Policy Routing

Overview

Policy routing is the method to steer traffic matching certain criteria to a certain gateway. This can be used to force some customers or specific protocols from the servers (for example HTTP traffic) to always be routed to a certain gateway. It can even be used to steer local and overseas traffic to different gateways.

RouterOS implements several components that can be used to achieve said task:

- routing tables
- routing rules
- firewall mangle marking

Routing Tables

A router can have multiple routing tables with its own set of routes routing the same destination to different gateways.

Tables can be seen and configured from the `/routing/table` menu.

By default, RouterOS has only the '**main**' routing table:

```
[admin@rack1_b33_CCR1036] /routing/table> print
Flags: D - dynamic; X - disabled, I - invalid; U - used
0 D name="main" fib
```

If a custom routing table is required, it should be defined in this menu prior to using it anywhere in the configuration.

Let's consider a basic example where we have two gateways 172.16.1.1 and 172.16.2.1 and we want to resolve 8.8.8.8 only in the routing table named '**my Table**' to the gateway 172.16.2.1:

```
/routing table add name=myTable fib
/ip route add dst-address=8.8.8.8 gateway=172.16.1.1
/ip route add dst-address=8.8.8.8 gateway=172.16.2.1@main routing-table=myTable
```


For a user-created table to be able to resolve the destination, the main routing table should be able to resolve the destination too.

In our example, the **main** routing table should also have a route to destination 8.8.8.8 or at least a default route, since the default route is dynamically added by the DHCP for safety reasons it is better to add 8.8.8.8 also in the main table.

```
[admin@rack1_b33_CCR1036] /ip/route> print detail Flags: D - dynamic; X - disabled, I - inactive, A - active;
c - connect, s - static, r - rip, b - bgp, o - ospf, d - dhcp, v - vpn, m - modem, y - cop
y;
H - hw-offloaded; + - ecmp
   DAd   dst-address=0.0.0.0/0 routing-table=main pref-src="" gateway=172.16.1.1
         immediate-gw=172.16.1.1%ether8 distance=1 scope=30 target-scope=10
         vrf-interface=ether8 suppress-hw-offload=no

 0  As   dst-address=8.8.8.8/32 routing-table=main pref-src="" gateway=172.16.1.1
         immediate-gw=172.16.1.1%ether8 distance=1 scope=30 target-scope=10 suppress-hw-offload=no

    DAc   dst-address=172.16.1.0/24 routing-table=main gateway=ether8 immediate-gw=ether8
         distance=0 scope=10 suppress-hw-offload=no local-address=172.16.1.2%ether8

    DAc   dst-address=172.16.2.0/24 routing-table=main gateway=ether7 immediate-gw=ether7
         distance=0 scope=10 suppress-hw-offload=no local-address=172.16.2.2%ether7

 1  As   dst-address=8.8.8.8/32 routing-table=myTable pref-src="" gateway=172.16.2.1
         immediate-gw=172.16.2.1%ether7 distance=1 scope=30 target-scope=10 suppress-hw-offload=no
```

But configuration above is not enough, we need a method to force the traffic to actually use our newly created table. RouterOS gives you two options to choose from:

- firewall mangle - it gives more control over the criteria to be used to steer traffic, for example, per connection or per packet balancing, etc. For more info on how to use mangle marking see Firewall Marking examples.
- routing rules - a basic set of parameters that can be used to quickly steer traffic. This is the method we are going to use for our example.

It is not recommended to use both methods at the same time or you should know exactly what you are doing. If you really do need to use both mangle and routing rules in the same setup then keep in mind that mangle has higher priority, meaning if the mangle marked traffic can be resolved in the table then route rules will never see this traffic.

Routing table count is limited to 4096 unique tables.

Routing Rules

Routing rules allow steering traffic based on basic parameters like a source address, a destination address, or in-interface as well as other parameters.

For our example, we want to select traffic with destination 8.8.8.8 and do not fall back to the **main** table:

```
/routing rule add dst-address=8.8.8.8 action=lookup-only-in-table table=myTable
```

Lets's say that we know that customer is connected to ether4 and we want only that customer to route 8.8.8.8 to a specific gateway. We can use the following rule:

```
/routing rule add dst-address=8.8.8.8 action=lookup-only-in-table table=myTable interface=ether4
```

If for some reason the gateway used in our table goes down, the whole lockup will fail and the destination will not be reachable. In active-backup setups we want the traffic to be able to fall back to the **main** table. To do that change the action from `lookup-only-in-table` to `lookup`.

Also, routing rules can be used as a very "basic firewall". Let's say we do not want to allow a customer connected to ether4 to be able to access the 192.168.1.0/24 network:

```
/routing rule add dst-address=192.168.1.0/24 interface=ether4 action=drop
```

List of all the parameters that can be used by routing rules:

Property	Description
action (*drop* | *lookup* | *lookup-only-in-table* | *unreachable*)	An action to take on the matching packet: • drop - silently drop the packet. • lookup - perform a lookup in routing tables. • lookup-only-in-table - perform lookup only in the specified routing table (see table parameter). • unreachable - generate ICMP unreachable message and send it back to the source.
comment (*string*)	
disabled (*yes* | *no*)	The disabled rule is not used.
dst-address()	The destination address of the packet to match.
interface (*string*)	Incoming interface to match.
min-prefix (*integer [0..4294967295]*)	Equivalent to Linux IP rule `suppress_prefixlength` . For example to suppress the default route in the routing decision set the value to 0.
routing-mark (*string*)	Match specific routing mark.
src-address (*string*)	The source address of the packet to match.
table (*name*)	Name of the routing table to use for lookup.