See also:

PBR Details

3.16.5 PBR Debugs

debug pbr events|map|nht|zebra

Debug pbr in pbrd daemon. You specify what types of debugs to turn on.

3.16.6 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.

3.16.7 Sample configuration

```
nexthop-group TEST
  nexthop 4.5.6.7
  nexthop 5.6.7.8
!
pbr-map BLUE seq 100
  match dst-ip 9.9.9.0/24
  match src-ip 10.10.10.0/24
  set nexthop-group TEST
!
int swp1
  pbr-policy BLUE
```

3.17 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.

3.17.1 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/frr/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*).

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.

3.17.2 RIP Configuration

`router rip [vrf NAME]`

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.

`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`

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`

Specify a RIP neighbor to send updates to. This is required when a neighbor is connected via a network that does not support multicast, or when it is desired to statically define a neighbor. RIP updates will be sent via unicast to each neighbour. Neighbour updates are in addition to any multicast updates sent when an interface is not in passive mode (see the *passive-interface* command). RIP will continue to process updates received from both the neighbor and any received via multicast. 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)`

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 [poisoned-reverse]`

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*.

If *poisoned-reverse* is also set, the router sends the poisoned routes with highest metric back to the sending router.

`allow-ecmp [1-MULTIPATH_NUM]`

Control how many ECMP paths RIP can inject for the same prefix. If specified without a number, a maximum is taken (compiled with `--enable-multipath`).

3.17.3 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 `2`.

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.

`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).

3.17.4 How to Announce RIP route

`redistribute <babel|bgp|connected|eigrp|isis|kernel|openfabric|ospf|sharp|static|table> [metric (0-16)]`

Redistribute routes from other sources into RIP.

If you want to specify RIP only static routes:

`default-information originate`

`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*.

3.17.5 Filtering RIP Routes

RIP routes can be filtered by a distribute-list.

`distribute-list [prefix] LIST <in|out> IFNAME`

You can apply access lists to the interface with a *distribute-list* command. If prefix is specified LIST is a prefix-list. If prefix is not specified then LIST is the access list name. *in* specifies packets being received, and *out* specifies outgoing packets. Finally if an interface is specified it will be applied against a specific interface.

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.

3.17.6 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)`

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`

3.17.7 RIP distance

Distance value is used in zebra daemon. Default RIP distance is 120.

`distance (1-255)`

Set default RIP distance to specified value.

`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`

Set default RIP distance to specified value when the route's source IP address matches the specified prefix and the specified access-list.

3.17.8 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.

3.17.9 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
Set the interface with RIPv2 MD5 authentication.

ip rip authentication mode text
Set the interface with RIPv2 simple password authentication.

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
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
!
```

3.17.10 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.

3.17.11 Show RIP Information

To display RIP routes.

`show ip rip [vrf NAME]`

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 [vrf NAME] 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
```

3.17.12 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.

3.17.13 Sample configuration

```
debug rip events
debug rip packet

router rip
 network 11.0.0.0/8
 network eth0
 route 10.0.0.0/8
 distribute-list private-only in eth0

access-list private-only permit 10.0.0.0/8
access-list private-only deny any
```

3.18 RIPng

ripngd supports the RIPng protocol as described in **RFC 2080**. It's an IPv6 reincarnation of the RIP protocol.

3.18.1 Invoking ripngd

There are no *ripngd* specific invocation options. Common options can be specified (*Common Invocation Options*).

3.18.2 ripngd Configuration

Currently ripngd supports the following commands:

router ripng [vrf NAME]
Enable RIPng.

network NETWORK
Set RIPng enabled interface by NETWORK.

network IFNAME
Set RIPng enabled interface by IFNAME.