
show ip ospf neighbor INTERFACE detail, 136
show ip ospf route, 136
show ip ospf router-info, 138
show ip ospf router-info pce, 138
show ip pim assert, 150
show ip pim assert-internal, 150
show ip pim assert-metric, 150
show ip pim assert-winner-metric, 150
show ip pim group-type, 150
show ip pim interface, 150
show ip pim join, 150
show ip pim local-membership, 150
show ip pim neighbor, 150
show ip pim nexthop, 150
show ip pim nexthop-lookup, 150
show ip pim rp-info, 150
show ip pim rpf, 150
show ip pim secondary, 150
show ip pim state, 150
show ip pim upstream, 150
show ip pim upstream-join-desired, 150
show ip pim upstream-rpf, 150
show ip prefix-list, 24
show ip prefix-list detail, 25
show ip prefix-list detail NAME, 25
show ip prefix-list NAME, 24
show ip prefix-list NAME A.B.C.D/M, 24
show ip prefix-list NAME A.B.C.D/M first-match, 24
show ip prefix-list NAME A.B.C.D/M longer, 24
show ip prefix-list NAME seq NUM, 24
show ip prefix-list summary, 25
show ip prefix-list summary NAME, 25
show ip prefix-list [NAME], 51
show ip protocol, 51
show ip rip, 162
show ip rip status, 162
show ip ripng, 166
show ip route, 51
show ip route isis, 113
show ip route table TABLEID, 98
show ip route vrf VRF, 48
show ip route vrf VRF table TABLENO, 48
show ip rpf, 49, 151
show ip rpf ADDR, 49
show ipforward, 52
show ipv6 bgp ipv6 vpn, 62
show ipv6 ospf6 database, 145
show ipv6 ospf6 interface, 145
show ipv6 ospf6 neighbor, 145
show ipv6 ospf6 request-list A.B.C.D, 145
show ipv6 ospf6 zebra, 145
show ipv6 ospf6 [INSTANCE_ID], 145
show ipv6 route, 51
show ipv6 route ospf6, 145
show ipv6forward, 52
show isis database, 113
show isis database <LSP id> [detail], 113
show isis database detail <LSP id>, 113
show isis database [detail], 113
show isis hostname, 112
show isis interface, 112
show isis interface <interface name>, 112
show isis interface detail, 112
show isis mpls-te interface, 113
show isis mpls-te interface INTERFACE, 113
show isis mpls-te router, 113
show isis neighbor, 113

show isis neighbor <System Id>, 113
show isis neighbor detail, 113
show isis summary, 112
show isis topology, 113
show isis topology [level-1|level-2], 113
show logging, 14
show memory vnc, 180
show pbr ipset IPSETNAME | iptable, 98
show route-map [NAME], 51
show rpki cache-connection, 94
show rpki prefix-table, 94
show version, 14
show vnc nves, 180
show vnc nves vn|un ADDRESS, 180
show vnc queries, 180
show vnc queries PREFIX, 180
show vnc registrations [all|local|remote|holddown|imported], 180
show vnc registrations [all|local|remote|holddown|imported] PREFIX, 180
show vnc responses [active|removed], 180
show vnc responses [active|removed] PREFIX, 180
show vnc summary, 180
show zebra, 52
show zebra fpm stats, 52
shutdown, 44
simple: debug babel KIND, 103
simple: no debug babel KIND, 103
smux peer OID, 39
smux peer OID PASSWORD, 39
Software architecture, 2
Software internals, 2
spf-interval (*1-120*), 110
spf-interval [level-1 | level-2] (*1-120*), 110
Supported platforms, 2
System architecture, 2

T

table TABLENO, 47
table-map ROUTE-MAP-NAME, 59
terminal length (*0-512*), 14
timers basic UPDATE TIMEOUT GARBAGE, 161
timers throttle spf DELAY INITIAL-HOLDTIME MAX-HOLDTIME, 129, 143

U

update-delay MAX-DELAY, 59
update-delay MAX-DELAY ESTABLISH-WAIT, 59
username USERNAME nopassword, 20

V

version VERSION, 157
vnc export bgp|zebra group-nve group GROUP-NAME, 178
vnc export bgp|zebra group-nve no group GROUP-NAME, 178
vnc l2-group NAME, 175
vnc nve-group NAME, 173
vnc redistribute bgp-direct (ipv4|ipv6) prefix-list LIST-NAME, 177
vnc redistribute bgp-direct no (ipv4|ipv6) prefix-list, 177
vnc redistribute bgp-direct no route-map, 178
vnc redistribute bgp-direct route-map MAP-NAME, 178
vnc redistribute ipv4|ipv6 bgp-direct-to-nve-groups view VIEWNAME, 177
vnc redistribute ipv4|ipv6 bgp|bgp-direct|ipv6 bgp-direct-to-nve-groups|connected|kernel|os 177
vnc redistribute lifetime LIFETIME|infinite, 177
vnc redistribute mode plain|nve-group|resolve-nve, 177
vnc redistribute nve-group GROUP-NAME, 177
vnc redistribute resolve-nve roo-ec-local-admin 0-65536, 177
vrf VRF, 48

W

who, 14
write file, 14
write integrated, 21
write terminal, 14

Z

zebra command line option
-b, -batch, 43
-e X, -ecmp X, 43
-k, -keep_kernel, 43
-n, -vrfwnetns, 43
-r, -retain, 43