Failover (WAN Backup)

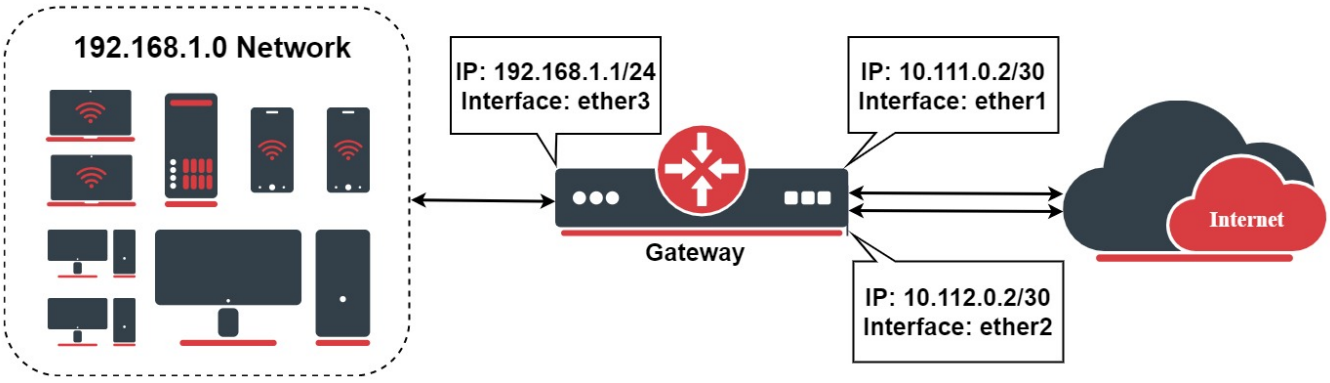
Do not forget to add routes with routing marks

Introduction

In this article, we will look at another advanced method of failover using recursive routing and scopes from the routing section. Recursive routing occurs when a route (either static or dynamically learned) has a next-hop that is not directly connected to the local router. In RouterOS, nexthop lookup is done only in the main routing table, even for routes with different values of routing-mark. It is necessary to restrict a set of routes that can be used to look up immediate next-hops. Nexthop values of RIP or OSPF routes, for example, are supposed to be directly reachable and should be looked up only using connected routes. This is achieved using a scope and target-scope properties.

Setup Overview

Let`s assume we have the following setup: our gateway device has two public network uplinks. Then we mark traffic in two parts, one with the name "ISP1" and the second as "ISP2" which goes through the ether1 and ether2 accordingly. In this setup, we want to monitor two hosts: Host1 and Host2. We will use Google DNS servers with IP 8.8.8.8 (Host1) and 8.8.4.4 (Host2), but it is not mandatory to use specifically these addresses.



Before a detailed example overview, in a setup where we have private IP addresses behind the public IP, we should configure source NAT:

```
/ip/firewall/nat
add chain=srcnat action=masquerade out-interface=ether1
add chain=srcnat action=masquerade out-interface=ether2
```

Let`s start with marking traffic by configuring routing tables and firewall mangle rules, so we will have everything preconfigured when we go to the routing section:

```
/routing/table
add fib name=to_ISP1
add fib name=to_ISP2

/ip/firewall/mangle
add chain=output connection-state=new connection-mark=no-mark action=mark-connection new-connection-
mark=ISP1_conn out-interface=ether1
add chain=output connection-mark=ISP1_conn action=mark-routing new-routing-mark=to_ISP1 out-interface=ether1
add chain=output connection-state=new connection-mark=no-mark action=mark-connection new-connection-
mark=ISP2_conn out-interface=ether2
add chain=output connection-mark=ISP2_conn action=mark-routing new-routing-mark=to_ISP2 out-interface=ether2
```

We will split the routing configuration into three parts. First, we will configure Host1 and Host2 as destination addresses in the routing section:

```
/ip/route/
add dst-address=8.8.8.8 scope=10 gateway=10.111.0.1
add dst-address=8.8.4.4 scope=10 gateway=10.112.0.1
```

Now configure routes that will be resolved recursively, so they will only be active when they are reachable with ping:

```
/ip/route/
add distance=1 gateway=8.8.8.8 routing-table=to_ISP1 check-gateway=ping
add distance=2 gateway=8.8.4.4 routing-table=to_ISP1 check-gateway=ping
```

Configure similar recursive routes for the second gateway:

```
/ip/route/
add distance=1 gateway=8.8.4.4 routing-table=to_ISP2 check-gateway=ping
add distance=2 gateway=8.8.8.8 routing-table=to_ISP2 check-gateway=ping
```

Adding Multiple Hosts

In the case where Host1 and Host2 fail, the corresponding link is considered failed too. In this section, we will use two additional hosts for redundancy. In our example, we will use OpenDNS servers Host1B (208.67.222.222) and Host2B (208.67.220.220):

```
/ip/route
add dst-address=8.8.8.8 gateway=10.111.0.1 scope=10
add dst-address=208.67.222.222 gateway=10.111.0.1 scope=10
add dst-address=8.8.4.4 gateway=10.112.0.1 scope=10
add dst-address=208.67.220.220 gateway=10.112.0.1 scope=10
```

Then, let's create destinations for "virtual" hops to use in further routes. We will use 10.10.10.1 and 10.20.20.2 as an example, but you can use different ones, be sure they do not override other configured IP addresses in your setup:

```
/ip/route
add dst-address=10.10.10.1 gateway=8.8.8.8 scope=10 target-scope=11 check-gateway=ping
add dst-address=10.10.10.1 gateway=208.67.222.222 scope=10 target-scope=11 check-gateway=ping
add dst-address=10.20.20.2 gateway=8.8.4.4 scope=10 target-scope=11 check-gateway=ping
add dst-address=10.20.20.2 gateway=208.67.220.220 scope=10 target-scope=11 check-gateway=ping
```

Do not forget to add routes with routing marks:

```
/ip/route
add distance=1 gateway=10.10.10.1 routing-table=to_ISP1
add distance=2 gateway=10.20.20.2 routing-table=to_ISP1
add distance=1 gateway=10.20.20.2 routing-table=to_ISP2
add distance=2 gateway=10.10.10.1 routing-table=to_ISP2
```