# IPv6

This lab session focuses on key aspects of IPv6 networking. We will cover link-local configuration, setting up and managing IPv6 addresses, conducting Neighbor Discovery Protocol experiments, configuring DNS for IPv6, and exploring advanced IPv6 tunneling techniques. The following is the requirements :
— For each experiment, take a screenshot of the successful execution and provide answers to open-ended questions to demonstrate your understanding.
— Each student must submit a detailed report by February 16, 2025.
— Please clearly label each question number in your report.
— This lab contributes 30% to your final grade.

## 1 Prerequisites

Before proceeding, ensure you have :
— **Oracle VirtualBox** installed. Download from : `https://www.virtualbox.org/`
— At least **8GB of free RAM** and **20GB of disk space**.
— The provided OVA files :
  — `IPv6-PC1.ova` (≈2.17GB)
  — `IPv6-PC2.ova` (≈2.18GB)

Follow these steps to import the virtual machines :

1. Open **Oracle VirtualBox**.
2. Click on **File** → **Import Appliance**.
3. Click on **Choose** and select the `IPv6-PC1.ova` file.
4. Click **Next** and verify the VM settings.
5. Click **Import** and wait for the process to complete.
6. Repeat the same steps for `IPv6-PC2.ova`.

### 1.1 Configuring the Network

1. Ensure both VMs are properly imported and listed in VirtualBox.
2. Select each VM, go to **Settings** → **Network**.
3. Adapter 1 should be set to **NAT** (for internet access).
4. Adapter 2 should be set to **Internal Network** with the name `ipv6lab`.
5. Click **OK** to save changes.

### 1.2 Starting the Virtual Machines

1. Select `IPv6-PC1` and click **Start**.
2. Select `IPv6-PC2` and click **Start**.
3. Open the terminal in each VM and run : `ip a`
4. Verify that both VMs have IPv6 addresses assigned.

### 1.3 Testing Connectivity

1. On `IPv6-PC1`, try to ping `IPv6-PC2` using : `ping6 <IPv6-ADDRESS-OF-PC2>`
2. If successful, you should see ICMPv6 echo replies.
3. If it fails, check network configurations.

After successfully importing and configuring the virtual machines, you are ready to proceed with the experiments. Alternatively, you may use your own Linux-based system to complete the exercises.

## 2 Link-local autoconfiguration

The goal of this experiment is to understand how IPv6 devices automatically configure link-local addresses, which are essential for local network communication. Through hands-on testing, we will : 1) Verify the presence and purpose of link-local IPv6 addresses ; 2) Verify the EUI-64 address generation process ; 3) Use `tcpdump` to capture ICMPv6 messages during IPv6 initialization ; 4) Test connectivity between two virtual machines using their link-local addresses.

The `ip a` command shows you all the information about all the network cards recognized on your system [1]. This experiment is conducted using two Virtual Machines (VMs), each configured as an independent IPv6 node. Each VM has :

— Adapter 1 : NAT (for internet access).
— Adapter 2 : Internal Network (to communicate using IPv6).

1. Verify the Loopback Address
   Every system has a loopback address (‘ : :1/128‘) that allows it to communicate with itself. Verify the presence of this address.

```
# Command
ip a show lo
# Expected Output
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
```

2. Identify the MAC Address of enp0s8
   Identify the MAC address of your network interface and explain its role in IPv6.

```
# Command
ip link show enp0s8
# Expected Output
enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
link/ether 08:00:27:11:2c:19 brd ff:ff:ff:ff:ff:ff
```

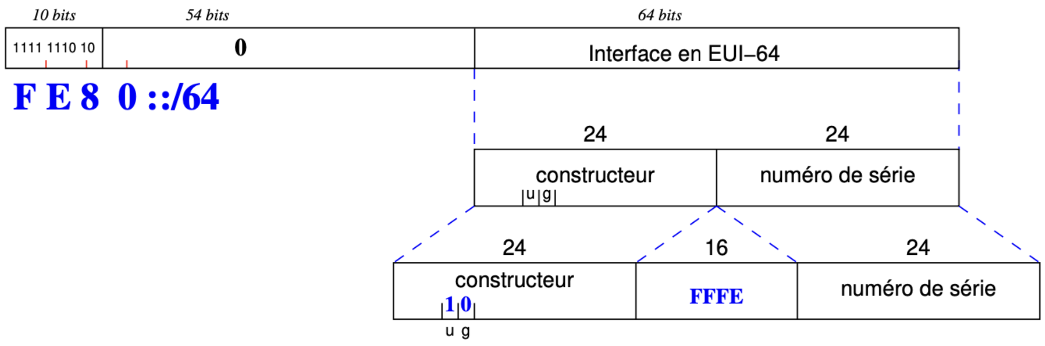
3. Enable IPv6 on enp0s8
   Ensure that IPv6 is enabled on your interface.

```
# Command:
sudo sysctl -w net.ipv6.conf.enp0s8.disable_ipv6=0
```

4. Check the Link-Local IPv6 Address
   IPv6 automatically assigns a link-local address (‘fe80 : :/64‘). Verify the assigned link-local address.

```
# Command
ip -6 a show enp0s8
# Expected Output
enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
inet6 fe80::a00:27ff:fe6d:6df9/64 scope link
valid_lft forever preferred_lft forever
```

5. Validate the Link-Local Address Format
   The link-local address should follow the EUI-64 format, derived from the MAC address.
   — Reference Format : FE80 : :/64 + Interface Identifier (Derived from MAC Address)



1. `ifconfig -a` is deprecated and the official recommendation is to use the ip command set.

6. Capture Neighbor Solicitation Messages
   Use `tcpdump` to monitor ICMPv6 messages generated during IPv6 initialization.

```
# On VM1 (\texttt{enp0s8} interface):
sudo tcpdump -i enp0s8 icmp6
# On VM2 (\texttt{enp0s8} interface), force IPv6 address discovery
sudo sysctl -w net.ipv6.conf.enp0s8.disable_ipv6=0
```

7. Test IPv6 Connectivity Between Two VMs
   Use `ping6` to test communication between the VMs using link-local addresses.

```
# On VM1, ping VM2's enp0s8 address
ping6 -c 4 fe80::a00:27ff:feXX:XXXX%enp0s8

# Expected output
PING fe80::a00:27ff:feXX:XXXX%enp0s8 (fe80::a00:27ff:feXX:XXXX%enp0s8): 56 data
     bytes
64 bytes from fe80::a00:27ff:feXX:XXXX%enp0s8: icmp_seq=1 ttl=64 time=0.2 ms
64 bytes from fe80::a00:27ff:feXX:XXXX%enp0s8: icmp_seq=2 ttl=64 time=0.3 ms
```

**Questions :**

1. **Loopback Address**
   — What is the purpose of the loopback address ?
   — Is there any scenario where a device **must** use the loopback address ?
2. **MAC Address in IPv6**
   — Why do we need a MAC address in an IPv6 network ?
   — What does the broadcast (‘BROADCAST‘) and multicast (‘MULTICAST‘) flag indicate ?
   — What does the ‘ff :ff :ff :ff :ff :ff‘ broadcast address mean ?
3. **Link-Local Addressing**
   — Why does every IPv6-enabled interface automatically get a link-local address ?
   — Why is there no need for DHCP to assign link-local addresses ?
   — How is the link-local IPv6 address generated from the MAC address ?
4. **Neighbor Solicitation Messages**
   — What is the purpose of Neighbor Solicitation (NS) ?
   — What is the destination address of Neighbor Solicitation messages ?

## 3 Neighbor Discovery

This experiment is designed to demonstrate the IPv6 Neighbor Discovery (ND) process, similar to ARP in IPv4 but for IPv6 networks. Using two virtual machines, we will explore how IPv6 nodes discover each other on the same local link and how they manage IP address resolution and duplication detection.

1. **Step 1 : Viewing Neighbor Cache.**
   Display the current entries in the neighbor cache to understand how IPv6 nodes maintain knowledge of each other. Use the following command :

   ```
   ip -6 neigh show
   ```

   This command lists the IPv6 addresses, their associated link-layer addresses, the interface through which they are reachable, and the status of each entry (permanent or temporary).
2. **Step 2 : Testing Connectivity and Observing Cache Updates.**
   Test network connectivity by sending a ping from one VM to another. This action should generate Neighbor Discovery Protocol traffic, updating the neighbor cache. Observe changes in the cache :

   ```
   # On VM1, ping VM2's enp0s8 IPv6 address
   ping6 -c 4 fe80::a00:27ff:feXX:XXXX%enp0s8
   ip -6 neigh show
   ```

   This tests the accessibility of another machine on the network and triggers updates to the neighbor cache, which can be viewed again to see new or updated entries.

3. **Step 3 : Capturing Neighbor Discovery Messages.**
   Use tcpdump to monitor ICMPv6 Neighbor Discovery messages, which are crucial for the address resolution process :
   (a) Start tcpdump on VM1 to capture all ICMPv6 messages :
   ```
   sudo tcpdump -i enp0s8 icmp6
   ```
   (b) From VM2, initiate traffic to VM1 to trigger Neighbor Discovery Protocol messages :
   ```
   ping6 fe80::a00:27ff:feXX:XXXX%enp0s8
   ```
   (c) Document the source and destination addresses of the Neighbor Solicitation and Advertisement messages.
4. **Step 4 : Additional Exercises.**
   Ping the multicast address 'ff02 : :1' to observe responses from all devices on the local link.
   ```
   ping6 -I enp0s8 -c 4 ff02::1
   ```

**Questions :**

(a) What are the source and destination addresses of the ICMPv6 "Neighbor solicitation" message ?
(b) What are the source and destination addresses of the ICMPv6 "Neighbor Advertisement" message ?
(c) Explain the States (FAILED, DELAY, STALE) in the Neighbor Cache :
(d) **Bonus :** Add a new network adapter set to 'Bridged' mode in your VM settings. This configuration allows your VM to function like a physical PC on the network, enhancing its ability to discover and communicate with other VMs seamlessly.

## 4 IPv6 Address Configuration and Management

In this section, we will explore IPv6 address configuration and management using the global prefix 2000 :100 :100 : :/48, which belongs to the Global Unicast Address (GUA) range. We will implement two addressing methods : first, manual configuration on machines acting as routers, and second, automatic configuration on other workstations. This will demonstrate how to set up globally routable IPv6 addresses in a network.

```
+-----------------+          +-----------------+
|    Router PC1   | -------- |    Router PC3   |
| 2000:100:100::1 |          | 2000:100:200::1 |
| (Manual Config) |          | (Manual Config) |
+-----------------+          +-----------------+
        |                            |
+-----------------+          +-----------------+
| Workstation PC2 |          | Workstation PC4 |
|  Auto-config    |          |  Auto-config    |
+-----------------+          +-----------------+
```

### 4.1 Configuring Bridge Mode for Router Interconnection

To enable interconnection between virtual machines located on different physical computers, we need to configure the router's 'enp0s3' interface to operate in bridge mode.

1. Stop the Virtual Machine
2. Click on "Settings" for the selected VM, and navigate to the "Network" tab.
3. Choose bridge adapter for "enp0s3". After setting the network adapter to bridged mode, click "OK" to save the settings.
4. You can verify the configuration by checking the network interface's status within the VM :
   ```
   ip addr show enp0s3
   ```
   To further ensure that the setup is correct, test connectivity to the internet :
   ```
   ping -c 4 google.com
   ```

### 4.2 IPv6 Address Configuration

The network consists of two routers (PC1 and PC3) and two workstations (PC2 and PC4). The routers use a direct connection for inter-router communication, while the workstations obtain their IPv6 addresses through Router Advertisement (RA).

| Device | Interface | Purpose | IPv6 Address | Configuration Type |
|---|---|---|---|---|
| PC1 (Router) | `enp0s8` | Local network (to PC2) | `2000:100:100::1/64` | Manual |
| PC1 (Router) | `enp0s3` | Router interconnection (to PC3) | `2000:100:300::1/64` | Manual |
| PC3 (Router) | `enp0s8` | Local network (to PC4) | `2000:100:200::1/64` | Manual |
| PC3 (Router) | `enp0s3` | Router interconnection (to PC1) | `2000:100:400::1/64` | Manual |
| PC2 (Workstation) | `enp0s8` | Auto-config (to PC1) | Assigned via RA | Automatic |
| PC4 (Workstation) | `enp0s8` | Auto-config (to PC3) | Assigned via RA | Automatic |

1. Manuel Configuration for Routers :
   Configure static IPv6 addresses and enable forwarding on routers PC1 and PC3 :

```
# Configure static IPv6 addresses of PC1
sudo ip -6 addr add 2000:100:100::1/64 dev enp0s8
sudo ip -6 addr add 2000:100:300::1/64 dev enp0s3

# Enable and persist IPv6 forwarding on PC1
sudo sysctl -w net.ipv6.conf.all.forwarding=1
echo "net.ipv6.conf.all.forwarding=1" | sudo tee -a /etc/sysctl.conf

# Configure static IPv6 addresses of PC3
sudo ip -6 addr add 2000:100:200::1/64 dev enp0s8
sudo ip -6 addr add 2000:100:400::1/64 dev enp0s3

# Enable and persist IPv6 forwarding on PC3
sudo sysctl -w net.ipv6.conf.all.forwarding=1
echo "net.ipv6.conf.all.forwarding=1" | sudo tee -a /etc/sysctl.conf
```

2. Auto-configuration for Workstations : To enable workstations PC2 and PC4 to automatically obtain their IPv6 addresses, routers PC1 and PC3 must be configured to send Router Advertisements (RA). The ‘radvd‘ daemon is used on Linux systems to manage these advertisements. The following steps detail the configuration process :

   1. **Install radvd :** First, ensure that ‘radvd‘ is installed on PC1 and PC3. If not already installed, you can install it using the package manager :

```
sudo apt-get update
sudo apt-get install radvd
```

   This step prepares the routers to handle the sending of IPv6 router advertisements.

   2. **Configure radvd :** Create or edit the ‘radvd.conf‘ file typically located at ‘/etc/radvd.conf‘ to specify the IPv6 prefixes to be advertised and other necessary parameters. Here is an example configuration for both routers :

```
# Configuration for PC1
sudo nano /etc/radvd.conf
interface enp0s8
{
    AdvSendAdvert on;
    prefix 2000:100:100::/64
    {
        AdvOnLink on;
        AdvAutonomous on;
    };
};

# Configuration for PC3
```

```
interface enp0s8
{
    AdvSendAdvert on;
    prefix 2000:100:200::/64
    {
        AdvOnLink on;
        AdvAutonomous on;
    };
};
```

These settings enable the routers to advertise necessary network parameters automatically.

3. **Enable and start radvd :** Once the ‘radvd‘ configuration is in place, the service must be enabled and started to begin the advertisement process :

```
sudo systemctl enable radvd
sudo systemctl start radvd
```

This command ensures that ‘radvd‘ will start automatically on boot and begins its operation immediately.

4. **Verify RA Operation :** Verify that ‘radvd‘ is sending advertisements :

```
sudo tcpdump -i enp0s8 icmp6
```

Look for packets specifically labeled as ”router advertisement” to verify that the router is correctly advertising the network configuration.

5. **Network Accessibility Verification :** Once router advertisements are confirmed, test the network accessibility from the workstations to ensure they can reach other network segments and the internet :

```
ping6 [IPv6 address of another device] # 2000:100:100::1
traceroute6 [IPv6 address of another device]
```

**Questions** :
— What steps are involved in manually configuring IPv6 addresses on routers, and why is this necessary ?
— Describe the process and importance of setting up ‘radvd‘ for automatic IPv6 address configuration in networks.

## 5 DNS configuration and management

In this section, we will configure a DNS server on PC1 to manage both forward and reverse DNS lookups for the network domain `societe-ngi.com`. This setup will provide the foundation for network services relying on domain name resolutions within our IPv6 infrastructure.

1. **Installation of DNS Server Software :** The first step involves installing BIND9, a widely used DNS server software that supports IPv6, on our designated server PCs.

```
sudo apt-get update
sudo apt-get install bind9 bind9utils bind9-doc
```

2. **Configuration of DNS Zones :** After installation, we will configure the primary DNS zones for our domain, including both forward and reverse lookup zones. This will be done by editing the BIND9 configuration files.

```
# Example configuration for forward lookup zone
sudo nano /etc/bind/named.conf.local
zone "societe-ngi.com" {
    type master;
    file "/etc/bind/db.societe-ngi.com";
};

# For IPv6 reverse lookup zones, you need to define the zone based on your
   network's IPv6 address range. If your network uses the '2001:DB8::/48'
   prefix, the reverse zone might be specified as follows:

zone "8.b.d.0.1.0.0.2.ip6.arpa" {
    type master;
    file "/etc/bind/db.8.b.d.0.1.0.0.2.ip6.arpa";
};
```

3. Navigate to the directory where BIND stores its zone files and create a new zone file for 'societe-ngi.com' :

```
cd /etc/bind
sudo cp db.local db.societe-ngi.com
sudo nano db.societe-ngi.com
```

4. **Setting Up Zone Files :** Edit the zone file to include DNS records for your domain. Here's an example of what entries might look like :

```
# Forward Zone File
$TTL    86400
@   IN  SOA ns1.societe-ngi.com. admin.societe-ngi.com. (
        2025010101 ; Serial
        3600       ; Refresh
        1800       ; Retry
        604800     ; Expire
        86400 )    ; Negative Cache TTL
@   IN  NS  ns1.societe-ngi.com.
ns1 IN  AAAA    2001:DB8::1


# Reverse Zone File
$TTL    86400
@   IN  SOA ns1.societe-ngi.com. admin.example.com. (
        2025102501 ; Serial
        3600       ; Refresh
        1800       ; Retry
        604800     ; Expire
        86400 )    ; Negative Cache TTL
1.0.0.0 IN  PTR ns1.societe-ngi.com.
```

5. **Starting the DNS Server and Testing :** With the configuration in place, we start the BIND9 service and test our setup to ensure that DNS resolution is functioning correctly.

```
# Start the BIND9 service and enable it to start at boot
sudo systemctl start bind9
sudo systemctl enable bind9

# Test the DNS configuration by querying the DNS server:
dig @2001:DB8::1 ns1.societe-ngi.com AAAA
dig -x @2001:DB8::1 2001:DB8::1
```

**Questions :**
— What is the purpose of DNS in a network ?
— Take a screenshot that you successfully did the DNS configuration.
— Given the network setup in the tutorial, if `societe-ngi.com` has an IPv6 address of `2001:DB8::1`, what would the PTR record look like in the reverse DNS zone file ?

— How would you verify that your DNS server is correctly resolving names to addresses using the dig command ?

## 6 Tunneling in Ipv6

This section details the implementation of an IPv6 tunnel over an existing IPv4 network. Assuming that the ISP router between Router PC1 and Router PC3 only supports IPv4, we need to create an IPv6 tunnel to ensure seamless IPv6 connectivity. This experiment is feasible using two virtual machines (VMs). An advanced setup can incorporate the topology discussed in section 4. .

### 6.1 Configuration Overview

The network setup comprises two virtual machines acting as routers :

— VM1 and VM2 are configured with both IPv4 for tunnel setup and IPv6 for internal and external communication.

— (Optional) Additional nodes (if any) can obtain their IPv6 addresses via Router Advertisement from VM1 and VM2.

We use the enp0s3 interface on both VM1 and VM2 as the tunneling interface.

| Device | Interface | Purpose | IPv6 Address | IPV4 Address |
|---|---|---|---|---|
| VM1 (Router) | `enp0s3` | Tunnel endpoint (to VM2) | `2001:db8:1::1/64` | `10.0.2.15/24` |
| VM2 (Router) | `enp0s3` | Tunnel endpoint (to VM1) | `2001:db8:2::1/64` | Manual |

### 6.2 Step-by-Step Tunnel Configuration

1. Assign IPv4 Addresses : Ensure each VM has a unique IPv4 address on the enp0s3 interface used for the tunnel.

```
# on VM1
sudo ip addr add 10.0.2.15/24 dev enp0s3
sudo ip link set enp0s3 up

# on VM2
sudo ip addr add 10.0.2.16/24 dev enp0s3
sudo ip link set enp0s3 up
```

2. **Create Tunnel Interfaces :** Establish a tunnel interface on both VM1 and VM2.

```
# On VM1
sudo ip tunnel add tun0 mode sit remote 10.0.2.16 local 10.0.2.15 ttl 255
sudo ip link set tun0 up
sudo ip -6 addr add 2001:db8:1::1/64 dev tun0

# On VM2
sudo ip tunnel add tun0 mode sit remote 10.0.2.15 local 10.0.2.16 ttl 255
sudo ip link set tun0 up
sudo ip -6 addr add 2001:db8:2::1/64 dev tun0
```

3. **Configure IPv6 Routing :** Add routing rules to ensure IPv6 packets travel through the tunnel.

```
# On VM1
sudo ip -6 route add 2001:db8:2::/64 dev tun0

# On VM2
sudo ip -6 route add 2001:db8:1::/64 dev tun0
```

4. **Verification :** Check connectivity using IPv6 ping.

```
# From VM1 to VM2
ping6 2001:db8:2::1

# From VM2 to VM1
ping6 2001:db8:1::1
```

**Questions :**
— What is the purpose of tunneling IPv6 traffic over an IPv4 network ?
— If IPv6 traffic is not being properly routed through the tunnel, what troubleshooting steps would you take to diagnose and resolve the issue ?

## 7 Use of application services (Bonus)

While TCP and UDP transport protocols are designed to work independently of the underlying IP protocol (IPv4 or IPv6), applications themselves often require modifications to handle IPv6 addresses properly. This is due to the differences in address handling between IPv4 and IPv6.

The goal of this experiment is to explore and verify how well applications support IPv6, particularly applications that are fundamental to internet infrastructure, such as web servers, mail servers, browsers, and mail clients. These applications need to handle IPv6 addresses efficiently to function properly in increasingly IPv6-dominated networks.

**Prerequirements** :
— Two virtual machines (VMs) with IPv6 network configuration.
— Operating system : Any recent Linux distribution (e.g., Ubuntu 20.04).
— Software : Apache web server, Firefox browser.

### 7.1 Server Configuration

1. Install Apache on the Server VM Begin by installing the Apache HTTP server on the first VM, which will act as the web server.

```
sudo apt update
sudo apt install apache2
```

2. Configure Apache to Listen on IPv6 Modify the Apache configuration to enable listening on an IPv6 address.

```
# Edit the ports configuration file
sudo nano /etc/apache2/ports.conf

# Add or ensure the following line is present
Listen [::]:80
```

Restart Apache to apply the changes :

```
sudo systemctl restart apache2
```

### 7.2 Client Configuration

1. Install Firefox on the Client VM Install Firefox on the second VM, which will be used to access the Apache server.

```
sudo apt update
sudo apt install firefox
```

2. Using IPv6 Address Open Firefox and navigate to the IPv6 address of the server enclosed in square brackets. If the DNS setup is not preferred or available, directly use the IPv6 address.

```
        http://[2001:db8::1]
```

3. Using DNS Name If a DNS name is configured, you can also access the server using the DNS name :

```
        http://www.example.com
```

## 7.3 Testing and Validation

Verify that the Apache web page is accessible from the client VM. Check the server's access logs to confirm that the access was logged correctly over IPv6.

```
cat /var/log/apache2/access.log
```

**Questions :**

— What modifications are generally required for applications to support IPv6, and why are these necessary ?
— Detail the steps to configure an Apache server to listen on an IPv6 address. Why is it important to specify the Listen [ : :] :80 directive in the Apache configuration file ?
— Suppose you cannot access the Apache server using the IPv6 address from the client VM. List potential issues that could cause this problem and suggest troubleshooting steps.