

Lab 1.4. IPv6

Objectives

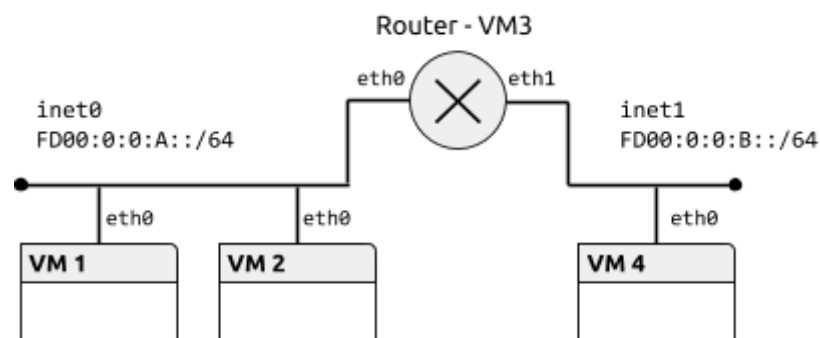
In this lab, we will study basic aspects of IPv6 protocol, how to manage different address types and configuration mechanisms. Also, we will analyze the most important characteristics of ICMPv6.

Contents

- Environment Preparation
- Link Local Addresses
- Unique Local Addresses
- Static Routing
- Persistent Configuration
- Autoconfiguration. Prefix Announcement
- ICMPv6

Environment Preparation

We will configure the network topology shown in the following figure:



The topology configuration file would have the following content:

```
netprefix inet
machine 1 0 0
machine 2 0 0
machine 3 0 0 1 1
machine 4 0 1
```

Link Local Addresses

A link-local address is valid only for communications within the network link that the interface is connected to. Packets with link-local addresses are never routed. The format prefix for these addresses is fe80::/10.

Exercise 1 [VM1,VM2]. Activate interfaces eth0 in VM1 and VM2 and check the link-local addresses assigned with the ip command.

Exercise 2 [VM1,VM2]. Check the connectivity between VM1 and VM2 with the ping6 command. When link-local addresses are used, it is necessary to specify the source interface, with the option -I or adding %<interface_name> to the address. Consult other options in the man page. Observe the generated traffic with the wireshark tool, specially the protocols encapsulated on each

datagram and the IPv6 parameters.

To know more... IPv4 also reserves an address block (169.254.1.0 - 169.254.254.255) for link-local addressing, when it is not possible to configure interfaces by other means. More details in RFC 3927.

Unique Local Addresses

A Unique Local Address (ULA) is globally unique and is intended for local communications. Packets with ULAs are not globally routable. Instead, they are routed inside of a more limited area, such as a site or between a limited set of sites. The format prefix is `fc00::/7`.

Exercise 1 [VM1, VM2]. Configure VM1 and VM2 to have an ULA address in network `fd00:0:0:a::/64` with command `ip`. The interface ID can be chosen freely, as long as it is not the same in both machines. **Note:** Do not forget to include the prefix length when setting the addresses.

Exercise 2 [VM1, VM2]. Check the connectivity between VM1 and VM2 with the `ping6` command using the new address. Observe the generated traffic with `wireshark`.

Exercise 3 [Router, VM4]. Activate the `eth0` interface in VM1 and the two interfaces in Router. Check the connectivity between Router and VM1, and between Router and VM4 using link-local addresses. Remember to specify the source interface.

Exercise 4 [Router, VM4]. Configure ULA addresses for both interfaces of Router (networks `fd00:0:0:a::/64` and `fd00:0:0:b::/64`) and the interface of VM4 (`fd00:0:0:b::/64`). As in the previous exercise, the interface ID must be different within the same network.

Exercise 5. Check the connectivity between Router and VM1, and between Router and VM4 using ULA addresses. Check also that VM1 can not reach VM4.

Static Routing

According to the topology, Router VM must route traffic between networks `fd00:0:0:a::/64` and `fd00:0:0:b::/64`. In this section, we are going to configure static routing, based on routes we will manually set in all VMs.

Exercise 1 [VM1, Router]. Consult the route tables in Router VM, using the `ip route` command. Consult the man page of the command to select IPv6 routes.

Exercise 2 [Router]. In order to Router VM to effectively act as a router, we have to enable IPv6 traffic forwarding. This can be temporarily done with command `sysctl -w net.ipv6.conf.all.forwarding=1`.

Exercise 3 [VM1, VM2, VM4]. Finally, we have to configure the route table in the VM. Add the address of Router as the default route with `ip route`. Check the connectivity between VM1 and VM4 using the `ping6` command.

Exercise 4. Delete the neighbor cache in VM1 and Router (with `ip neigh flush dev eth0`). Using `wireshark`, complete the following table with the `ping6` command between VM1 and VM4 with all messages until the first ICMP Echo Reply:

Network fd00:0:0:a::/64 - VM1

Src MAC addr	Dst MAC addr	Src IPv6 addr	Dst IPv6 addr	ICMPv6 type

Network fd00:0:0:b::/64 - VM4

Src MAC addr	Dst MAC addr	Src IPv6 addr	Dst IPv6 addr	ICMPv6 type

Persistent Configuration

The configuration done in previous sections are volatile and will disappear when the server are restarted. During system boot, it is possible to automatically configure some interfaces according to the information stored on the server's disk.

Exercise 1 [Router]. Create two files, `ifcfg-eth0` and `ifcfg-eth1`, in directory `/etc/sysconfig/network-scripts/` with the configuration of each interface. Use the following options:

For IPv6	For IPv4
Type=Ethernet BOOTPROTO=static IPv6ADDR=<IP address in CIDR format> IPv6_DEFAULTGW=<IP address of the default router (if it exists)> DEVICE=<Interface name (eth0...)>	Type=Ethernet BOOTPROTO=static IPADDR=<IP address in CIDR format> GATEWAY=<IP address of the default router (if it exists)> DEVICE=<Interface name (eth0...)>

Nota: These options are described in detail in `/usr/share/doc/initscripts-*/sysconfig.txt`.

Exercise 2 [Router]. Check the automatic configuration with commands `ifup` and `ifdown`.

Autoconfiguration. Prefix Announcement

The neighbour discovery protocol is also used for the autoconfiguration of network interfaces. When an interface is enabled, a message for router discovery is sent. Then, the available routers answer with an advertisement containing, among others, the network prefix.

Exercise 1 [VM1, VM2, VM4]. Remove ULAs from interfaces (`ip addr del`) and disable them (with `ip link set eth0 down`).

Exercise 2 [Router]. Configure zebra to make Router announce network prefixes. To do that, create file `/etc/quagga/zebra.conf` and include prefix information for both networks as in the following listing:

```
interface eth0
  no ipv6 nd suppress-ra
  ipv6 nd prefix fd00:0:0:a::/64
```

Finally, start the service with command `service zebra start`.

NOTE: In `/usr/share/doc/quagga-0.99.22.4` there are example configuration files.

Exercise 3 [VM4]. Check the autoconfiguration of the network interface of VM4, by enabling it, and consult the assigned address.

Exercise 4 [VM1 and VM2]. Analyze messages corresponding to the neighbour discovery protocol:

- Enable the interface in VM2, check that it is correctly configured and start a traffic capture with wireshark.
- Enable the interface in VM1 and analyze ICMP messages of type “Router solicitation” and “Router advertisement”.
- Check source and destination addresses of datagrams, as well as source and destination addresses of the Ethernet frame. Especially, the relation between the multicast group ID with IP and MAC addresses. Analyze the output of command `ip maddr`.

To know more... In the autoconfiguration process, the interface ID is also generated following the “*Extended Unique Identifier*” (EUI-64) described in RFC 4193. The configuration of router advertisement protocol has many options that can be consulted in the zebra documentation (e.g. interval between unsolicited advertisements). If only prefix announcement is needed, without any routing functionality, the open source project “*Router Advertisement Daemon*”, `radvd`, can also be used.

Exercise 5 [VM1]. The interface ID generation through EUI-64 poses a privacy problem for client machines, as they can be tracked by their MAC addresses. In this case, it is convenient to activate the privacy extensions, which consist on generating a pseudorandom temporary interface ID for global addresses. Activate the privacy extensions in VM1 with command `sysctl -w net.ipv6.conf.eth0.use_tempaddr=2`.

ICMPv6

ICMPv6 provides messages for network control, both to detect errors and to get information from the network. In this lab we have already seen the most important ones.

Exercise 1. Generate messages of the following types in the network and analyze them with wireshark:

- Echo request and reply.
- Router solicitation and advertisement
- Neighbour solicitation and advertisement
- Destination unreachable - No route to destination (Code: 0)