

Project no. 015926

6DISS

IPv6 Dissemination and Exploitation

Instrument: SPECIFIC SUPPORT ACTION

Thematic Priority 2

Report on the Hands-on material

Actual submission date: September 2007

Start date of project: April 1st 2005

Duration: 30 months

Organization name of lead contractor for this document:

RENATER

Revision: V0.1

Executive Summary

This document is a report gathering hands-on material used for the 6DISS IPv6 technical workshops. The report includes information about different topics: *Host Configuration*, *DHCPv6*, *Routing*, *DNS*, *Services/Applications*, *Management* and *Security*.

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	



Table of Contents

1	INTRODUCTION	3
2	HOST CONFIGURATION HANDS-ON	5
2.1	LINUX	5
2.2	WINDOWS	14
3	STATEFUL AUTOCONFIGURATION HANDS-ON.....	20
4	ROUTING HANDS-ON.....	26
4.1	ROUTING HANDS-ON: PARIS.....	26
4.2	ROUTING HANDS-ON: LOCAL LAB.....	32
4.3	ROUTING HANDS-ON: HELP COMMANDS	42
5	DNS HANDS-ON.....	49
6	APPLICATION/SERVICES HANDS-ON	55
7	MANAGEMENT HANDS-ON.....	61
8	SECURITY HANDS-ON	68

1 Introduction

This document is a report gathering the hands-on material used for the 6DISS IPv6 technical workshops. The report includes the following topics:

- **Host Configuration:** Two hands-on are proposed to illustrate how IPv6 works on **Linux** and **Windows** operating systems. IPv6 protocol is analysed in depth: Neighbour Discovery, Autoconfiguration, etc
- **Stateful autoconfiguration:** In this hands-on, the trainees install a DHCPv6 client (Dibbler) and make a basic configuration.
- **Routing:** In this hands-on, IPv6 routing protocols are deployed by the trainees in a testbed. IGP (OSPF) and EGP (BGP) protocols are tested. Two hands-ons are depicted: One for the Paris lab and another for the local lab.
- **DNS:** In this hands-on, the trainees has to manipulate IPv6 resource records (AAAA, PTR) in a DNS server. This hands-on is performed on Linux OS.
- **Services/Applications:** In this hands-on, the trainees install IPv6 services like web servers (with virtual hosts) or FTP servers. This hands-on is performed on Linux OS. The other trainees will check that services are available by using IPv6 web/ftp clients (on Linux or Windows).
- **Management:** After having tested the services, the trainees will install a management application (Argus) to supervise the routers, PCs and these IPv6 services (web, ftp, etc).
- **Security:** Finally, the trainees will add filters on the routers (ACL) and on the PCs (iptables) to allow/deny some PCs/services.



Host Configuration hands-on



2 Host Configuration Hands-on

Depending on the workshops and the requirements expressed by the local organisers, Linux and/or Windows *Host Configuration* hands-on were included in the workshop programme.

2.1 Linux

Laboratory Exercise: *Host Configuration (Linux)*

Objectives

In this laboratory exercise you will complete the following tasks:

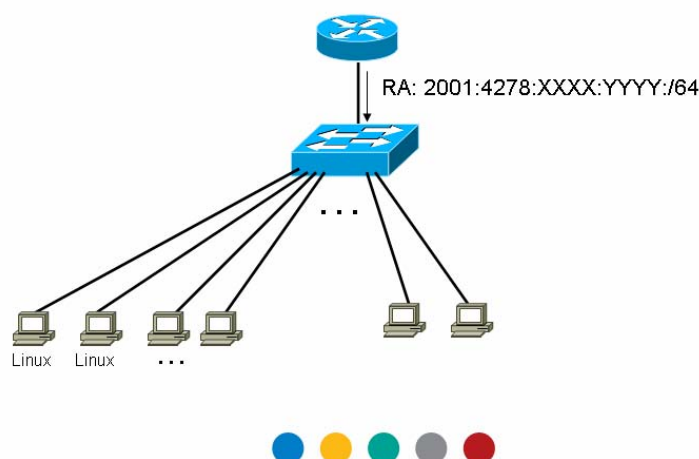
- *Check for IPv6 support in the running kernel*
- *Understand basic IPv6 concepts (NDP)*
- *Manually add/remove IPv6 addresses on Linux*
- *Use some basic IPv6 related tools*

Visual Objective

The following figure shows the topology of the current laboratory. This laboratory is similar with the one on the *Windows XP* configuration module.



Architecture



3

Figure 1: Scenario topology



Scenario

The router is sending periodically the correspondent router advertisement messages.

Now that we have IPv6 support on each link, we are going to verify if IPv6 is enabled on our Operating System. We are also going to make some basic IPv6 related configurations.

Note that each PC has a number (01...NN)

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 and which IPv6 addresses are already configured

Run the following commands:

- `ip` command
`ip -6 addr show dev <interface>`
- `ifconfig` command
`ifconfig <interface>`
- `netstat` command
`netstat -inet6 -g`



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
- 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
- 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)

Step 3: Using `tcpdump` (`tcpdump -t -n -vv -s 512 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 4: Using `tcpdump` (`tcpdump -t -n -vv -s 512 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 5: Display your current IPv6 routing table (**Tip:** Use command `route -A inet6`). Identify the next hop for your default gateway.



Task 4: Add/Remove IPv6 addresses

Step 1: Manually add an IPv6 address

On your network interface, add the following address: 2001:4278:XXXX:YYYY::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 the following lines:

```
iface eth0 inet6 static
# turn off autoconf up systemctl -q -w net.ipv6.conf.eth0.autoconf=0
address 2001:660:Y:Z::1
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`)



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” (Linux)

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:DB8:CAFE:22::1 native over a local link

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

```
tcpdump: listening on eth0
2001:db8:cafe:22:2e0:18ff:fe90:9205 > 2001:db8:cafe:22::1: icmp6: echo
↪ request (len 64, hlim 64)
2001:db8:cafe:22::1 > 2001:db8:cafe:22:2e0:18ff:fe90:9205: icmp6: echo
↪ reply (len 64, hlim 64)
```

Appendix B: Kernel settings in */proc*-filesystem (Linux)

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



2.2 Windows

Laboratory Exercise: *Host Configuration (Windows XP)*

Objectives

In this laboratory exercise you will complete the following tasks:

- *Activate the IPv6 protocol stack on WinXP PC's*
- *Understand basic IPv6 concepts*
- *Manually add/remove IPv6 addresses on Win XP*

Visual Objective

The following figure shows the topology of the current laboratory.

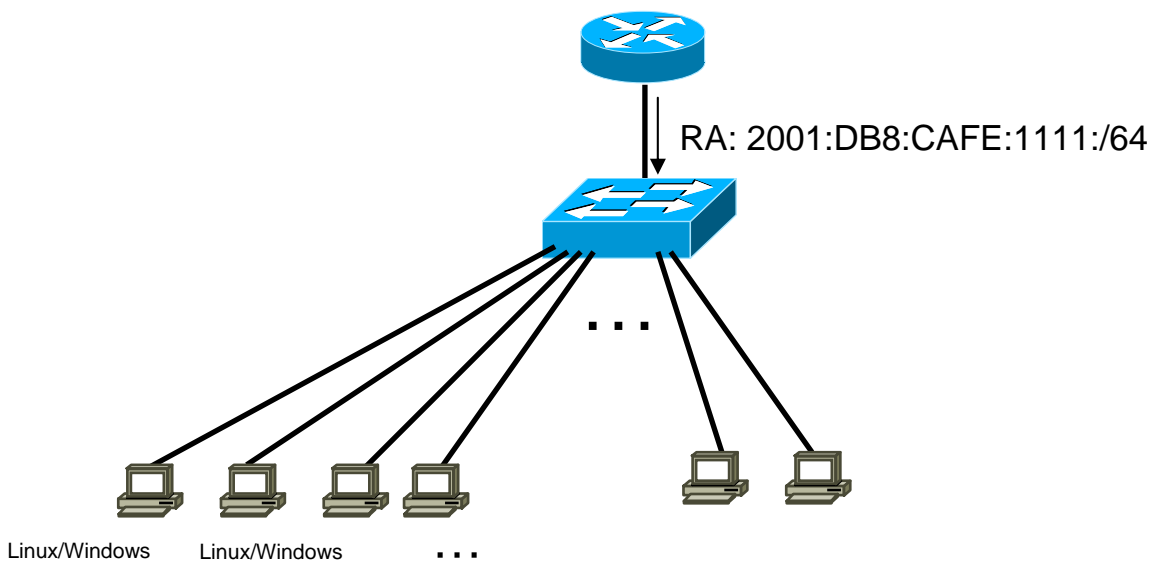


Figure 2: Scenario topology



Task 1: *Enabling IPv6 on Windows XP*

Complete the following exercise's steps:

Step 1: Enable IPv6 on your Windows XP (SP2)

(**Tip:** There are two alternative methods to do it)

- Using the WinXP GUI

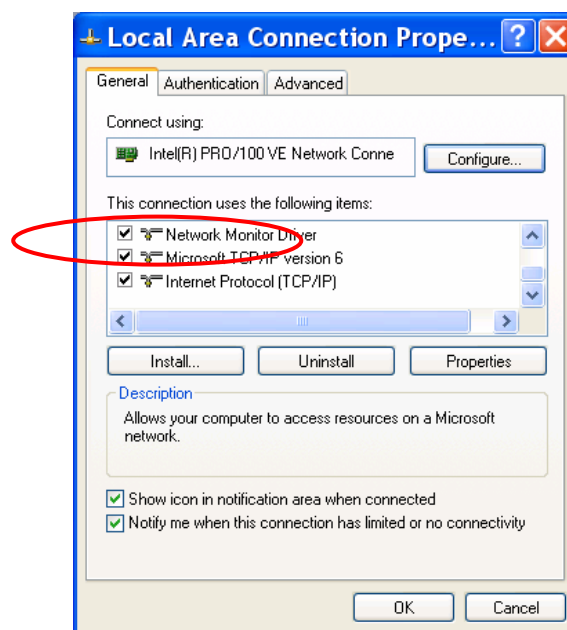


Figure 3: IPv6 GUI installation

- Or from a CLI run `ipv6 install`



Task 2: Display and identify existing IPv6 addresses

Complete the following exercise's steps

Step 1: Identify the different interfaces at your PC. Which ones are related to IPv6 transition mechanisms?

From a CLI run the following commands:

- `ipconfig /all`
- `netsh interface ipv6 show interface` (look at the end of the document for `netsh` basic information).
- `ipv6 -v if`

Step 2: Identify different types of IPv6 addresses (Notice: You already have IPv6 addresses and you didn't have to make any configuration on your host machine)

- Link local (**Tip:** Search for `fe80::...`)
- auto-configuration IPv6 address (**Tip:** Search for `...ff:fe...`)
- IPv6 address due to privacy extension
- multicast addresses
- validity of addresses (**Tip:** Use the command `netsh interface ipv6 show address <interface>`)

Task 3: Using some IPv6 related tools

Step 1: Ping local IPv6 addresses

- Ping the IPv6 localhost address (`::1`)
- Ping your host's link-local and global addresses

Step 2: Without looking into the router, identify the router's link-local address (for your vlan)

- What's the appropriate command?
 - `Traceroute?`
 - `Show neighbors?`
 - `ipconfig?`
- Ping router's addresses (link-local and global addresses). Did you successfully ping router's link-local address?

Step 3: Display your current IPv6 routing table (**Tip:** Use command `netsh interface ipv6 show routes level=verbose`).



Task 4: Add/Remove IPv6 addresses

Complete the following exercise's steps

Step 1: Manually add an IPv6 address

- On your local area connection, add the following address:
2001:DB8:CAFE:XY::10, remember, XY=your vlan number

(Tip 1: `netsh ipv6 interface add ...`)

(Tip 2: `<interface -> interface number or name`)

Step 2: Manually remove an IPv6 address

- Remove the address created on the previous step (Tip: `netsh ipv6 interface delete ...`)

Step 3: Disable privacy extensions (RFC3041). (Tip: Use the command `netsh interface ipv6 set privacy`)

- What could be the problems in terms of security if you enable/disable privacy extension?

Step 4: Disable 6to4 and isatap virtual interfaces

(Tip 1: Use the command `netsh interface ipv6 6to4 set state ...`)

(Tip 2: Use the command `netsh interface isatap set ...`)

Summary

After completing these exercises, you should be able to:

- *Enable and configure IPv6 addresses on windows XP*
- *Identify different address types*
- *Manually add/remove IPv6 addresses*
- *Disable 6to4 and isatap virtual interfaces*



Appendix A: Using Netsh (Windows)

Netsh is a command-line scripting utility, for the Windows Operating System, which allows you to display or modify a computer's network configuration currently running.

Netsh contexts

To run a **netsh** command, you must start **netsh** from the CLI prompt and change to the context that contains the command you want to use. The contexts that are available to you depend on which networking components you have installed. For example, if you type **dhcp** at the **Netsh** command prompt, you change to the DHCP context, but if you do not have DHCP installed the following message appears:

The following command was not found: dhcp.

Running Netsh commands from the Netsh.exe command prompt

Netsh uses the following standard commands in all contexts that you can run from a Netsh.exe command prompt (that is, netsh>). There might be functional differences between Netsh context commands on Windows 2003 and Windows XP.

1. To view the command syntax, click a command.
2. **..** Moves to the context that is one level up.
3. **{/?|?|help|h}** Displays help at the command prompt.
4. **Abort** Discards any changes made in offline mode. **Abort** has no effect in online mode.
5. **quit** Exits Netsh.exe

Example

The following sample script changes a context from the root context to the **interface ipv6** context and adds an IPv6 address:

```
C:\> netsh
netsh>
netsh> interface ipv6
netsh interface ipv6>
netsh interface ipv6> add address interface=7 address=2001:ABBA::1
```



Stateful Autoconfiguration Hands-on



3 Stateful autoconfiguration hands-on

Laboratory Exercise: *Stateful Auto-configuration*

Objectives

In this laboratory exercise you will complete the following tasks:

- *Minor DHCPv6 client configuration*
- *Experience the usage of both auto-configuration modes (stateless and stateful)*

Visual Objective

The following figure shows the topology of the current laboratory.

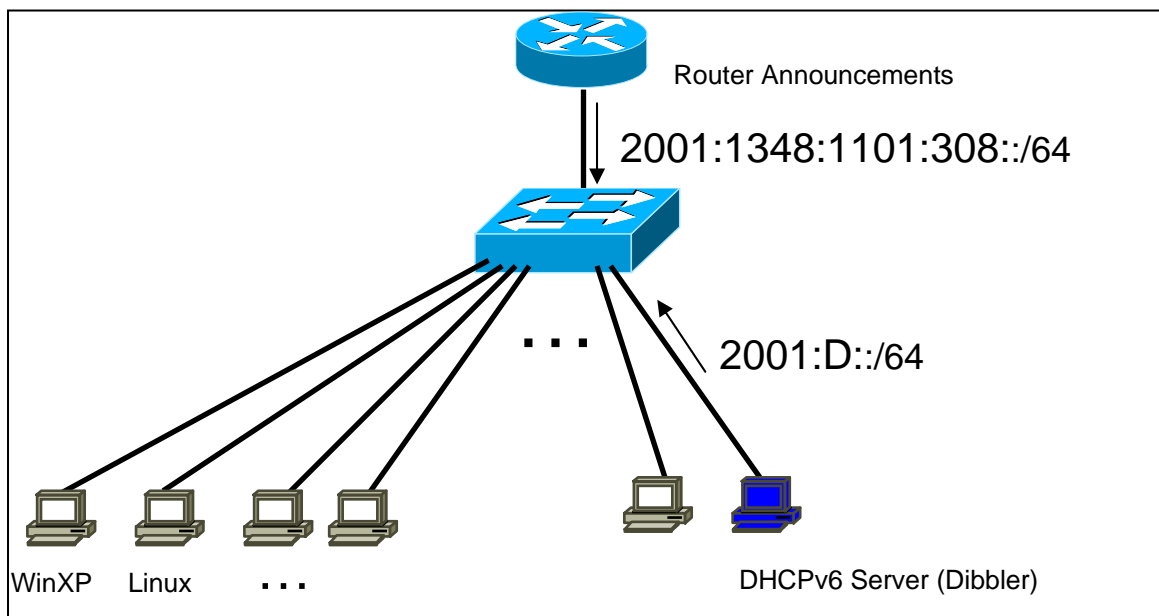


Figure 4: Scenario topology

Setup

All hosts connected to a specific router will be on the same VLAN.

Please, verify that your IPv6 address belongs to the following network:

- *2001:1348:1101:308::/64*

If you have manually configured an IPv6 address that doesn't belong to one of these networks please remove it.

On this scenario there is a DHCPv6 server (Linux) with one network interface.

Scenario

On the previous scenario, you changed the values of two auto-configuration related parameters, on the router advertisement messages. But at the present those flags are ignored by Operating Systems. So, we have to run an external DHCPv6 client.

For both client and server we are going to use the Dnsmasq implementation. Currently there are ports available for Windows XP, Windows 2003 and Linux systems.

On the present setup the server will run on a Linux and the clients will run on Windows XP and/or Linux.

*The DHCPv6 client is configured via the `/etc/dnsmasq/client.conf` (**C:\dnsmasq\client.conf**) file. You already have a configuration file there. We are only going to do some basic configurations. Therefore, you can use this file as a starting point for your client configuration.*

Please remove the unnecessary comments (character "#") in order to have a configuration file similar to this one:

```
log-mode short
# 7 = omit debug messages
log-level 7
# Windows :
iface "Local Area Connection" {
```

```
# Linux:
iface eth0 {
    option dns-server
    option domain
    ia
}
```

If you want to know how to fully configure the client, please consult the *Dibbler User's Guide* (</usr/local/dibbler/doc/dibbler-user.pdf>, or <C:\dibbler\dibbler-user.pdf>).

For the purpose of this scenario what's important for you to understand is that with this configuration, for interface *eth0*, the client will request the following parameters from the server:

- List of *dns-servers* (keyword **option dns-server**)
- List of *domains* (keyword **option domain**)
- One IPv6 address (keyword **ia**)

The DHCPv6 server, has one interface, over which it hands out addresses from a pool.

<i>Network</i>	<i>Interface</i>	<i>Pool</i>
<i>2001:D::/64</i>	<i>Eth0</i>	<i>2001:D::1 - 2001:D::FFFF</i>

Table 1: Address pool

The *dns-server* address and the domain name:

<i>Network</i>	<i>Interface</i>	<i>DNS server</i>	<i>Domain</i>
<i>2001:D::/64</i>	<i>Eth0</i>	<i>2001:D::53</i>	<i>6diss.org</i>

Table 2: DNS Server and Domain name

Task 1: Using Router Advertisement messages and DHCPv6

Complete the following exercise's steps:

Step 1: Configure a DHCPv6 client

Enter your Linux operating system and configure the DHCPv6 client according to the instructions on the previous Setup/Scenario section.

Check if your computer has more than one network interface. If so, make sure you configure the correct interface.

Step 2: Check your current global IPv6 address(es).

Step 3: With Ethereal, start capturing IPv6 packets and then run the DHCPv6 client.

You can run Dibbler client as a Daemon – detached from console and run in the background - but during the test phase it is better to watch its behaviour in real time. In this case use the `run` parameter. Dibbler will show its log messages in the console.

(Tip: `/usr/local/dibbler/dibbler-client run`
or `cd c:\dibbler and dibbler-client.exe run`)

Step 2: On the DHCPv6 client's running console, check the messages and verify if you received the requested data (IPv6 address, DNS servers and Domains).

Step 4: With Ethereal, analyze the request messages sent to the DHCPv6 server. What are the source and destination addresses? Then, examine the response messages from the server. Can you identify on these messages the response data from the server?

Step 5: Using Linux commands check if you received the data you requested.

(Tip: `ifconfig` and `cat /etc/resolver.conf`)

Check also your current IPv6 routing table (`route -A inet6 -n`). Do you have a default gateway?

Step 6: Kill your DHCPv6 client (Ctrl + C on the running console)

Step 7: Check that you don't have any address assigned by the server neither the information regarding the DNS and domain lists.



Task 2: *No Router Advertisement messages and DHCPv6*

On this task the Router Advertisement messages are disabled. This means there are no RA messages on the network.

Step 1: Restart your network (**Tip:** `/etc/init.d/network restart`)

Step 2: Check if you have any IPv6 address.

Step 3: Run Dibbler client.

Step 4: Check if you have IPv6 addresses and the DNS server list.

Step 5: Now check your current IPv6 routing table (**Tip:** `route -A inet6 -n`). Do you have a default gateway? Why?

Summary

After completing these exercises, you should be able to:

- *Do basic DHCPv6 client configuration*
- *Understand the usage of both auto-configuration modes (stateless and stateful)*



Routing Hands-on



4 Routing hands-on

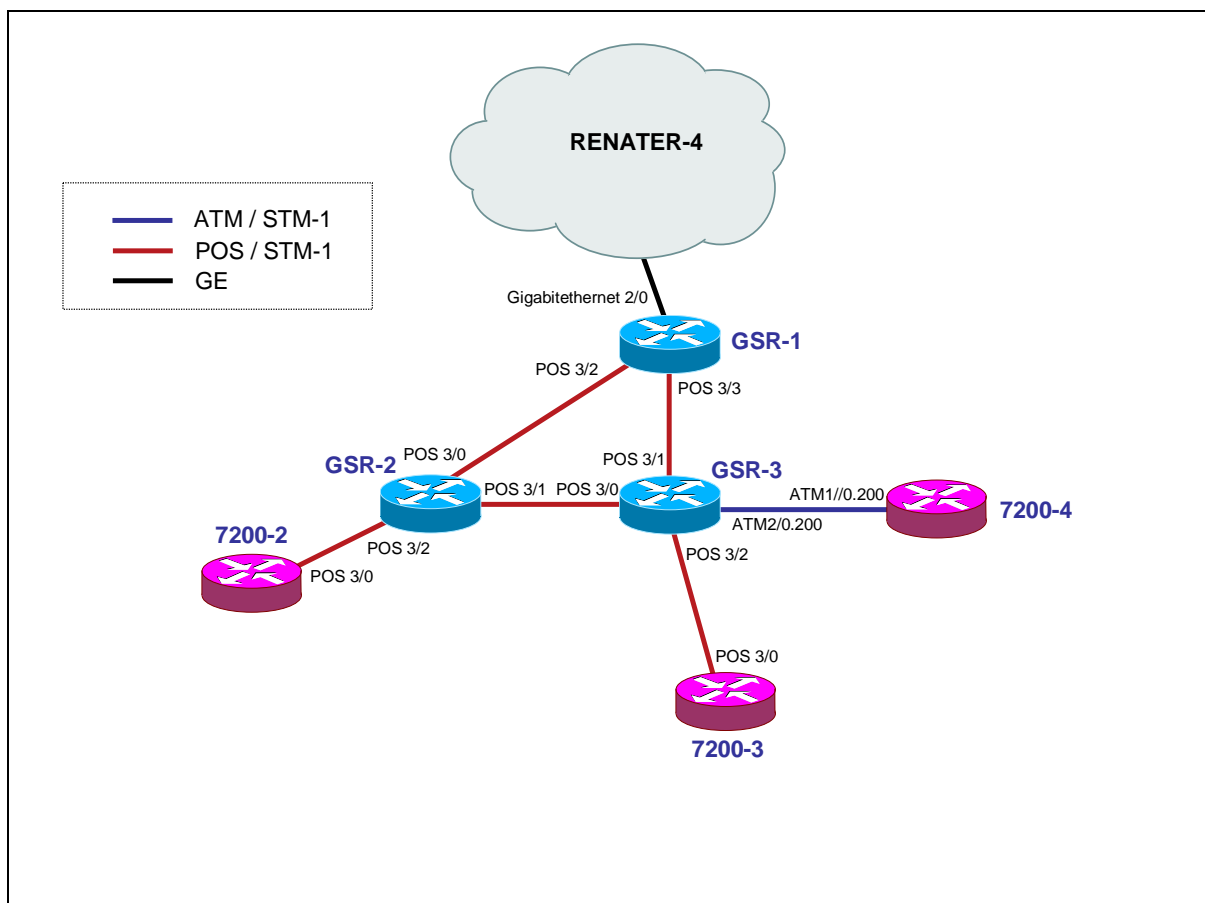
Initially, there were two remote laboratories (Paris and Brussels) available for the routing hands-on. Occasionally, when the internet connectivity was poor in the place hosting the workshop, we installed a local lab to avoid problems accessing the remote labs. Six small C1800 series routers could be shipped if the local organisers couldn't provide this material. Due to the disappearance of the Brussels remote lab few months ago, we had to install in the last workshops a local lab.

The labs can be booked by sending an email to labs@6diss.org.

4.1 Routing hands-on: Paris

Laboratory Exercise: *Routing Configuration / Paris laboratory*

Testbed setup



Testbed diagram



Routers login:

Use telnet protocol with the following addresses:

Router	IPv4 address
GSR-1	193.51.190.242
GSR-2	193.51.190.246
GSR-3	193.51.190.247
7200-2	193.51.190.249
7200-3	193.51.190.250
7200-4	193.51.190.253

Routers connection information

Login: 6diss

Password: 6diss

Task 1: Addressing configuration

1°) Configure the following addressing plan on the routers.

Loopbacks:

Name	IPv6 Loopback address	IPv4 Loopback address (for router-ID)
GSR-1	2001:660:3008:8001::1/64	194.254.101.130
GSR-2	2001:660:3008:8002::1/64	194.254.101.131
GSR-3	2001:660:3008:8003::1/64	194.254.101.132
7200-2	2001:660:3008:8007::1/64	194.254.101.133
7200-3	2001:660:3008:8008::1/64	194.254.101.134
7200-4	2001:660:3008:8009::1/64	194.254.101.135

Interconnections:

Interconnections (R1 - R2)	Prefix
GSR-1 - GSR-2	2001:660:3008:8101::/64
GSR-1 - GSR-3	2001:660:3008:8102::/64
7200-2 - GSR-2	2001:660:3008:8103::/64
GSR-2 - GSR-3	2001:660:3008:8104::/64
GSR-3 - 7200-3	2001:660:3008:8105::/64
GSR-3 - 7200-4	2001:660:3008:8108::/64

R1 has address = prefix::1

R2 has address = prefix::2

2°) *Check you can ping address of the routers connected to the router you manage.*

3°) *Take a look at the IPv6 details of an interface. Write down the different addresses you observe and give their types and usage.*



Task 2: OSPF configuration for IPv6

1°) Enable OSPFv3 routing protocol for IPv6 on all routers.

2°) Enable CEF switching for IPv6 on CISCO routers

3°) Enable the OSPFv3 process you have configured in question 1 on all interfaces of the lab (except loopback interfaces). Use area 0 for OSPFv3.

4°) Check OSPFv3 connections are established between routers.

5°) Redistribute the loopback addresses in OSPFv3.

6°) Check all routers in the labs receive all interconnection and loopback prefixes via OSPFv3.

7°) Check reachability of all routers loopback addresses from your router using ping command.

Task 3: BGP configuration for IPv6

1°) Configure an eMBGP peering between GSR-2 and GSR-1 and another peering between GSR-3 and GSR-1. For this purpose, interconnection addresses must be used to setup the peerings. Also note that:

- AS number of GSR-1 is 65152
- AS number of GSR-2 is 65153
- AS number of GSR-3 is 65154

Note that you have to disable OSPF in “external” interfaces:

- for GSR1, OSPF must be disabled in POS3/2 and POS3/3
- for GSR2, OSPF must be disabled in POS3/0 and POS3/1
- for GSR3, OSPF must be disabled in POS3/0 and POS3/1

2°) Configure an iMBGP peering between:

- GSR-2 and 7200-2
- GSR-3 and 7200-3
- GSR3 and 7200-4

Note: For iMBGP peerings, you have to specify the ipv6 address used for the BGP routing updates:

```
router bgp xxxx
...
...
address-family ipv6
...
...
neighbor X:X:X:X::X update-source Loopback 0
...
```

Note that the iMBGP full mesh is configured between loopback addresses of the routers. This is the reason why OSPF is needed to reach loopback addresses.

3°) Check the status of the eMBGP and iMBGP peerings. They must be in established state before going to the next step.

4°) Check that you receive prefixes via the eMBGP peerings. Check they are properly propagated to the routers of the lab through iMBGP peerings.

**Bonus:**

5°) Check the connectivity to the IPv6 internet. Use the ping / traceroute commands from the routers to some well known IPv6 web servers

- www.6diss.org
- www.renater.fr
- www.kame.net
- ...

6°) Enforce policies on the eMBGP peerings to accept only legacy IPv6 prefixes. Some more details about this legacy prefixes and the way you can configure the policy can be found at <http://www.space.net/~gert/RIPE/ipv6-filters.html>

7°) Apply a policy to prefer the path between GSR-1 and GSR-2. For this purpose, configure on GSR-2 the local-preference 200 on prefixes received from GSR-1. Configure on GSR-3 the local-preference of 150 on prefixes received from GSR-1.

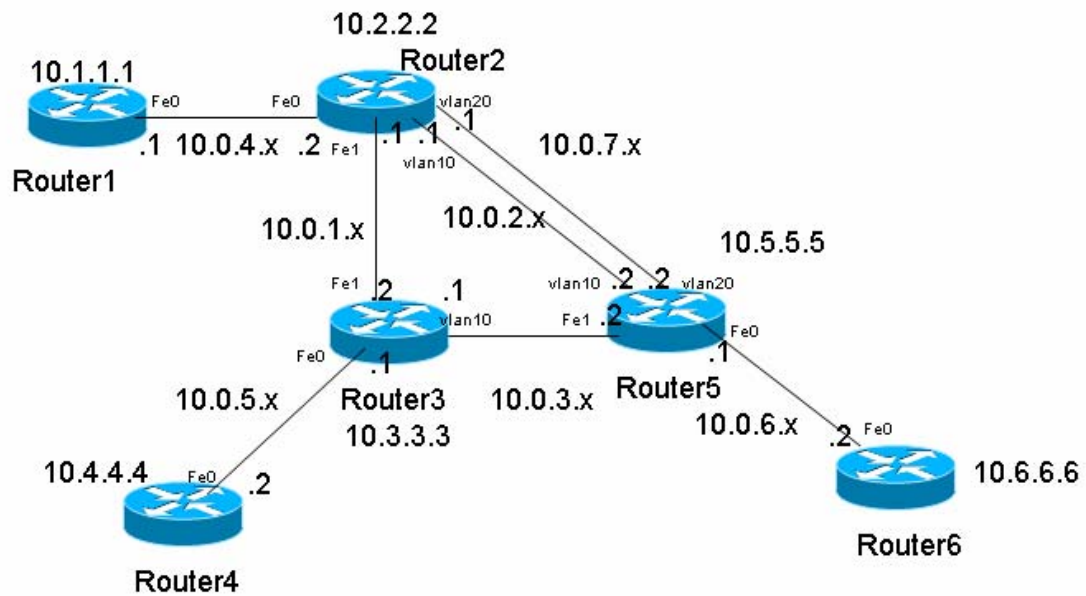
8°) Check the BGP details to make sure the policy is properly configured. Using traceroute command, make sure that the path between GSR-2 and GSR-1 is preferred.



4.2 Routing hands-on: Local lab

Laboratory Exercise: *Routing Configuration / Local laboratory*

Testbed setup



Testbed diagram

Routers login:

Use telnet protocol with the

Router	IPv4 loopback address	Ipv4 connection address
Router1	10.1.1.1	194.67.205.101
Router2	10.2.2.2	194.67.205.102
Router3	10.3.3.3	194.67.205.103
Router4	10.4.4.4	194.67.205.104
Router5	10.5.5.5	194.67.205.105
Router6	10.6.6.6	194.67.205.106

Routers connection information

Login: 6diss

Password: 6diss

Enable password: 6diss

Hints:

Connect to your router. Use the IPv6 auto configuration and plug your PC to any FastEthernet port on the router and telnet to it using the following authentication data:

Login: 6diss

Password: 6diss

Enable password: 6diss

The first step you must do is to see if your router has IPv6 routing active. The global `ipv6 unicast-routing` command should appear in the configuration.

Try to ping another router that is not directly connected to yours.

Note: do not modify Vlan1 configuration – you might lose connectivity to router

Task 1 :Addressing configuration

1^o) *Configure the following addressing plan on the routers.*

Loopbacks:

Name	IPv6 Loopback address	IPv4 Loopback address (for router-ID)
Router1	2001:DB8:CAFE:8001::1/64	10.1.1.1
Router2	2001:DB8:CAFE:8002::1/64	10.2.2.2
Router3	2001:DB8:CAFE:8003::1/64	10.3.3.3
Router4	2001:DB8:CAFE:8007::1/64	10.4.4.4
Router5	2001:DB8:CAFE:8008::1/64	10.5.5.5
Router6	2001:DB8:CAFE:8009::1/64	10.6.6.6

Interconnections:

Interconnections (R1 - R2)	Prefix
router1 - router2	2001:DB8:CAFE:8101::/64
router2 - router3	2001:DB8:CAFE:8102::/64
router2 - router5 (VLAN10)	2001:DB8:CAFE:8103::/64
router2 - router5 (VLAN20)	2001:DB8:CAFE:8104::/64
router3 - router4	2001:DB8:CAFE:8105::/64
router3 - router5	2001:DB8:CAFE:8106::/64
Router5 – router6	2001:DB8:CAFE:8107::/64

R1 has address = prefix::1

R2 has address = prefix::2

2°) *Check you can ping address of the routers connected to the router you manage.*

3°) *Take a look at the IPv6 details of an interface. Write down the different addresses you observe and give their types and usage.*



Task 2 :OSPF configuration for IPv6

1°) Enable OSPFv3 routing protocol for IPv6 on all routers.

Hint:

Activate OSPF on the interface

```
RouterX# enable
```

```
RouterX# configure terminal
```

```
RouterX(config)# interface fastethernet[X]
```

```
RouterX#(config-if)# ipv6 ospf processID area areaid
```

Where process_ID is the specific name of the OSPFv3 process you will configure.

2°) Enable CEF switching for IPv6 on CISCO routers

Hint:

Activate CEF on router

```
RouterX# conf t
```

```
RouterX(config)# ipv6 cef
```

3°) Enable the OSPFv3 process you have configured in question 1 on all interfaces of the lab (except loopback interfaces). Use area 0 for OSPFv3.

Hints:

If you look, you can see that the routing process is already created:

```
Router1# show configuration | inc ospf
```

```
ipv6 ospf 1000 area 0
```

```
ipv6 router ospf 1000
```

There are two lines, the one you configured before and the routing process that was automatically created.

```
Router1(config)# ipv6 router ospf 1000
```

```
Router1(config-rtr)#router-id 10.1.1.1
```

Hint2: Be sure that you use correct router-id!

4°) Check OSPFv3 connections are established between routers.

5°) Propagate the loopback addresses in OSPFv3.

Hints:

There are several ways to achieve this:



1. Redistribution

```
Router1(config)# ipv6 router ospf 1000
```

```
Router1 (config-rtr)# redistribute connected
```

```
Router1 (config-rtr)# redistribute static
```

Note: The routes from an interface will only be announced if that interface is up, or if you add its address to the routing table, for example by introducing a static route:

```
Router1(config)# ipv6 route 2001:DB8:CAFE:A::/64 null 0
```

2. Including in OSPFv3 with passive

```
Router1(config)# interface loopback0
```

```
Router1#(config-if)# ipv6 ospf 1000 area 0
```

```
Router1(config)# ipv6 router ospf 1000
```

```
Router1 (config-rtr)# passive-interface loopback 0
```

6°) Check all routers in the labs receive all interconnection and loopback prefixes via OSPFv3.

7°) Check reachability of all routers loopback addresses from your router using ping command.

Hints:

Step 1: Check OSPFv3 interfaces

```
Router1# show ipv6 ospf
```

```
It is an autonomous system boundary router
```

```
Originate Default Route with metric 100 always
```

```
(...)
```

```
Number of areas in this router is 1. 1 normal 0 stub 0 nssa
```

```
Area BACKBONE(0)
```

```
Number of interfaces in this area is 2
```

```
(...)
```

```
Router1# show ipv6 ospf interfaces
```

```
(...)
```

```
FastEthernet0 is up, line protocol is up
```

```
Link Local Address FE80::216:C8FF:FE30:5FC4, Interface ID 2
```

```
Area 0, Process ID 1000, Instance ID 0, Router ID 3.3.3.3
```

```
Network Type BROADCAST, Cost: 1
```

```
(...)
```

```
Designated Router (ID) 1.1.1.1, local address FE80::7D2
```

```
Backup Designated router (ID) 3.3.3.3, local address FE80::FC4
```

```
(...)
```

Step 2: Check OSPFv3 neighborsRouter3# **show ipv6 ospf neighbor**

Neighbor ID	Pri	State	Dead Time	Interface ID	Interface
4.4.4.4	1	FULL/BDR	00:00:30	2	Vlan32
1.1.1.1	1	FULL/DR	00:00:37	2	FastEthernet0

Step 3: Check the OSPFv3 databaseRouter1# **show ipv6 ospf database**

```

OSPFv3 Router with ID (1.1.1.1) (Process ID 1000)
      Router Link States (Area 0)
ADV Router    Age      Seq#          Fragment ID Link count Bits
  1.1.1.1     81      0x80000047 0              1          E
  3.3.3.3     76      0x80000040 0              1          E
  (...)

      Net Link States (Area 0)
ADV Router    Age      Seq#          Link ID      Rtr count
  1.1.1.1     87      0x80000008 2            2
  (...)

      Link (Type-8) Link States (Area 0)
ADV Router    Age      Seq#          Link ID      Interface
  1.1.1.1     1320   0x80000028 2            Fa0
  (...)

      Intra Area Prefix Link States (Area 0)
ADV Router    Age      Seq#          Link ID      Ref-lstype Ref-LSID
  1.1.1.1     327    0x80000008 1002         0x2002 2
  (...)

      Type-5 AS External Link States
ADV Router    Age      Seq#          Prefix
  1.1.1.1     563    0x80000006 2001:DB8:CAFE:A::/64
  (...)

```

Step 4: Looking at the routesRouter1# **show ipv6 route**

```

IPv6 Routing Table - 5 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
       U - Per-user Static route
       I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea, IS -
ISIS summary

```



O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 -
OSPF ext 2

ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2

OE2 2001:DB8:CAFE:A::/64 [110/20]

via FE80::217:E0FF:FED6:7D2, FastEthernet0

C 2001:DB8:CAFE:13::/64 [0/0]

via ::, FastEthernet0

L 2001:DB8:CAFE:13::3/128 [0/0]

via ::, FastEthernet0

L FE80::/10 [0/0]

via ::, Null0

L FF00::/8 [0/0]

via ::, Null0

8) Some useful commands

- To see the number of routes by prefix

```
Router1# show ipv6 route summary
```

- Forcing the SPF recalculation

```
Router1#clear ipv6 ospf process
```

- If you want to recalculate the SFP algorithm again, clear the OSPF database. If you type `clear ipv6 ospf force-spf` instead, the database will not be cleared before you run the SFP algorithm.

- Authentication neighbors on an interface

```
RouterX(config-if)# ipv6 ospf authentication ipsec spi 1000 md5  
12345678900987654321ascdefedcba0
```

- Where SPI value means Security Policy Index (a value between 256 and 4294967295) and the values after MD5 are the key in HEX format. One can also choose the SHA-1 algorithm instead of MD5.

- Authentication neighbors on an OSPF area

```
RouterX(config-rtr)# area 0 authentication ipsec spi 1000 md5  
12345678900987654321ascdefedcba0
```

Where SPI value means Security Policy Index (a value between 256 and 4294967295) and the values after MD5 are the key in HEX format. One can also choose the SHA-1 algorithm instead of MD5.

- Debug commands - Try these commands and analyse their output.

```
debug ipv6 ospf packets
```

```
debug ipv6 events
```

```
debug ipv6 ospf adj
```



Task 3 :BGP configuration for Ipv6

0°) Remove OSPFv3 configuration between router2 router3 and router5:

tip: disable ipv6 ospf on the necessary interfaces

1°) Configure an eMBGP peerings between router2, router3 and router5. For this purpose, interconnection addresses must be used to setup the peerings. Also note that:

- AS number of router2 is 65151
- AS number of router3 is 65152
- AS number of router5 is 65153

2°) Configure an iMBGP peering between:

- router1 and router2 (AS65151)
- router3 and router4 (AS65152)
- router5 and router6 (AS65153)

Note that the iMBGP full mesh is configured between loopback addresses of the routers.

Tips:

Configure the BGP main process on your router. Remember that in the case of MBGP you will have to create an IPv6 address family and configure a BGP router ID.

```
router bgp <as_number>
address-fammily ipv6
neighbor <neighbor> remote-as <as_number>
```

Also note, that if you don't have any IPv4 addresses on your router, you must configure a router ID, or your BGP process will not start and you'll get an error message "%BGP-4-NORTRID: BGP could not pick a router-id. Please configure manually."

```
bgp router-id <router_id>
```

Note:

With iBGP you should not calculate the next-hop. So all iBGP neighbours should be configured with next-hop-self option.

3°) Check the status of the eMBGP and iMBGP peerings. They must be in established state before going to the next step.

Tips:

- Check BGP Summary

See the status of your BGP process and how many routes you are receiving.

```
show bgp ipv6 unicast summary
```

Note: In case you are having trouble, look at your synchronization and auto-summary configuration.

- Check advertised routes

Look at the route you are advertising to your peer. Are they correct?

```
show bgp ipv6 unicast neighbor <neighbor> advertised-routes
```

- Check received routes

```
show bgp ipv6 unicast neighbor <neighbor> routes
```

Verify the routes you are receiving from your peers. Are they correct?

Is the AS Path for each route correct?

4°) *Advertise your route.*

Now advertise your routes to your peers.

(Tip: network ...)

The network you should use is listed in the following table:

Group	Advertised Network
1	2001:DB8:CAFE:1::/64
2	2001:DB8:CAFE:2::/64
3	2001:DB8:CAFE:3::/64
4	2001:DB8:CAFE:4::/64
5	2001:DB8:CAFE:5::/64
6	2001:DB8:CAFE:6::/64

5°) *Check that you receive prefixes via the eMBGP peerings. Check they are properly propagated to the routers of the lab through iMBGP peerings.*

6°) *Add another route to announce to your peer according to the following table:*

Group	Advertised Network
1	2001:DB8:CAFE:11::/64
2	2001:DB8:CAFE:12::/64



3	2001:DB8:CAFE:13::/64
4	2001:DB8:CAFE:14::/64
5	2001:DB8:CAFE:15::/64
6	2001:DB8:CAFE:16::/64

Check that you receive prefixes via the MBGP peerings.

See if you are advertising the route.

Now reset the BGP process.

How long does it take to have the peers exchanging routes again?

Perform a soft reset to the BGP process. What is the difference?

Remember that you can only advertise routes that you are able to announce. So if the network you are advertising is not being used, you must force it to be up. For example, to force the route on router 3, do:

```
Router3#(config)# ipv6 route 2001:DB8:CAFE:3::/64 Null0
```

Bonus:

5°) Enforce policies on the eMBGP peerings to accept only one loopback prefix (e.g: 2001:DB8:CAFE:8007::1/64).

6°) Apply a policy to prefer the path between router2 and router3 . For this purpose, configure on router3 the local-preference 200 on prefixes received from router2. Configure on router5 the local-preference of 150 on prefixes received from router2.

7°) Check the BGP details to make sure the policy is properly configured. Using traceroute command, make sure that the path between router2 and router3 is preferred.

Debug commands

- `debug bgp ipv6 updates`
- `debug bgp ipv6 neighbour 2001:DB8:CAFE:<Y>::1 updates in`
- `debug bgp ipv6 neighbour 2001:DB8:CAFE:<Y>::1 updates out`



4.3 Routing hands-on: Help Commands

A document including a commands glossary is distributed to the trainees for the routing hands-ons. This document helps the people who are not very familiar with the routing configuration. The labs are mainly composed of Cisco routers. There is also a Juniper and an Alcatel router, so the commands glossary includes a section for each vendor.

Routing configuration / commands glossary

Cisco commands

Enable IPv6 on an interface

```
interface xxxxx
  ipv6 enable
```

Configure an address

```
interface xxxxx
  ipv6 address X:X:X:X::X/<0-128> (general address)
  ipv6 address X:X:X:X::X (link-local address)
  ipv6 address autoconfig (auto-configuration)
```

Example (LAN interface)

```
interface Ethernet0/0
  ip address 192.168.1.254 255.255.255.0
  ipv6 address 2001:db8:123:1::2/64
```

Configure a tunnel

Configure an IPv6 in IPv4 tunnel

```
interface tunnel x
  tunnel source interface
  tunnel destination X.X.X.X
  ipv6 address X:X:X:X::X/<0-128>
  tunnel mode ipv6ip (for direct tunneling)
  tunnel mode gre ip (for gre encapsulation)
```

Configure an IPv6 in IPv6 tunnel

```
interface tunnel x
  tunnel source interface
  tunnel destination X.X.X.X
  ipv6 address X:X:X:X::X/<0-128>
  tunnel mode ipv6 (for direct tunneling)
  tunnel mode gre ipv6 (for gre encapsulation)
```

Enable IPv6 routing

```
ipv6 unicast-routing
```

Configure static routes

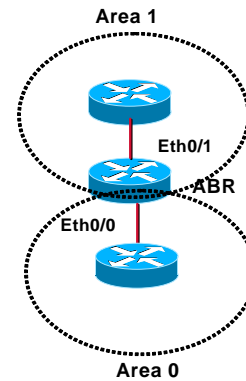
```
ipv6 route prefix/prefixlen next_hop
```



```
ipv6 route ::/0 2001:db8:10a:1001::1
```

Routing (OSPFv3)

```
interface Ethernet0/0
  ipv6 address 2001:db8:1:1::1/64
  ipv6 ospf 1 area 0
  !
interface Ethernet0/1
  ipv6 address 2001:db8:1:2::2/64
  ipv6 ospf 1 area 1
  !
ipv6 router ospf 1
  router-id 2.2.2.2
```



Routing (BGP)

```
router bgp xxxx
  no bgp default ipv4-unicast
  bgp router-id a.b.d.f
  neighbor X:X:X:X::X remote-as ...
  neighbor X:X:X:X::X ...
  address-family ipv6
    neighbor X:X:X:X::X activate
    neighbor X:X:X:X::X ...
    network 2001:db8::/32
    no synchronization
  exit
```

Routing policy filtering

```
ipv6 prefix-list bgp-in-6net seq 5 deny ::/0
```

Means filter ::/0 exactly

```
ipv6 prefix-list bgp-in-6net seq 10 deny 3FFE:300::/24 le 28
ipv6 prefix-list bgp-in-6net seq 15 deny 2001:db8::/35 le 41
ipv6 prefix-list bgp-in-6net seq 20 permit 2002::/16
ipv6 prefix-list bgp-in-6net seq 25 permit 3FFE::/17 ge 24 le 24
ipv6 prefix-list bgp-in-6net seq 30 permit 3FFE:8000::/17 ge 28 le 28
```

Means every prefix matching 3FFE:8000::/17 with length 28

```
ipv6 prefix-list bgp-in-6net seq 35 permit 3FFE:4000::/18 ge 32 le 32
ipv6 prefix-list bgp-in-6net seq 40 permit 2001::/16 ge 32 le 35
```

Means every 2001::/16 derived prefix, with length between 32 and 35

Access Control Lists

```
ipv6 access-list vty-ipv6
  permit tcp 2001:db8:0:401::/64 any eq telnet
  deny ipv6 any any log-input
```

Applying an ACL to an interface

```
ipv6 traffic-filter <acl_name> in | out
```



Restricting access to the router

```
ipv6 access-class <acl_name> in | out
```

Applying an ACL to filter debug traffic

```
debug ipv6 packet [access-list <acl_name>] [detail]
```

Show commands

```
show bgp
show bgp ipv6 unicast/multicast/all summary
show bgp ipv6 neigh <addr> routes
show bgp ipv6 neigh <addr> advertised-routes
show bgp ipv6 neigh <addr> received-routes
show ipv6 route
show ipv6 interface
show ipv6 neighbors
```



Juniper commands

Interface configuration

```

interfaces {
  name of interface {
    unit x {
      family inet {
        address X.X.X.X/prefixlength;
      }
      family iso {
        address Y.Y.Y.Y.Y.Y;
      }
      family inet6 {
        address Z.Z.Z.Z::Z/prefixlength;
      }
    }
  }
}

```

Router advertisements (stateless autoconfiguration)

```

protocols {
  router advertisement {
    interface interface name {
      prefix IPv6_prefix::/prefixlength;
    }
  }
}

```

Configure tunnel (with Tunnel PIC)

```

interface {
  ip-x/x/x {
    tunnel {
      source ipv4_source_address;
      destination ipv4_destination_address;
    }
    family inet6 {
      address ipv6_address_in_tunnel/prefixlength;
      gr-x/x/y/z {
        unit 0 {...}
      }
    }
  }
}

```

Static routes

```

Routing options {
  rib inet6.0 { -> Means IPv6 unicast routing table
    static {
      route IPv6_prefix next-hop IPv6_address;
    }
  }
}

Routing options {
  rib inet6.0 {
    static {
      route IPv6_prefix discard; -> Useful to originate a network
    }
  }
}

```

Routing (OSPFv3)

```

protocols {
  ospf3 {

```



```
preference 20;
area 0.0.0.0 {
interface ge-0/3/0.808 {
metric 100;
}
interface lo0.0 {
passive;
}}}}
```

Routing (BGP)

```
protocols {
  bgp {
    local-as local_AS_number;
    group EBGP_peers {
      type external;
      family inet6 {
        (any | multicast | unicast) }
      neighbor neighbor_IPv6_address;
      peer-as distant_AS_number;
      import in-PS;
      export out-PS; }
  }
}
```

Policy routing

```
policy statement in PS {
  term from_outside_accept {
    from {
      route-filter 2002::/16 exact;
      route-filter 3FFE::/17 prefix-length-range /24-/24;
      route-filter 3FFE:8000::/17 prefix-length-range /28-/28;
      route-filter 3FFE:4000::/18 prefix-length-range /32-/32;
      route-filter 2000::/3 prefix-length-range /16-/16;
      route-filter 2001::/16 prefix-length-range /29-/35; }
    then {
      accept; }
    then reject; }
}
```

Show commands

```
show bgp summary
show route advert bgp <addr>
show route rece bgp <addr>
show route table inet6.0 (terse)
show interfaces
show ipv6 neighbors
```



Alcatel commands

Enable IPv6 on a VLAN interface

```
vlan "number"  
show vlan  
vlan "number" port default a/b-c  
ipv6 interface "name" vlan number  
ipv6 address "2001:XXXX::3/prefix" "name"
```

Tunnels configuration

```
vlan "number"  
vlan "number" port default a/b  
ip interface "name-v4" vlan "number"  
ip interface "name-v4" address D.E.F.G mask H.I.J.K  
ipv6 interface "name-v6" tunnel "number"  
ipv6 interface "name-v6" tunnel source "@v4" dest. @v4"  
ipb6 address "2001:XXXX::3/prefix" "name-v6"
```

Router Advertisements and auto-configuration

```
show ipv6 interface "name"  
ipv6 interface "name" parameter numerical-value/yes/no  
ipv6 interface name ra-send no  
ipv6 interface ns-interval value  
ipv6 interface ra-interval value  
...
```

Starting routing

```
ipv6 route IPv6_prefix/length IPv6_address
```

Starting and configuring RIP routing

```
ipv6 load rip  
ipv6 rip status enable  
ipv6 rip interface "name of the IPv6 VLAN"  
show ipv6 routes  
  
ipv6 rip interface "name of the IPv6 VLAN" send-version\  
[v1|v2|v1compatible|none]  
ipv6 rip interface "name of the IPv6 VLAN" receive-version\  
[v1|v2|v1compatible|none]  
ipv6 rip interface "name of the IPv6 VLAN" metric [1-15]
```



DNS hands-on



5 DNS hands-on

Objectives

In this laboratory exercise you will complete the following tasks:

- *Create a forward zone*
- *Insert IPv6-related records*
- *Do some A and AAAA queries to the server*

Visual Objective

The following figure shows the topology of the current laboratory. This laboratory is similar with the ones on the first day. Now we have a PC performing as a DNS server.

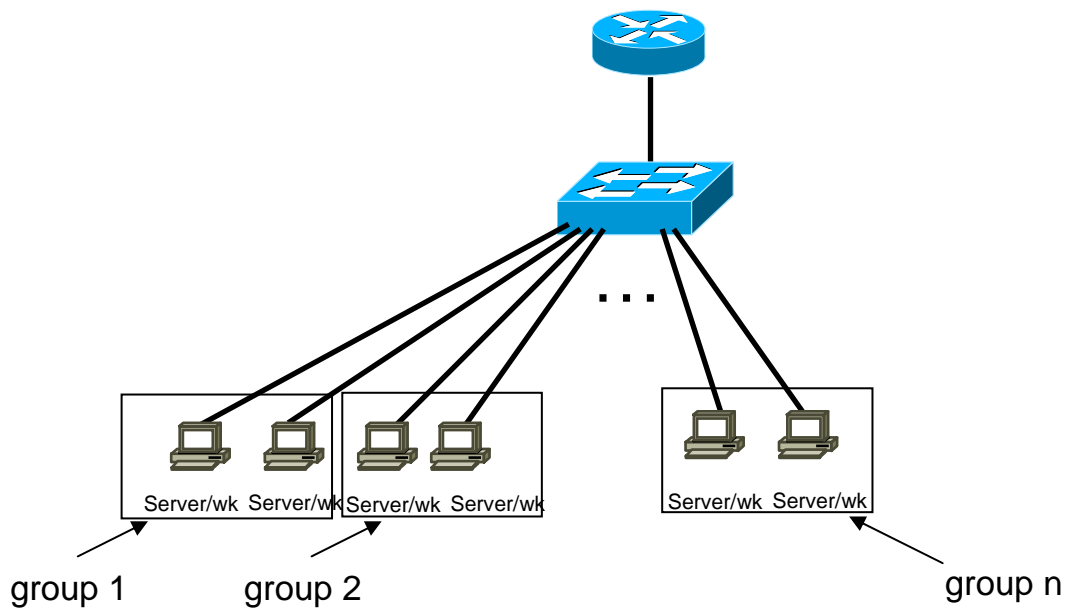


Figure 5: Scenario topology

Setup

Your IPv4 address is assigned by DHCPv4.

Scenario

*A laptop with Linux Ubuntu is running BIND 9. The access to the DNS server is allowed via ssh (port 22) at the addresses **2001:DB8:CAFE:1111::53** using the login/password: 6diss/6diss.*

*In the **/etc/bind/** directory there are already files with IPv4 DNS (A, PTR) records. Trainees should create the relevant IPv6 DNS (AAAA) records. (Students should have basic knowledge of BIND/DNS)*

*The DNS server is configured via the **/etc/bind/named.conf** file. The “forward” files that contain the DNS entries are defined in the **/etc/named.conf** via the following syntax:*

Forward zone: Example

```
zone "group3.rabat.6diss.org" in {  
    type master;  
    file "/etc/bind/rabat/group3";  
};
```

*In order to add IPv6 DNS entries for PC X in group Y, the zone file **groupY.rabat.6diss.org** has to be created. (The file **/etc/bind/named.conf** already contains the appropriate entries.)*



Task 1: *Create an IPv6 forward zone file and insert IPv6 records*

Complete the following exercise's steps:

- **Step 1:** In `/etc/bind/rabat/` directory of DNS Server create and populate the file correspondent to your zone (group`Y.rabat.6diss.org`). For your host's name use `pcX`. For example: `pc1.group4.rabat.6diss.org` (**Tip:** See configuration examples at the end of the document and group7 file in DNS server)
- **Step 2:** Validate the zone file using the command `named-checkzone`
- **Step 3:** Restart DNS server (**Tip:** In order to restart the DNS server, root privileges are required. DNS will be restarted on-demand by the trainer). Use the command `rndc` specifying your zone.
- **Step 4:** Validate DNS queries using your lab PC.

Summary

After completing these exercises, you should be able to:

- *Create a forward zone*
- *Insert IPv6 related records*



Appendix A: *Examples BIND files (DNS)*

Forward-zone file for workshop.org

```
workshop.org. IN SOA server.workshop.org. root.server.workshop.org. (
    1          ; Serial
    10800     ; Refresh after 3 hours
    3600      ; Retry after 1 hour
    604800    ; Expire after 1 week
    86400 )   ; Minimum TTL of 1 day
;
; Name servers
;
workshop.org. IN NS  server.workshop.org.
;
; Host addresses
;
localhost.workshop.org.    IN A    127.0.0.1
server.workshop.org.      IN A    192.168.38.5
pc28.workshop.org.       IN A    192.168.28.10
pc18.workshop.org.       IN A    192.168.28.10
;
; Multi-homed hosts
;
router1.workshop.org.     IN A    10.0.12.1
router1.workshop.org.     IN A    10.0.13.1
;
; Aliases
;
www                       IN CNAME server
;
; IPv6 host addresses
;
localhost.workshop.org.   IN AAAA  ::1
server.workshop.org.     IN AAAA  2001:DB8:CAFE:38::5
pc28.workshop.org.       IN AAAA  2001:DB8:CAFE:28::10
pc18.workshop.org.       IN AAAA  2001:DB8:CAFE:18::10
```



Reverse-zone file for workshop.org

```
1.1.1.1.E.F.A.C.8.B.D.0.1.0.0.2.ip6.arpa. IN SOA server.workshop.org. root.server.workshop.org. (
    1          ; Serial
    10800      ; Refresh after 3 hours
    3600       ; Retry after 1 hour
    604800     ; Expire after 1 week
    86400 )    ; Minimum TTL of 1 day
;
; Name servers
;
1.1.1.1.E.F.A.C.8.B.D.0.1.0.0.2.ip6.arpa. IN NS  server.workshop.org.
;
; Addresses point to canonical name
;
0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.1.1.1.E.F.A.C.8.B.D.0.1.0.0.2.ip6.arpa.      IN  PTR
pc28.workshop.org.
5.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.1.1.1.E.F.A.C.8.B.D.0.1.0.0.2.ip6.arpa.      IN  PTR
server.workshop.org.
```



Applications hands-on



6 Application/Services hands-on

Objectives

In this laboratory exercise you will complete the following tasks:

- *Configure and run an IPv6 (virtual) web server*
- *Configure and run an IPv6 ftp server*
- *Test services with IPv6 clients (web, ftp)*

Visual Objective

The following figure shows the topology of the current laboratory.

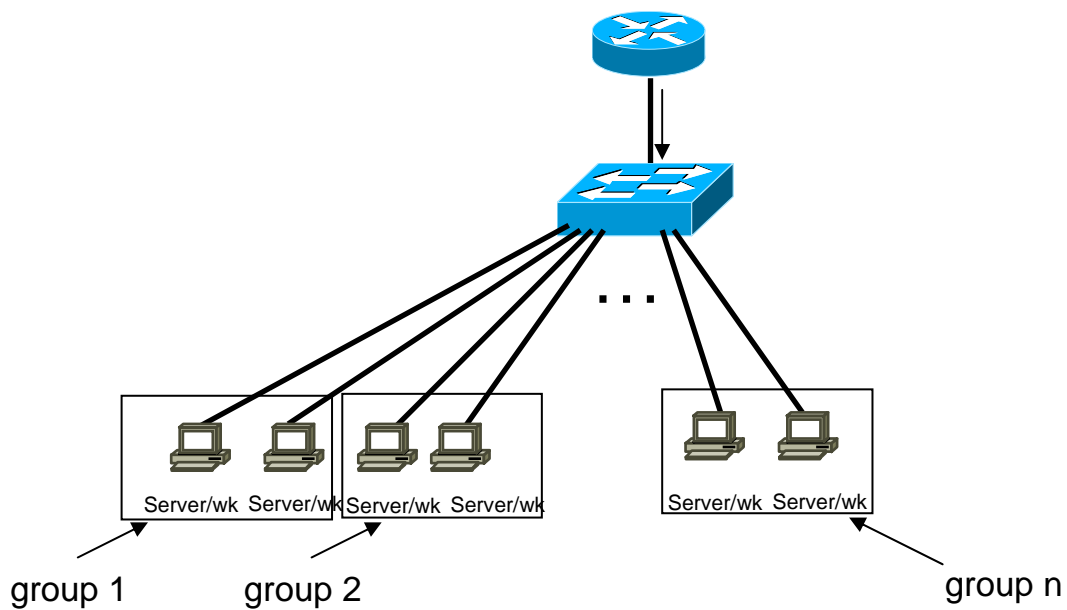


Figure 6: Scenario topology



Setup/Scenario

For this hands-on, you will work by binomial (server/workstation). In the server, you will configure the services. The workstation will be used to check that the services are running well. For this, you will install client applications like web browsers, ftp clients, etc.

The server and the workstation are running a Linux distribution.

For each PC, you will install the server and client applications.

Step 1 - Configure the following IPv6 address in your eth0 interface and update the DNS server according your addresses:

Group/PC	IPv6 Address
1/1	2001:DB8:CAFE:1111::0011/64
1/2	2001:DB8:CAFE:1111::0012/64
...	...
N/M	2001:DB8:CAFE:1111::00NM/64

With: N groupID and $M = 1$ or 2 (PCs in the group)

Task 1: Server Configuration: Configure and launch a web server (apache2)

- **Step 1:** Install apache2
`~#apt-get install apache2`
- **Step 2:** In /etc/apache2/ directory, check that the file ports.conf is not IPv4 specific:
e.g: `Listen 80`

remark: With this configuration, the web server will listen IPv4 and IPv6 addresses. If you want that web server only listen IPv6 addresses you have to specify the web server IPv6 address in ports.conf:

e.g: `Listen [2001:db8:CAFE:1111::nm]:80`

- **Step 3:** Launch apache2 daemon:
`/etc/apache2#apache2ctl start`



- **Remark:** The configuration of IPv6 virtual hosts is similar to IPv4 configuration: In `/etc/apache2/sites-enabled/default` file, you have to specify the IPv6 web server address and update the DNS server: Here is an example (The DNS has to be updated: two names for the same server)

```
NameVirtualHost [2001:db8:CAFE:1111::nm]:80
```

```
NameVirtualHost 192.168.X.Y:80
```

```
<VirtualHost [2001:DB8:CAFE:1111::nm]:80 192.168.x.y.:80>
```

```
    ServerName webA-M.groupN.rabat.6diss.org
```

```
    DocumentRoot /www/
```

```
</VirtualHost>
```

```
<VirtualHost [2001:DB8:CAFE:1111::nm]:80 192.168.1.1:80>
```

```
    ServerName webB-M.groupN.rabat.6diss.org
```

```
    DocumentRoot /www/rabat
```

```
</VirtualHost>
```

Task 2: Server Configuration: Configure and launch an FTP server

- **Step 1:** Install the IPv6 compliant FTP server Proftpd

```
~#apt-get install proftpd
```

- **Step 2:** Launch the FTP server

```
~# proftpd&
```

Task 1': Workstation configuration: install and test a web browser (parallel task with Task 1)

- **Step 1:** Download and install a web browser supporting IPv6

Download and Install Firefox (<http://www.mozilla.org/firefox/>)

- **Step2:** Check that you can reach the web server in IPv6 (of your neighbor)



After having installed Firefox, try to access the web server of your group using IPv6 support. To do that, you must use this syntax:

`http://[2001:DB8:CAFE:1111::NM]`



The square brackets avoid the confusion with the “.” of protocol, address and the port number.

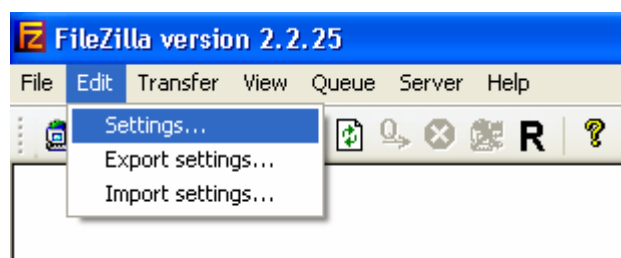
- **Step 3:** Analyse with wireshark the packets and check that the TCP connection on port 80 is over IPv6.
- **Step 4:** Test that you can also access the web server with IPv4 address
- **Step 5:** Modify the web server configuration to only listen in the IPv6 address:

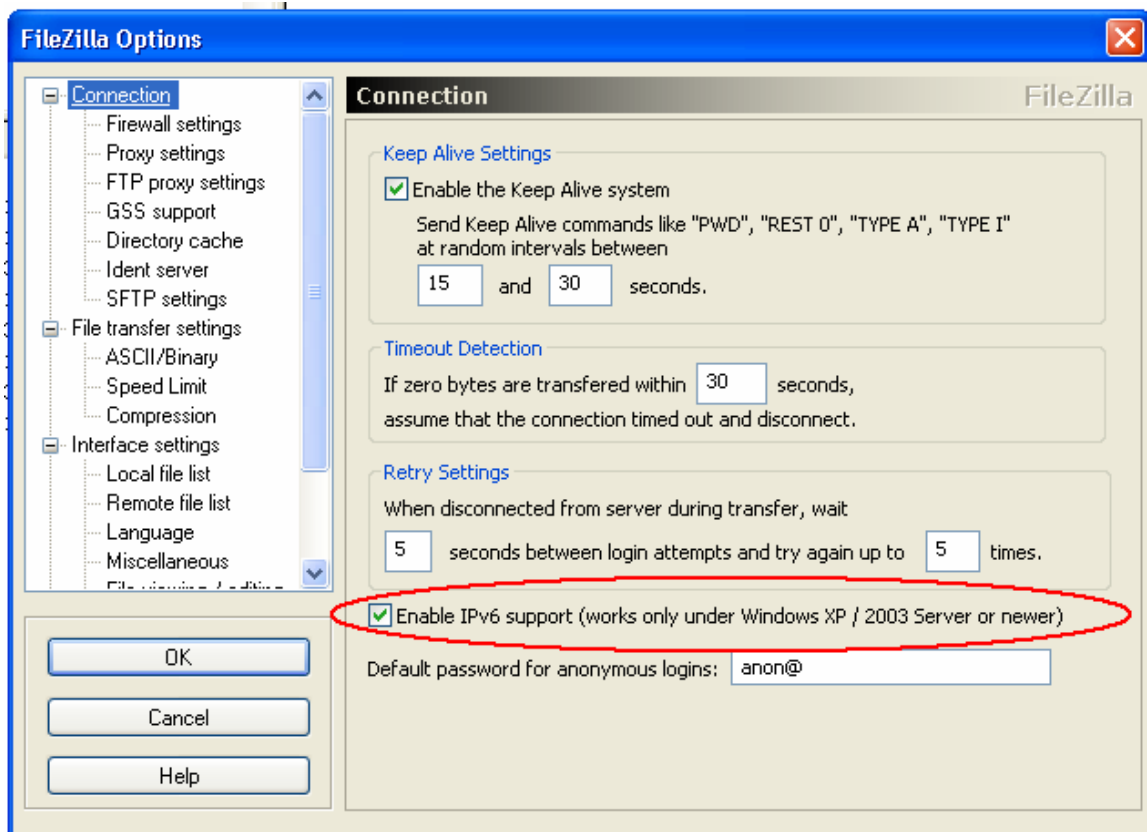
`Listen [2001:DB8:CAFE:1111::NM]:80`

- **Step 6:** Check that you can only access the web service with IPv6 address.

Task 2': Workstation configuration: install and test an FTP client (parallel task with task 2)

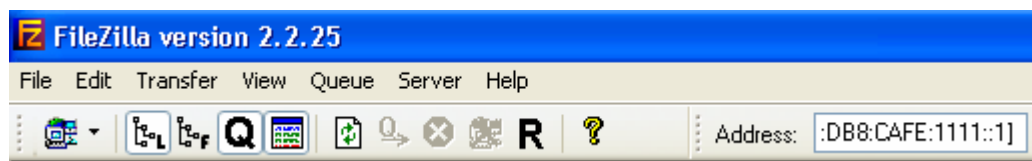
- **Step 1:** Download and install an IPv6 FTP client
 - Download and install Filezilla (<http://filezilla.sourceforge.net>)
 - Check that IPv6 is enable:





- **Step2:** Check that you can reach the ftp server in IPv6

Try to access the FTP server of your group using IPv6 support. To do that, you must use this syntax in the address box: [2001:DB8:CAFE:1111::NM]



- **Step 3:** Use wireshark to analyse the IPv6 requests and responses.

Summary

After completing these exercises, you should be able to:

- *Configure and run an IPv6 servers (web, ftp)*
- *Install and configure IPv6 client (ftp clients, web browsers)*



Management hands-on



7 Management hands-on

Objectives

In this laboratory exercise you will complete the following tasks:

- *Install a tool to monitor machines and associated services (argus)*
- *Test additional tools (Looking glass, ASPathTree, etc)*

Visual Objective

The following figure shows the topology for this exercise.

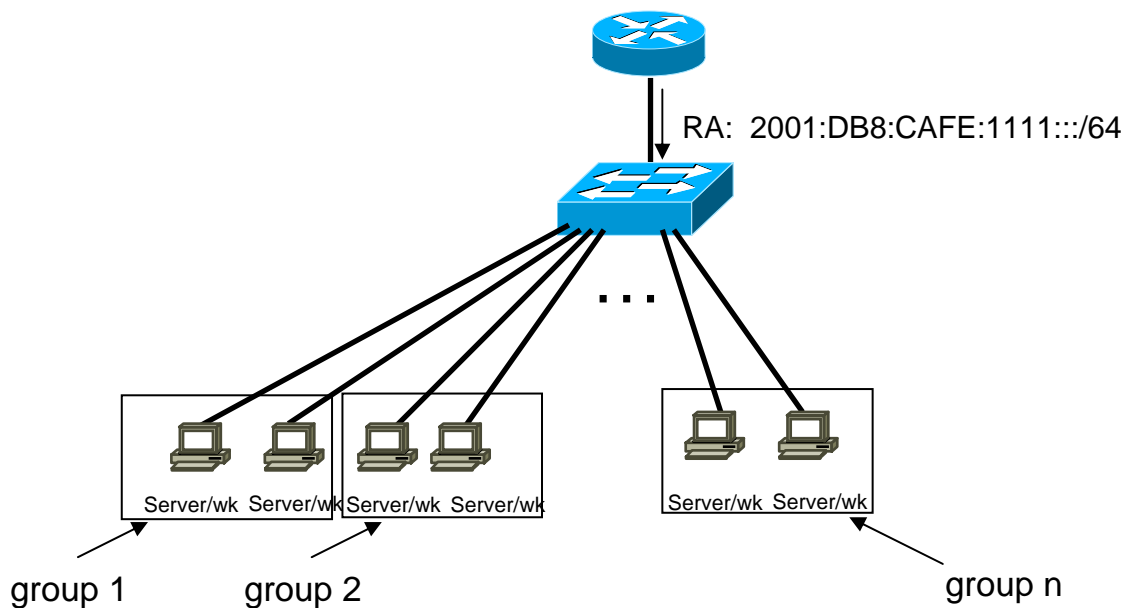


Figure 7: Scenario topology



Setup/Scenario

The main goal of this hands-on is to monitor the router, the PCs and the services installed during the last hands-on on the LAN.

All the PCs must run Linux.

After having installed a tool to do that, you will test additional monitoring tools available in the web.

Task 1: Install a tool to manage the equipment and the services

Complete the following exercise's steps:

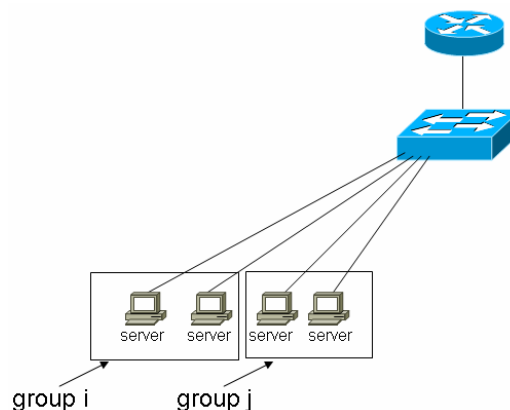
- **Step 1:** Install Argus (<http://argus.tcp4me.com/>) in your PC
 - Download argus (version 3.4)
 - Untar: `tar -xvzf argus-3.4.tar`
 - Install argus: `./Configure, make, make install`. Note that you must be root for the last one.

Note: For running Argus in this hands-on, you'll need:

- i. Perl 5
- ii. Fping and fping6
- iii. Apache

During the Argus installation, if these packages are not installed in your PC, the Argus installer will propose you to install them.

- **Step 2:** Configuration of Argus (creation of a file to monitor the router, PCs and services running (web, ftp), "user" file and Apache configuration).
 - You have to monitor 2 groups and the router:





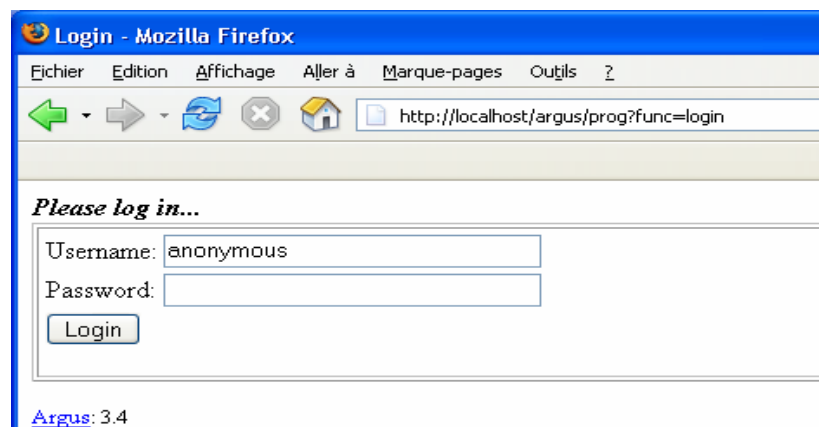
- a “server” of group i will monitor:
 - The router: **ping**
 - The other server of group i and the servers of group j : **ping** and services (**web** and **ftp**)

NB: You can start monitoring only one sever and check if Argus is running well.

- To do that, retrieve the IPv4 and IPv6 addresses of the router and your neighbours (group i and j)
- Create the configuration file of Argus (see Appendix A):
 - Copy the file “config.example” to “config”
 (Tip 1: Use the command `argus-config --datadir` to find the config folder ...)
 - Modify the file “config” to integrate new services supervision
- Create the user file:
 - Copy the file “users.example” to “users”
 - Modify the file “users” to add your own user.
- **Step 3:** Web server configuration:
 - Be sure that \$datadir is writable by the `www` user (or whatever uid your web server runs as)
 - Copy icons to somewhere accessible by your web server (these locations get specified in config file) or feel free to replace them with your own icons, or no icons at all.
 - Create a symlink in your cgi-bin directory to the arguscgi program: You can add in the web server (apache2) config file a ScriptAlias line:
 - e.g: `ScriptAlias /argus/prog /usr/local/sbin/arguscgi`
- **Step 4:** Run argus daemon:
 - `#argusd &`
 - Check with your web browser that argus is running well:

<http://localhost/argus/prog?func=login>

You will obtain the login screen:





(**Tip 1:** See the log file if you encountered some problems ...)

- **Step 5:** Play with Argus
 - Put some PCs or services down and check that Argus detects the failovers.

Task 2: Test additional management tools or applications available in the web

Complete the following exercise's steps:

- **Step 1:** Access an AS Path web site. Find the number of IPv6 AS numbers and the number of routing entries in the IPv6 routing table (e.g: <https://noc.cssi.renater.fr/AS-Path-Tree/>)
- **Step 2:** Use a Looking Glass web site. Identify the different IPv6 requests that you can achieve (e.g: <https://noc.cssi.renater.fr/-l-g-v-6-.php>)
 - Try to ping or traceroute an IPv6 address
 - Retrieve the IPv6 BGP table
 - ...

Summary

After completing these exercises, you should be able to:

- *Install and configure monitoring tools (argus) for manage a network*
- *Find useful information in the web concerning IPv6 management tools*



Appendix A: Configuration file for argus (Management)

```
Group "Routers" {
    #Router supervision
        Host "GW_IPv6" {
            hostname: 2001:db8:CAFE:1111:...
            graph: yes
            Service Ping
        }
}

Group "Group 1" {
    #Server supervision
        Host "server_IPv4" {
            hostname: x.x.x.x      # IPv4 address
            graph: yes

            # Services that you want to monitor
            Service TCP/HTTP
            Service TCP/FTP
            Service Ping

            # Example to monitor DNS service
            Service UDP/DNS{
                zone:  rabat.workshop.6diss.org
                class:  IN
                query:  AAAA
            }
        }

        Host "server_IPv6" {
            hostname: 2001:db8:CAFE:1111:...
            graph: yes

            # Services that you want to monitor
            Service TCP/HTTP
            Service TCP/FTP
            Service Ping

            # Example to monitor DNS service
            Service UDP/DNS{
```



```
zone:  rabat.workshop.6diss.org
class: IN
query: AAAA
}
}
```



Security hands-on



8 Security hands-on

Task 1: Securing the servers

1°) *Boot on linux, check that the IPv6 connectivity is fine.*

2°) *From application hands-on, a web server should be running on your host. Add filters to make sure that your server filters incoming IPv6 packets except TCP 80 packets that access the server (help: Appendix A)*

3°) *Ask other students to check that rules are properly working.*

4°) *Now configure the filter to allow TCP port 80 to be accessed for IPv6 except from certain groups of the room*

5°) *Again ask other students to check that rules are properly working.*

Task 2: Securing the network

Laboratory information

Refer to the routing laboratories for getting access parameters.

Access list configuration

1°) *Take an IPv4 access-list that you have configured in your network. Build a similar access-list for IPv6 and configure it on the lab router you manage. Don't apply it on an interface to avoid causing access damage to the router (Help: Appendix B).*



Appendix A: *Configuring iptables (security)*

For this case, we will use the table « Filter » that includes these three default chains :

- INPUT : control ingoing packets
- FORWARD : Filter packets crossing a network interface to another
- OUTPUT : control outgoing packets.

Remark : It is possible to create additional chains if necessary.

The command that permit to manipulate the tables is *iptables*. This command can be used in CLI, but it is recommended to have a script generating the tables.

When we launch the script, these actions are executed :

- Clean the table : clear the tables : **iptables -F** and delete the non-default chains : **iptables -X**
- Add new chains : **iptables -N name-of-chain + actions**
- Definition of policy : Generally **DROP** or **ACCEPT**, using **iptables -P name-of-chain action**
- Definition of local usage : **Iptables -A name-of-chain rule**

Example of script

```
#!/bin/bash

# low ports
dw_ports="0:1023"

# high ports
up_ports="1024:65535"

start_fw()
{

# clearing rules for all the tables**
iptables -F

# permit to delete the non-default chains (not included in table filter e.g
LOG_ACCEPT)
iptables -X

# add a new chain that log in syslog what it is accepted
iptables -N LOG_ACCEPT
iptables -A LOG_ACCEPT -j LOG -m limit --limit 500/hour --log-level 6 -log-
prefix "[iptables-accept]"
iptables -A LOG_ACCEPT -j ACCEPT

# update policies : by default all is rejected
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# The local processes can communicate between them
```



```
# entre eux via l'interface locale :
ip6tables -A INPUT -i lo -j ACCEPT
ip6tables -A OUTPUT -o lo -j ACCEPT

# SSH access from DMZ and log
ip6tables -A OUTPUT -p tcp --sport ssh -d X:X:X:X::/64 --dport $sup_ports ! --syn -
j LOG_ACCEPT
ip6tables -A INPUT -p tcp --dport ssh --sport $sup_ports -s X:X:X:X::/64 -j
LOG_ACCEPT

#DNS access UDP 53 to 53
ip6tables -A OUTPUT -p udp --dport domain --sport domain -j ACCEPT
ip6tables -A INPUT -p udp --sport domain --dport domain -j ACCEPT

#DNS access udp 53 and tcp 53 (long requests)
ip6tables -A OUTPUT -p udp --dport domain --sport $sup_ports -j ACCEPT
ip6tables -A INPUT -p udp --sport domain --dport $sup_ports -j ACCEPT
ip6tables -A OUTPUT -p tcp --dport domain --sport $sup_ports -j ACCEPT
ip6tables -A INPUT -p tcp --sport domain --dport $sup_ports -j ACCEPT

# DNS server udp 53 and tcp 53 (long requests)
ip6tables -A INPUT -p udp --dport domain --sport $sup_ports -j ACCEPT
ip6tables -A OUTPUT -p udp --sport domain --dport $sup_ports -j ACCEPT
ip6tables -A INPUT -p tcp --dport domain --sport $sup_ports -j ACCEPT
ip6tables -A OUTPUT -p tcp --sport domain --dport $sup_ports -j ACCEPT

# FTP for DMZ
ip6tables -A OUTPUT -p tcp --dport 21 --sport $sup_ports -d X:X:X:X::/64 -m state
--state NEW,ESTABLISHED -j ACCEPT
ip6tables -A INPUT -p tcp --sport 21 --dport $sup_ports -s X:X:X:X::/64 -m state -
-state ESTABLISHED -j ACCEPT

# HTTP and HTTPS
ip6tables -A INPUT -p tcp --sport http --dport $sup_ports -m state --state
ESTABLISHED,RELATED -j ACCEPT
ip6tables -A OUTPUT -p tcp --dport http --sport $sup_ports -j ACCEPT
ip6tables -A INPUT -p tcp --sport https --dport $sup_ports -m state --state
ESTABLISHED,RELATED -j ACCEPT
ip6tables -A OUTPUT -p tcp --dport https --sport $sup_ports -j ACCEPT

# ICMPv6
ip6tables -A INPUT -p icmpv6 -icmpv6-type echo request -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 -j ACCEPT

# Packets rejected : log in syslog
ip6tables -A INPUT -j LOG -m limit --limit 500/hour --log-level 6 --log-prefix
"[ip6tables-in-reject]"
ip6tables -A OUTPUT -j LOG -m limit --limit 500/hour --log-level 6 --log-prefix
"[ip6tables-out-reject]"

}

load_modules()
{

echo -en " Loading modules : "
echo -en "ip_tables, " ; /sbin/modprobe ip_tables
echo -en "ip_conntrack, " ; /sbin/modprobe ip_conntrack
```



```
echo -en "ip_contrack_ftp, " ;/sbin/modprobe ip_contrack_ftp
echo "ipt_limit" ; /sbin/modprobe ipt_limit

}

stop_fw()
{

# clearing rules for the tables :
ip6tables -F

# Delete the non-default chains (not in filter chain)
ip6tables -X

# Initial policy : ACCEPT
ip6tables -P INPUT ACCEPT
ip6tables -P OUTPUT ACCEPT
ip6tables -P FORWARD ACCEPT

}

case "$1" in

start)

load_modules
start_fw
echo "firewall started"

stop)

stop_fw
echo "firewall stopped"

restart)

stop_fw
echo "firewall stopped"
load_modules
start_fw
echo "firewall restarted"

*)

echo "usage: $0 [start|stop|restart]" >&2

;;

esac
```



Appendix B: Configuring Access-lists (Cisco)

```
Router1# configure terminal
Router1(config)# ipv6 access-list v6test
Router1(config-ipv6-acl)# permit ipv6 host 2001:DB8:CAFE:3::1 any
Router1(config-ipv6-acl)# permit ipv6 host 2001:DB8:CAFE:13::3 any
Router1(config-ipv6-acl)# permit ipv6 host 2001:DB8:CAFE:34::3 any
Router1(config-ipv6-acl)# exit
```

Applying Access Lists to your interfaces

In IPv4 the command to apply an access-list to an interface was *ip access-group <access-list_number_or_name> <in|out>*. In IPv6 the command is *ipv6 traffic filter*.

```
Router1(config)# interface FastEthernet1
Router1(config-if)# ipv6 traffic-filter v6test in
```

If you notice, you don't peer with any neighbors now. The reason is that the updates are sent via link local addresses. You have to permit these packets to your router.

```
Router1(config)# ipv6 access-list v6test
Router1(config-ipv6-acl)# permit ipv6 FE80::/16 any
```

Checking your Access-lists

To explicitly deny any IPv6 ICMP configure:

```
Router1(config)# ipv6 access-list v6test
Router1#(config-ipv6-acl)# deny icmp any any
```

To check the access-list counters do:

```
Router1# show ipv6 access-list v6test
IPv6 access list v6test
permit ipv6 host 2001:DB8:CAFE:3::1 any sequence 10
permit ipv6 host 2001:DB8:CAFE:13::3 any (7 matches) sequence 20
permit ipv6 host 2001:DB8:CAFE:34::3 any sequence 30
permit ipv6 FE00::/8 any (10 matches) sequence 40
deny icmp any any (5 matches) sequence 50
```