IPv6 Workshop : Location

Date

Host Configuration (Linux)

Trainer Name

Laboratory Exercise: *Host Configuration (Linux)*

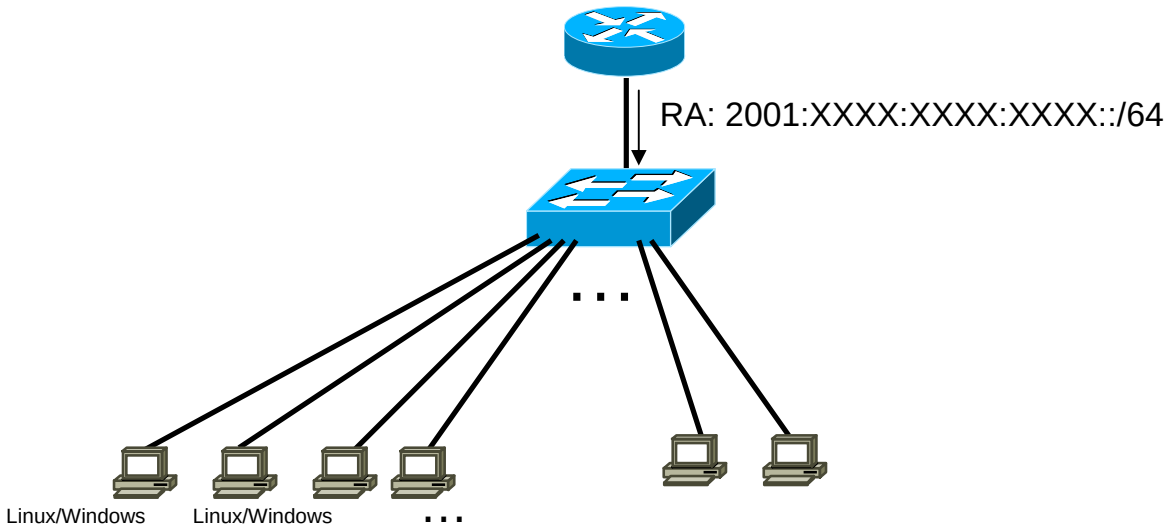
Objectives

In this laboratory exercise you will complete the following tasks:

- *Check for IPv6 support in the running kernel*
- *Understand basic IPv6 concepts (NDP)*
- *Manually add/remove IPv6 addresses on Linux*
- *Use some basic IPv6 related tools*

Visual Objective

The following figure shows the topology of the current laboratory.



Task 1: *Verify IPv6 support in your Linux*

Complete the following exercise's steps:

Step 1: Check for IPv6 support in the running kernel.
Modern Linux distributions already contain IPv6−ready kernels, the IPv6 capability is generally compiled as a module. To check whether your current running kernel supports, or not, IPv6 the following file must exist: `/proc/net/if_inet6`

It's possible that the IPv6 module is not loaded automatically on startup. So, verify that the module is running by listing the current loaded modules:

```
lsmod |grep ipv6
```

Task 2: *Display and identify existing IPv6 addresses*

Complete the following exercise's steps:

Step 1: Check if IPv6 addresses are already configured

Run the following commands:

- `ip` command
  `ip -f inet6 addr show dev <interface>`

- `ifconfig` command
  `ifconfig <interface>`

Step 2: Using the previous commands, identify the different types of IPv6 addresses

- Link local (**Tip:** Search for fe80::...)
- auto-configuration IPv6 address (**Tip:** Search for ...`ff:fe`...)
- multicast address (use netstat with –g option)
- validity of addresses

Task 3: Using some IPv6 related tools

Complete the following exercise's steps:

Step 1: Ping IPv6 addresses

- Ping the IPv6 localhost address (**::1**)
- Ping your host's link-local and global addresses (**Tip:** Use the option `-I` in `ping6` command)
- Ping your host's multicast addresses (**Tip:** Use the option `-I` in `ping6` command)

Step 2: Without looking into the router, identify the router's link-local address for your lan (**Tip:** Use the command **`ip -6 neigh …`**)

- Ping router's addresses (link-local and global addresses). Did you successfully ping router's link-local address? (**Tip:** Use the option **`-I`** in **`ping6`** command)

Task 4: Add/Remove IPv6 addresses

Step 1: Manually add an IPv6 address

On your network interface, add the following address:
`2001:XXXX:XXXX:XXXX`::NN (where NN is the number of your PC)

You can accomplish this task using different methods:

- Using **`ip`** command (temporary address).

```
ip -6 addr add <ipv6address>/<prefixlength> dev <interface>
```

- Using **`ifconfig`** command (temporary address).

```
ifconfig <interface> inet6 add <ipv6address>/<prefixlength>
```

- Editing the file /**`etc/network/interfaces`** and and the following lines:

```
                iface eth0 inet6 static
turn off autoconf:   up sysctl -q -w net.ipv6.conf.eth0.autoconf=0
                     address 2001:XXXX:XXXX:XXXX::2
                     netmask 64
```

Then you'll need to restart your network /**`etc/init.d/networking restart`**)

Step 2: Manually remove an IPv6 address

Remove the address created on the previous step.

You can accomplish this task using different methods:

- Using **`ip`** command.

```
ip -6 addr del <ipv6address>/<prefixlength> dev <interface>
```

- Using **`ifconfig`** command.

```
ifconfig <interface> inet6 del <ipv6address>/<prefixlength>
```

- Use this method only if you used a similar method in the previous step. Editing the file /**`etc/network/interfaces`** and removing the **`inet6`** entries (then you'll need to restart your network /**`etc/init.d/networking restart`**)

Task 5: Analyse IPv6 messages (NS/NA, RS/RA)

Step 1: Using *tcpdump* (`tcpdump -t -n -vv -s 0 ip6 -i <Interface>`) or `wireshark (Ethereal)`, capture neighbour advertisement and neighbour solicitation messages. To do that, retrieve the IPv6 address of your neighbour and ping this address.

- Which IPv6 addresses (source - destination) are used in this messages?

Step 2: Now, The router is sending router advertisements.
Using *tcpdump* (`tcpdump -t -n -vv -s 0 ip6 -i <Interface>`) or `wireshark (Ethereal)`, capture router advertisement and router solicitation messages. If you want, look at Appendix *A* for *tcpdump* basic information or use your linux manual pages.

- Which IPv6 addresses (source - destination) are used in this messages?

Step 3: Display your current IPv6 routing table (**Tip:** Use command `route -A inet6 or ip -6 route`). Identify the next hop for your default gateway.

Summary

After completing these exercises, you should be able to:

- *Verify if a kernel supports IPv6*
- *Check if the IPv6 module is loaded*
- *Identify different types of addresses*
- *Manually add/remove IPv6 addresses*
- *Visualize the IPv6 routing table*

Appendix A

Using "IPv6 tcpdump"

On Linux, *tcpdump* is the major tool for packet capturing. Below you can find some examples. IPv6 support in tcpdump is available since version 3.6. tcpdump uses expressions for filtering packets to minimize the "noise":

- **icmp6:** filters native ICMPv6 traffic
- **ip6:** filters native IPv6 traffic (including ICMPv6)
- **proto ipv6:** filters tunneled IPv6−in−IPv4 traffic
- **not port ssh:** to suppress displaying SSH packets for running *tcpdump* in a remote SSH session

Some more command line options are very useful to catch and print more information in a packet, mostly interesting for digging into ICMPv6 packets:

- **-s 512**: increase the snap length during capturing of a packet to 512 bytes
- **-n**: don't convert host addresses to names.
- **-i**: Listen on <interface>
- **-vv:** Even more verbose output

Example: IPv6 ping to 2001:XXXX:XXXX:XXXX::3 native over a local link

```
tcpdump -t -n -vv -s 512 ip6 -i eth0
```

```
tcpdump: listening on eth0
2001:XXXX:XXXX:XXXX:2e0:18ff:fe90:9205 > 2001:XXXX:XXXX:XXXX::3: icmp6: echo
¬ request (len 64, hlim 64)
2001:XXXX:XXXX:XXXX::3 > 2001:XXXX:XXXX:XXXX:2e0:18ff:fe90:9205: icmp6: echo
¬ reply (len 64, hlim 64)
```

Appendix B

Kernel settings in /proc−filesystem

The virtual filesystem that we call */proc* contains lots of different data structures and information gathered from the kernel at runtime, and updated whenever you try to list or view the information. However, most of the files available through the */proc* filesystem are only available in read only mode, which means they can't be changed. This is because they only supply us with informational data.
On the other hand, all of the variables located in */proc/sys* (and the correspondent subdirectories) are writable as well as readable.

How to set variables

The *ipsysctl* variables may be set in two different ways which entails two totally different methods. The first one uses the /proc filesystem, which should come with any linux installation as long as you have a kernel that has /proc filesystem turned on. The other way is via the **`sysctl`** application provided with most distributions per default these days.

Using `cat` and `echo` (/proc filesystem)
Using **`cat`** and **`echo`** is the simplest way to access the */proc* filesystem

You need to have read and sometimes also write access (normally root only) to the /proc−filesystem

- Retrieving a value
  The value of an entry can be retrieved using *cat*:
  ```
  cat /proc/sys/net/ipv6/conf/all/forwarding
  0
  ```

- Setting a value
  A new value can be set (if entry is writable) using "echo":
  ```
  echo "1">/proc/sys/net/ipv6/conf/all/forwarding
  ```

Using `sysctl`
Using the **`sysctl`** program to access the kernel switches is a common method today.

- Retrieving a value
  The value of an entry can be retrieved through:
  ```
  sysctl net.ipv6.conf.all.forwarding
  net.ipv6.conf.all.forwarding = 0
  ```

- Setting a value
  A new value can be set (if entry is writable):
  ```
  # sysctl -w net.ipv6.conf.all.forwarding=1
  ```

```
net.ipv6.conf.all.forwarding = 1
```

Note: Don't use spaces around the "=" on setting values. Also on multiple values per line, quote them like e.g.

```
# sysctl -w net.ipv4.ip_local_port_range="32768 61000"
net.ipv4.ip_local_port_range = 32768 61000
```

Values

There are several formats seen in /proc−filesystem:

- **BOOLEAN**: simple a "0" (false) or a "1" (true)

- **INTEGER:** an integer value can be unsigned, too more sophisticated lines with several values: sometimes a header line is displayed also, if not, have a look into the kernel source to retrieve information about the meaning of each value...

In */proc/sys/net/ipv6/…* you can find plenty of IPv6 kernel parameters that can be configured at runtime.

Next you can find a small list of IPv6 related variables (Consult Documentation/ip-sysctl.txt to see all the existent variables)

- **use_tempaddr** - INTEGER
  Preference for Privacy Extensions (RFC3041).
  <= 0 : disable Privacy Extensions
  == 1 : enable Privacy Extensions, but prefer public addresses over temporary addresses.
  > 1 : enable Privacy Extensions and prefer temporary addresses over public addresses.
  Default: 0 (for most devices)
  -1 (for point-to-point devices and loopback devices)

- dad_transmits - INTEGER
  The amount of Duplicate Address Detection probes to send.
  Default: 1

- mtu - INTEGER
  Default Maximum Transfer Unit
  Default: 1280 (IPv6 required minimum)

- router_solicitation_delay - INTEGER
  Number of seconds to wait after interface is brought up before sending Router Solicitations.
  Default: 1

- router_solicitation_interval - INTEGER
  Number of seconds to wait between Router Solicitations.
  Default: 4

- router_solicitations - INTEGER
  Number of Router Solicitations to send until assuming no routers are present.
  Default: 3