Manual:IPv6 Overview

From MikroTik Wiki

Contents

Applies to RouterOS: **v3beta10+, v4, v5+**

IPv6 overview

Package requirement: `ipv6`

Internet Protocol version 6 (IPv6) is the new version of the Internet Protocol (IP). It was initially expected to replace IPv4 in short enough time, but for now it seems that these two version will coexist in Internet in foreseeable future. Nevertheless, IPv6 becomes more important, as the date of unallocated IPv4 address pool's exhaustion approaches.

The two main benefits of IPv6 over IPv4 are:

- much larger address space;
- support of stateless and statefull address autoconfiguration;
- built-in security;
- new header format (faster forwarding).

Supported programms

MikroTik IPv6 support at the moment:

- DHCPv6 prefix delegation for DHCP server.
- DHCPv6-PD client.
- IPv6 Prefix Delegation over PPP interfaces.
- static addressing and routing;
- router advertisement daemon (for address autoconfiguration);
- dynamic routing: BGP+, OSPFv3, and RIPng protocols;
- firewall (filter, mangle, address lists, connection table);
- queue tree, simple queue, pcq;
- DNS name servers;
- 6in4 (SIT) tunnels;
- EoIPv6, ip/ipv6 over ipv6 (IPIPv6) tunnel interface (starting from v5RC6)
- IPSEC;
- VRRPv3;
- IPv6 forwarding over all PPP (Point-to-point protocols);
- SSH, telnet, FTP, WWW access, Winbox, API;
- ping;
- traceroute;
- web proxy;
- sniffer and fetch tools;
- IP services and User allowed IPv6 address support;
- torch, bandwidth test and other tools;

Features not yet supported:

- automatic tunnel creation;
- policy routing;
- multicast routing;
- MPLS;

Addressing

IPv6 uses 16 bytes addresses compared to 4 byte addresses in IPv4. IPv6 address syntax and types are described in RFC 4291.

Read more>>

Stateless Autoconfiguration

Read more >>

Routing

For static routing, the basic principles of IPv6 are exactly the same as for IPv4. `Read more >>`

Note: Link local addresses are required for dynamic routing protocols to function!

Warning: All dynamic routing protocols also require a valid Router ID to function. If the Router ID is not configured manually, one of router's IPv4 addresses are used as the Router ID. If no IPv4 addresses are present, the router ID selection process will fail. This means that dynamic routing will not work on a router that has no IPv4 addresses, unless you configure the Router ID manually!

BGP

Because of it's design BGP naturally supports multiple address families, and migration to IPv6 is straightforward here.

Example: configure iBGP between routers A and B, AS 65000, that will exchange IPv4 and IPv6 routes.

Router A:

```
[admin@A] > routing bgp peer add remote-address=10.0.0.134 remote-as=65000 address-families=ip,ipv6
```

Router B:

```
[admin@B] > routing bgp peer add remote-address=10.0.0.133 remote-as=65000 address-families=ip,ipv6
```

Redistribute a route from router A to router B:

```
[admin@A] > ipv6 route add dst-address=2001::/16 gateway=fe80::1%ether1
[admin@A] > routing bgp network add network=2001::/16
[admin@A] > routing bgp advertisements print
PEER     PREFIX               NEXTHOP          AS-PATH  ORIGIN     LOCAL-PREF
peer1    2001::/16            fe80::1200:ff...          igp        100
```

```
[admin@B] > ipv6 route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 #      DST-ADDRESS              GATEWAY                  DISTANCE
 0 ADb  2001::/16                fe80::1200:ff:fe00:10... 200
```

IPv6 addresses can also be used in peer configuration in **remote-address** and **update-source** fields - to make a BGP connection over IPv6.

OSPF

Unlike to BGP, adding IPv6 support to OSPF required a lot of changes and resulted in a new, incompatible, version of OSPF: protocol version 3. (For IPv4, OSPF version 2 is used). The new version is described in RFC 2740.

OSPFv3 uses the same fundamental mechanisms as OSPFv2 — LSAs, flooding, the SPF algorithm, etc. However, it adds not only support to a new address family, but also some improvements to the protocol itself. The new version avoids some potential problems and inefficiencies present in the operation of OSPFv2.

OSPFv3 configuration syntax largely remains the same as for OSPFv2. One mayor difference is that there is no configuration for networks anymore, and interface configuration becomes mandatory, since OSPFv3 runs on link, not IP subnet, basis.

Example:

Configure OSPF on router A:

```
[admin@A] > routing ospf-v3 interface add interface=ether1 area=backbone
```

Configure OSPF on router B:

```
[admin@B] > routing ospf-v3 interface add interface=ether1 area=backbone
```

Redistribute a route from router A to router B:

```
[admin@A] > ipv6 route add dst-address=2001::/16 gateway=fe80::1%ether1
[admin@A] > routing ospf-v3 instance set default redistribute-static=as-type-1
[admin@A] > routing ospf-v3 route print
 # DESTINATION                                STATE          COST
 0 2001::/16                                  imported-ext-1 20
```

```
[admin@B] > ipv6 route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 #      DST-ADDRESS              GATEWAY                 DISTANCE
 0 ADo  2001::/16                fe80::1200:ff:fe00:10... 110
```

RIP

Similarly to OSPF, a new version of RIP was required to add IPv6 support. The new version is called RIPng (RIP new generation) and described in RFC 2080. Just like OSPFv3, RIPng runs on link, not IP subnet, basis - this means that you need to configure interfaces, not IP networks, on which to run RIPng.

Example:

Configure RIP on router A:

```
[admin@A] > routing ripng interface add interface=ether1
```

Configure RIP on router B:

```
[admin@B] > routing ripng interface add interface=ether1
```

Redistribute a route from router A to router B:

```
[admin@A] > ipv6 route add dst-address=2001::/16 gateway=fe80::1%ether1
[admin@A] > routing ripng set redistribute-static=yes
[admin@A] > routing ripng route print
Flags: C - connect, S - static, R - rip, O - ospf, B - bgp
 #   DST-ADDRESS
 0 S 2001::/16
```

```
[admin@B] > ipv6 route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 #      DST-ADDRESS              GATEWAY                  DISTANCE
 0 ADr  2001::/16                fe80::1200:ff:fe00:10... 120
```

6to4 (6in4) tunnels

This describes solution using global 6to4 relay address. For a solution using a tunnel broker see Setting up an IPv6 tunnel via a tunnel broker.

First, you will need a global routable IPv4 address. We assume the address 1.2.3.4 for the sake of this example.

Then you need to make user that the global 6to4 relay anycast address 192.88.99.1 is reachable and that it really provides relay services (since it's anycast address, your connection should be routed to the host having this addresses that is the closest to your location).

Then add *6to4* interface without specifying remote address and using your global IPv4 address as *local-address*:

```
interface 6to4 add mtu=1280 local-address=1.2.3.4 disabled=no
```

Now you need to add a IPv6 address to the tunnel interface. The address should be in form *"2002 + <IPv4 address in hex> + <custom id>"* . A bash script can be used to generate such IPv6 address for you:

```
 atis@atis-desktop:~$ ipv4="1.2.3.4"; id="1"; printf "2002:%02x%02x:%02x%02x::$id\n"
 `echo $ipv4 | tr "." " "`
2002:0102:0304::1
```

Add the generated address to the 6to4 interface:

```
ipv6 address add address=2002:0102:0304::1/128 interface=sit1
```

Add route to global IPv6 Internet through the tunnel interface using the anycast IPv4 address:

```
ipv6 route add dst-address=2000::/3 gateway=::192.88.99.1,sit1
```

Syntax for RouterOS v4.x, or RouterOS 3.x with **routing-test**:

```
ipv6 route add dst-address=2000::/3 gateway=::192.88.99.1%sit1
```

Now try to ping some IPv6 host (e.g. ipv6.google.com, 2001:4860:a003::68) to check your IPv6 connectivity.

See also 6in4 (http://en.wikipedia.org/wiki/6in4) and 6to4 (http://en.wikipedia.org/wiki/6to4) in Wikipedia.

Using dual stack

All IP services that listen to IPv6 also accept IPv4 connections. We take the web proxy for an example.

To force the web proxy to listen to IPv6 connections:

```
/ip proxy set src-address=::
```

To demonstrate that the dual stack is working, we connect to the web proxy at *10.0.0.131/fc00:1::1* using telnet, issue "GET /" request, and observe generated error message.

Connecting via IPv4:

```
$ telnet 10.0.0.131 8080
Trying 10.0.0.131...
Connected to 10.0.0.131.
Escape character is '^]'.
GET /

HTTP/1.0 404 Not Found
Content-Length: 518
...
Generated Mon, 18 Dec 2006 12:40:03 GMT by 10.0.0.131 (Mikrotik HttpProxy)
```

Connecting via IPv6:

```
$ telnet -6 fc00:1::1 8080
Trying fc00:1::1...
Connected to fc00:1::1.
GET /

HTTP/1.0 404 Not Found
```

```
Content-Length: 525
...
Generated Mon, 18 Dec 2006 12:38:51 GMT by ::ffff:10.0.0.131 (Mikrotik HttpProxy)
```

[Top | Back to Content **]**