Lab
2

Single Subnet

What you will learn in this lab:

- How to capture and filter network traffic
- How to configure a network interface for IPv4 and IPv6 addresses
- How IPv4 and IPv6 addresses co-exist
- How to find the MAC addresses associated for an IPv4 or IPv6 address
- How to access statistical information on network interfaces

Updated: October 2021

Note to ECE461 students (Oct. 2021)

The PCs seem to have many background processes running that generate extraneous traffic.
We plan to have a revised install Ubuntu later on that fixes this issue.
For now, you can disable the traffic with the following commands.

```
$ sudo service isisd stop
$ sudo service ospfd stop
$ sudo service pimd stop
$ sudo service zebrad stop
```

For unknown reasons, the `isisd` process restarts automatically after it is stopped.

Table of Content

Study Material for Lab 2

1. **Linux command *ip*:** Review information on the Linux commands *ip.* Information is available at:

   - https://baturin.org/docs/iproute2/

   - https://www.cyberciti.biz/faq/linux-ip-command-examples-usage-syntax/

   - https://access.redhat.com/sites/default/files/attachments/rh_ip_command_cheatsheet_1214_jcs_print.pdf

   Use these resources to determine how to use the *ip* command for the following tasks:

   a. Configure IPv4 and IPv6 addresses for a network interface
   b. List the configuration of all network interfaces
   c. Display the neighbor cache
   d. Delete all entries in the neighbor cache

2. **Wireshark**: There are numerous websites and videos that explain the operation of *Wireshark*. Find a few of these sources and learn about capture filters and display filters for *Wireshark.*

3. **EUI-64 address format:** Read up about the EUI-64 address format, which generates a 64-bit identifier from a 48-bit MAC address. You find an explanation at

   - https://community.cisco.com/t5/networking-documents/understanding-ipv6-eui-64-bit-address/ta-p/3116953

4. **Link-local IPv6 unicast address:** Read up about the types of IPv6 addresses and their format. Resources for information are:

   - https://www.tutorialspoint.com/ipv6/ipv6_address_types.htm 
   - https://en.wikipedia.org/wiki/IPv6_address 
   - https://www.networkingwithfish.com/understanding-ipv6-what-is-solicited-node-multicast-part-4-of-7/ 

   Use these resources to learn about the following types of IPv6 addresses.

   a. Link-local unicast address
   b. Unique-local unicast address
   c. Global scope unicast address
   d. Solicited-node multicast address

Prelab 2

1. Write the syntax for an *ip* command that sets the IPv4 address of the interface *eth0* to *142.150.235.21* with broadcast address *142.150.235.63*.

2. Describe what the command `ip a' does.

3. Write the syntax for an *ip* command that adds an entry to the neighbor cache with the following information: IP address *142.150.235.21*, MAC address *2:3:4:5:6:7,* interface *eth0*.

4. Provide a command to display the entries of the neighbor cache.

5. Provide a command that delete all entries from the neighbor cache that were created by the protocol ARP.

6. Create an EUI-64 address from the MAC address *ea:db:6a:65:15:0b*.

7. Create a link-local IPv6 address for a network interface with MAC address *ea:db:6a:65:15:0b*.

8. Create the solicited-node multicast address for the link-local address from Question 7.

Lab 2 - Single Subnet

In Lab 2 you become acquainted with IP configuration issues for a group of PCs, all connected to the same Ethernet switch, that form an IP subnetwork or subnet.

Create a single Ethernet segment consisting of four PCs (*PC1* - *PC4*) by connecting the PCs to an Ethernet hub (Hub1) as shown in Figure 2.1. The IP addresses for the PCs are as shown in Table 2.1.

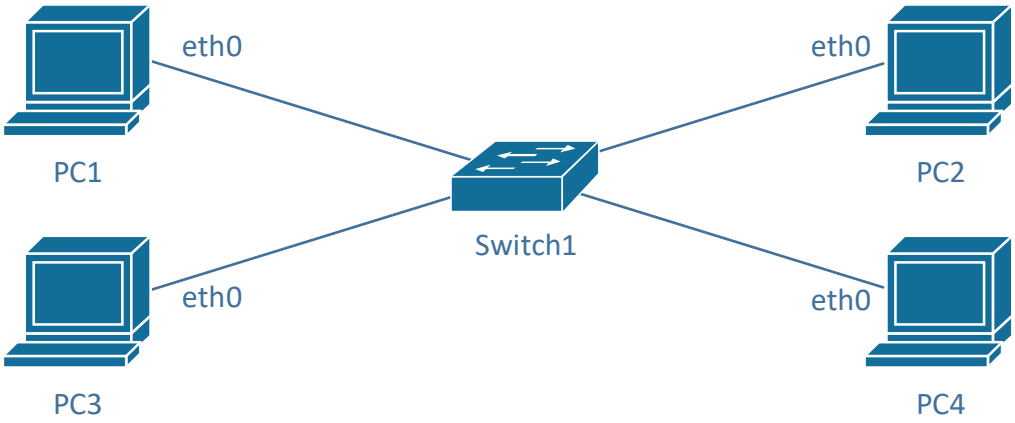


Figure 2.1. Configuration for Lab 2.

Table 2.1. IP Addresses for Figure 2.1.

PCs	IP Address of *eth0*
PC1	10.0.1.11 / 24
PC2	10.0.1.22 / 24
PC3	10.0.1.33 / 24
PC4	10.0.1.44 / 24

Part 1. IPv4 Configuration

In this part you set up the network topology and configure IPv4 addresses for Lab 2.

Exercise 1-a. Setup of topology

Create a network topology as shown in Figure 2.1, where four PCs (*PC1* - *PC4*) are connected to an Ethernet switch. Refer to the instructions in Lab 1 for the steps involved in configuring a network topology.

Reboot all PCs.

Exercise 1-b. Configuration of IP addresses

Configure the IP addresses of the *eth0* interfaces of the PCs as shown in Table 2.1.

Step 1: For each PC, open a console and check the IP configuration of interface *eth0*. For *PC1*, this is done by typing the command

```
PC1:~# ip addr show eth0
```

Verify that no no IPv4 addresses are configured. If IP addresses of the PCs are already configured and are different from Table 2.1, delete each existing IPv4 address with the command

```
PC1:~# sudo ip addr del  <IPv4 address>/<prefixlength> dev eth0
```

Here, `<IPv4 address>` is the existing IPv4 address and `<prefixlength>` is the length of the network prefix.

Step 2: Configure the addresses on all PCs. On *PC1*, the command is

```
PC1:~# sudo ip addr add 10.0.1.11/24 dev eth0
```

Step 3: If the IP addresses are configured correctly, all machines should be able to ping each other. Verify this by pinging all other PCs from *PC1*.

```
PC1:~# ping -c2 10.0.1.22
PC1:~# ping -c2 10.0.1.33
PC1:~# ping -c2 10.0.1.44
```

Adding a broadcast address to an interface

The command sudo `ip addr add` does not automatically configure the broadcast address of an interface (even though the CIDR prefix length implies it). To add a broadcast address, in addition to the IP address, you need to add `` `brd +' ``, as shown here

```
sudo ip addr add 10.0.1.14/24 brd + dev eth0
```

Since broadcast addresses other than the local address 255.255.255.255 are rarely used, omitting the broadcast address from the IP configuration has generally no negative consequences.

Part 2. Address Resolution Protocol (ARP)

This part of this lab explores the operation of the Address Resolution Protocol (ARP) that resolves a MAC address for a given IPv4 address. The results of the address resolution are stored in a neighbor cache, also referred to as the ARP cache. The lab exercises use the Linux command `ip neighbor` for manipulating the contents of the neighbor cache, which, in the context of IPv4, is also called ARP cache. The relevant applications of the `ip neighbor` command are listed below.

Manipulating the neighbor cache:

`ip neigh`
`ip -s neigh`
Display the content of the neighbor cache. The "-s" option displays additional details.

`sudo ip neigh del <ip-address> dev <interface>`
Deletes an entry for IPv4 address *<ip-address>* for interface *<interface>* from the neighbor cache.
Example:
`sudo ip neigh del 10.0.1.12 dev eth0`

`sudo ip neigh add <ip-address> lladdr <mac-address> dev <interface>`
Adds a permanent entry to the neighbor cache, which associates IPv4 address *<ip-address>* with MAC address *<mac-address>* on interface *<interface>*.
Example:
`sudo ip neigh add 10.0.1.12 lladdr 82:b9:d4:40:64:01 dev eth0`

`sudo ip neigh flush dev <interface>`
Delete all non-permanent entries for interface `<interface>` from the neighbor cache.
Example:
`sudo ip neigh flush dev eth0`

`sudo ip neigh flush all`
Delete all non-permanent entries for all interfaces from the neighbor cache

Exercise 2-a. A simple experiment with ARP

The objective of this exercise is to observe when ARP messages are sent and how ARP updates the neighbor cache.

Step 1: On *PC1*, check the content of the neighbor cache with

```
PC1:~# ip -s neigh
```

and then delete all entries with

```
PC1:~# sudo ip neigh flush all
```

Step 2: Also delete the neighbor cache on *PC2*.

Step 3: Start *Wireshark* for the traffic between *PC1* (*eth0*) and the Ethernet switch.

> **Note to ECE461 students (Oct. 2021)**
>
> To limit the displayed packets to ARP and IPv4 packets, set a display filter to `arp or icmp`.

Step 4: Issue a ping command from *PC1* to *PC2* by

```
PC1:~# ping -c2 10.0.1.22
```

Observe the ARP packets and ICMP packets that are captured by *Wireshark*.

Take a snapshot of the pane in *Wireshark* that shows the list of packets (see Lab 1, Part 9). Make sure that the snapshot fully shows all columns in the pane.

View the neighbor cache again with the command shown in Step 1.

- Explain the observed order of ARP and ICMP packets.
- Take a close look at the destination MAC addresses in the MAC headers of ARP packets. How many broadcast packets do you observe?
- Inspect the content of the Sender/Target MAC and IP addresses in the ARP packets. Compare the content of these fields in an ARP Request and an ARP Reply.

Step 5: Display the neighbor cache again at *PC1* with the command

```
PC1:~# ip -s neigh
```

How has the neighbor cache changed since you displayed it last in Step 2?

Step 6: Repeat Step 2, and observe the ARP and ICMP packets in the *Wireshark* display. The order of ARP and ICMP packets should now be different.

- Explain the different order of packets.
- Take a close look at the destination MAC addresses in the MAC headers of ARP packets. How many broadcast packets do you observe? Explain!

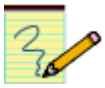
Take a snapshot of the pane in *Wireshark* that shows the order of packets.

Step 7: Gather evidence (as screen snapshots or saved data) to support **one** of the following behaviors of Linux:

a. When an ARP Reply is received, the neighbor cache entry is marked as "REACHABLE". After 30 seconds of inactivity, the entry is marked as "STALE".
b. When an IP datagram is sent to destination, whose neighbor cache entry is "STALE", the entry is marked as "DELAY". After several seconds, an ARP Request is sent for this entry.

Step 8: Stop the traffic capture and save the details of the captured traffic as a plain text file.

Lab Questions/Report



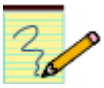
1. Include the screen snapshots from Step 4 and Step 6.
2. Answer the questions from Step 4. Use the saved file (Step 8) if you need to look at details of a captured packet.
   a. Explain the observed order of ARP and ICMP packets.
   b. Which packets are sent as broadcast?
   c. Take the ARP Request from *PC1* and the corresponding ARP Reply. Compare the content of the Sender/Target MAC and IP addresses in the ARP packets.
3. Answer the questions from Step 6.
   a. Explain the different order of packets in Step 6 (compared to Step 4).
   b. Which packets are sent as broadcast? Explain the observation.

Exercise 2-b. Matching IP addresses and MAC addresses

Identify the MAC addresses of all the interfaces connected to the Ethernet switch and enter them in Table 2.2. You can obtain the MAC addresses from the ARP cache of a PC by issuing a ping command from that host to every other host on the network. Alternatively, you can obtain the MAC addresses from the output of the `ip addr show eth0` command in the console window of each PC. Save this table.

Table 2.2. IP and MAC addresses.

PCs	IP Address of *eth0*	MAC address of *eth0*
PC1	*10.0.1.11 / 24*	
PC2	*10.0.1.22 / 24*	
PC3	*10.0.1.33 / 24*	
PC4	*10.0.1.44 / 24*	



Lab Questions/Report

1. Include the created table in your report.

Exercise 2-c. ARP requests for a non-existing address

Observe what happens when an ARP request is issued for an IP address that does not exist in the local subnet.

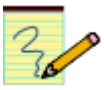
Step 1: Start a traffic capture for the traffic from and to interface *eth0* of *PC1*.

Step 2: Issue a ping command from *PC1* to *10.0.1.220*. (Note that this address does not exist in this network configuration.)

```
PC1:~# ping -c5 10.0.1.220
```

Step 3: Take a snapshot of the pane in *Wireshark* that shows the order of packets.

Step 4: On *PC1* display the ARP cache and observe the entry (entries) in the cache.



Lab Questions/Report

1. Use the snapshot from Step 3 to explain how ARP handles non-existing addresses.

Part 3. More on IPv4 address configuration

You already have encountered the Linux command for configuring an IPv4 address. The command *ip* can also be used to activate and deactivate network interfaces, as well as for querying the status of network interfaces. Below is a list of relevant commands.

iproute2 commands
For IP address configurations on Linux, the labs use *iproute2*, a collection of utilities for managing the network configuration in the Linux kernel, including routing tables, network interfaces, IP addresses, and more. The *iproute2* utilities replaces older – possibly more familiar – commands such as *ifconfig*, *netstat*, and *arp*, which are destined to become obsolete. The *ip* command, which you have encountered earlier, is also part of *iproute2*.

Here are examples of important *ip* commands for configuring network interfaces. These commands are needed in all future labs.

Examples of `ip` commands for configuring network interfaces:

`sudo ip addr add` *`10.0.1.11/24`* **`dev`** *`eth0`*
`sudo ip addr add` *`10.0.1.11/24`* **`brd + dev`** *`eth0`*
Add IP address *10.0.1.11* with CIDR prefix length 24 to network interface *eth0*. The (optional) "`brd +`" parameter also sets the broadcast address for this interface. Note that you can configure multiple IP addresses for a single interface.

`sudo ip addr del` *`10.0.1.11/24`* **`dev`** *`eth0`*
Remove the IP address `10.0.1.11/24` from the *`eth0`* interface.

`sudo ip addr` ***`flush`*** **`dev`** ***`eth0`***
Remove (flush) all IP addresses from the *`eth0`* interface.
We recommend to not use this command for the labs, since it removes the link-local IPv6 addresses, which are needed in later parts of the lab.

`sudo ip link set dev` ***`eth0`*** **`up`**
`sudo ip link set dev` ***`eth0`*** **`down`**
Activate/De-activate network interface *`eth0`*. On the PCs in the lab, all network interfaces are activated as part of the boot-up procedure. The activation of an interface is independent of the configuration of an IP address. In particular, you can configure an IP address for a de-activated interface.

`ip addr show eth`
Show the IP configuration of all network interfaces. There are various other versions for this command (`ip address show`, `ip addr show`, `ip addr s`, `ip a`).

`ip addr show eth0`
Show the IP configuration of network interface *`eth0`*.

Exercise 3-a. Duplicate IPv4 addresses

You will change the IPv4 addresses of *PC3* and *PC4* so that it is equal to that *PC2*. Then you observe the effects of having multiple hosts with the same IP address in a network.

Step 1: Verify that *PC1* and *PC2* have IPv4 addresses as listed in Table 2.1.

Step 2: On *PC3* and *PC4*, change the IPv4 address of interface *eth0* of *PC4* to *10.0.1.22/24*.
Note: To change the IPv4 address, you need to first delete the current address.

Step 3: Start a traffic capture for the traffic from and to the *eth0* interface of *PC1*.

Step 4: On *PC1*, flush the content of the neighbor cache at *PC1* with

```
PC1% sudo ip neigh flush all
```

and then issue a *ping* command to the duplicate IP address 10.0.1.12, by typing

```
PC1% ping -c10 10.0.1.22
```

Take a snapshot of the top pane in *Wireshark* to show all packets that were captured in Step 4. Resize the pane if necessary.

- Observe the captured ARP packets and determine which of the PCs reply to the ARP *Request* of *PC1*, and in which sequence.
- Compare the MAC addresses in the Ethernet headers of the captured *ICMP Reply* messages with Table 2.2 to determine which PC sends *ICMP Echo Reply* messages to *PC1*. Record your finding.

Step 5: Run the command

```
PC1% ip neigh show
```

to determine which PC in the neighbor cache of *PC1* is associated with *10.0.1.22*. Reconcile the output with the captured traffic.

Take a screenshot of the output.

Step 6: Repeat Steps 4 and 5 two more times.

- Do the outcomes change? Record your finding.

Step 7: On the PC which sent the *ICMP Replies* in your most recent *ping*, delete the configured IPv4 address on *PC2*. Without flushing the neighbor cache at *PC1*, run another ping at *PC1* with

```
PC1% ping –c10 10.0.1.22
```

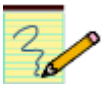
and run the command

```
PC1$ ip neigh show
```

- Explain the ARP traffic observed in this scenario. (Also remark when an ARP packet is expected but not seen).

- Take a snapshot of the top pane in *Wireshark* to show all packets that were captured. Resize the pane if necessary. It is also recommended that you save traffic captured by *Wireshark (*`File` → `Save as`*…),* in case you need details for the lab report.

Step 8: Stop the traffic capture and reset the IP addresses of *PC2*, *PC3*, and *PC4* to the original values given in Table 2.1.



Lab Questions/Report

1. Include the screenshots taken in Step 4 and Step 5, answer the questions in Step 4.
2. Report your findings from Step 6.
3. Include the screenshot from Step 7. Explain the ARP traffic observed in this scenario. (Also remark when an ARP packet is expected but not seen).

Exercise 3-b. Multiple IP addresses on the same network interface

Here, you learn that it is possible to assign multiple IPv4 addresses to a single network interface.

Step 1: Verify that the IP addresses of the PCs are as given in Table 2.1. Flush the neighbor cache at all PCs.

Step 2: Add the IP address *10.0.3.11/24* to interface *eth0* of *PC1*, and add the IP address *10.0.3.33/24* to interface *eth0* of *PC3*.

To confirm that the second IPv4 address has been configured type

```
PC1% ip addr
```

Step 3: On *PC1*, issue the following ping commands

```
PC1% ping -c2 10.0.1.33
PC1% ping -c2 10.0.3.33
```

and display the neighbor cache with

```
PC1% ip neigh show
```

Step 4: On *PC3*, display the neighbor cache on *PC3*.

- Convince yourself that *PC1* has cache entries for both IP addresses of *PC3*. Likewise, *PC3* has entries for both IP addresses of *PC1*.

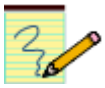
Step 5: Remove the IP address *10.0.3.33/24* from interface *eth0* of *PC3*. Display the neighbor cache at *PC3* and observe the neighbor cache is unchanged.

Step 6: On *PC1*, issue the ping command

```
PC1% ping -c2 10.0.3.33
```

- Try to explain why this ping fails, even though there is a neighbor entry for *10.0.3.33* in the neighbor cache of *PC1*.
- You may want to run *Wireshark* to determine where the ping fails.

Step 7: Take snapshots of the consoles of *PC1* and *PC3* which show the commands issued in Steps 3—7, and the output of the commands.



Lab Questions/Report

1. Include the screenshots taken in Step 7.
2. Provide your explanation why the ping in Step 6 fails, even though *PC1* has a neighbor cache entry for *10.0.3.33*, and *PC3* has a cache entry for *10.0.3.11*.
3. Can you think of advantages and/or disadvantages of using multiple IP addresses for a network interface?

Exercise 3-c. Loopback addresses in IPv4 and IPv6

A packet that is sent to the loopback address is delivered locally, that is, to the system that sent the packet. In IPv4, the standard loopback address is *127.0.0.1*. In fact, all addresses in the range *127.0.0.1—127.255.255.254* are loopback addresses. In IPv4, the loopback address is *::1/128*.

The loopback address is also associated with the host name *localhost*. The binding of the name localhost to the addresses *127.0.0.1* and ::1 is done in the file */etc/hosts*.

The purpose of this exercise is to observe that traffic sent to the loopback address does not create network traffic.

Step 1: Start a traffic capture for the traffic from and to *PC1*.

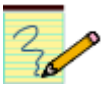
Step 2: On *PC1*, issue ping commands as follows

```
PC1% ping -c2 127.0.0.1
PC1% ping -c2 127.255.255.254
PC1% ping -c2 -4 localhost
PC1% ping -c2 -6 localhost
PC1% ping -c2 -6 ::1
```

Here, the options "-4" and "-6" force ping to use IPv4 and IPv6, respectively. Confirm that Wireshark did not capture packets for any of the commands.

Step 3: Take a snapshot of the output of the commands in Step 2.

Lab Questions/Report



1. Provide the screen snapshot from Step 3.

Part 4. Changing netmasks

In this part of the lab you test the effects of changing the netmask of a network configuration. In the table below, two hosts (*PC2* and *PC4*) have been assigned different network prefixes

Exercise 4-a. Changing netmasks

Table 2.4. IP addresses for Part 4.

PCs	IP Addresses of *eth0*
PC1	*10.0.64.130/**20***
PC2	*10.0.64.145/**24***
PC3	*10.0.64.71/**26***
PC4	*10.0.0.130/**16***

Step 1: Change the IPv4 addresses of the *eth0* interfaces of the PCs to the values shown in Table 2.4.

Note: To change the IPv4 address, you need to first delete the current address(es). To see the current IP addresses, type the command `ip addr show eth0'.

Step 2: Start a new *Wireshark* traffic capture for the traffic from and to *PC1*, and run the following *ping* commands

a) From *PC1* to *PC2*: `PC1:~$ ping -c2 10.0.64.145`
b) From *PC2* to *PC1*: `PC2:~$ ping -c2 10.0.64.130`
c) From *PC1* to *PC3*: `PC1:~$ ping -c2 10.0.64.71`
d) From *PC3* to *PC1*: `PC3:~$ ping -c2 10.0.64.130`
e) From *PC1* to *PC4*: `PC1:~$ ping -c2 10.0.0.130`
f) From *PC4* to *PC1*: `PC4:~$ ping -c2 10.0.64.130`

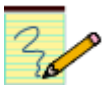
Determine whether the ping commands are successful or not. If the commands are not successful, observe the error messages on the console and the captured traffic determine the reason for the failed pings. Record your findings.

Step 3: On all PCs, take snapshots showing the output of the *ping* commands of (a)—(f) in Step 2.

Step 4: Take a snapshot for of the top pane of *Wireshark* for *PC1* traffic. The snapshot should show the list of ICMP and ARP packets from and to *PC1* in Step 2.

Note: For the lab report, you are asked to present the output and the packets separately for each *ping* command. For this, you can either crop the snapshot, or you can take several snapshots in Step 3 and Step 4.

Lab Questions/Report



1. For each ping command in Step 2, use screenshots (or parts of the screenshots) from Step 3 and Step 4 to show the output of the command. Also, briefly state why a ping command succeeds or fails.

Part 5. IPv6 Configuration

In this part, you work with IPv6 addresses. You will realize that steps of the configuration are overall similar to configuring IPv4 addresses.

Exercise 5-a. Link-local IPv6 addresses

The first thing to note about IPv6 configuration is that each IPv6 capable network interface has an automatically configured IPv6 address, called link-local address, which is created from the MAC address of the interface. We will first try to get these addresses to work.

Step 1: Display the link-local IPv6 address of the *eth0* interfaces of all PCs with the command `ip addr show eth0', and record the addresses in a table as shown in Table 2.5. Save the IPv6 addresses.

No link-local IPv6 address? Restore it.

Some commands, e.g., *sudo ip addr flush dev eth0*, remove the link-local IPv6 address of an interface. Rebooting the PC will restore the values. As an alternative, you can set the link-local address of interface *eth0* with the command

```
ip address add dev eth0 scope link <link-local address>/64
```

To restore the original address, set `<link-local address>` to the link-local address that has the interface identifier set to the EUI-64 address of interface *eth0*. However, any IPv6 address from the address block *FE80::/10*, e.g., *FE80::1*, will also work.

Table 2.5. Link-local IPv6 Addresses

PC	Link-local IPv6 Address (*eth0*)
PC1	
PC2	
PC3	
PC4	

Step 2: Start a traffic capture for the traffic from and to the *eth0* interface of *PC1*.

Note to ECE461 students (Oct. 2021)

In Parts 5 and 6, limit the displayed packets to IPv6 packets with a display filter set to `ivp6'.

Step 3: Ping *PC1* from *PC2* with the command

```
PC1$ ping6 -c2 <link-local address of PC2>
```

you will notice that it is not successful.

- Note the output of the command which shows the reason for the failure.

If you append a `%' and the name of the network interface where you want to the packets to be transmitted (on *PC1*), as in,

```
PC1$ ping6 -c2 <link-local address of PC2>%eth0
```

you observe that the remote PC will reply.

ping6
The command for pinging an IPv6 address is *ping6*. An alternative is *ping -6.*

Step 4: Stop the traffic capture.

- Did *Wireshark* capture any traffic other than *ICMPv6* packets?

Lab Questions/Report

1. Include the IPv6 link local addresses from Table 2.5. Explain how the addresses are related to the MAC addresses in Table 2.2.

Exercise 5-b. Configuring IPv6 addresses

The next step is to assign IPv6 addresses to the PCs. The addresses are drawn from the address block fd00::/8, which designates unique-local addresses (ULA). These addresses play the same role as the 10.0.0.0./8 address block in IPv4.

An IPv6 address has three parts: (1) a global routing prefix ID, (2) a subnet ID, and (3) an interface ID (64 bits).

We pick the following values:

- Global routing prefix ID = *fd01:2345:6789* (48 bits),
- Subnet ID = *0001* (16 bits),
- Interface ID = *1* (for *PC1*), *2* (for *PC2*), *3* (for *PC3*), *4* (for *PC4*).

This results in the IPv6 addresses for the PCs shown in Table 2.6. With this selection, the CIDR prefix length is 64.

Table 2.6. IPv6 Addresses for *eth0* network interfaces of the PCs.

PCs	IPv6 Addresses of Ethernet Interface *eth0*
PC1	fd01:2345:6789:1::1/64
PC2	fd01:2345:6789:1::2/64
PC3	fd01:2345:6789:1::3/64
PC4	fd01:2345:6789:1::4/64

The configuration of IPv6 addresses is done with the same command used for IPv4.

```
$ sudo ip addr add <ip6-address>/<prefix> dev <interface>
```

That is, to configure the IPv6 address of *PC1*, type

```
PC1$ sudo ip addr add fd01:2345:6789:1::1/64 dev eth0
```

Step 1: Start Wireshark to capture the traffic from and to the *eth0* interface of *PC1*.

Step 2: Configure the IPv6 addresses of all PCs as shown in Table 2.6. (There is no need to delete the IPv4 addresses.)

- Take a screen snapshot of the packets that are captured by *Wireshark*. Only show the top pane (packet list). Make sure you fully capture the source and destination addresses, as well as the *Info* field.

Step 3: Observe the captured packets when you set an IPv6 address. You will see three types of packets:

- **ICMPv6 Multicast Listener Report:** This is part of the *Multicast Listener Discovery Version 2* (MLDv2). When sending a listener report, a host indicates that it is interested in receiving messages sent to a multicast address.
- **ICMPv6 Neighbor Solicitation:** This is part of the *Neighbor Discovery Protocol (NDP),* which resolves IPv6 addresses to *MAC* addresses, similar to *ARP* in IPv4. You learn about *NDP* in Part 6 of this lab.
- **ICMPv6 Router Solicitation:** With this message, a host indicates that it is looking for an IP router. This is covered in Lab 3.

Without getting into the details of the observed packets, explore the source and destination IPv6 addresses of the observed packets:

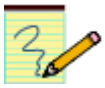
- The source addresses of the packets for router solicitation and multicast listener report have the link-local address that you recorded earlier in Table 2.5.
- The source address of the neighbor solicitation packets is set to all zeros.
- The destination addresses of all packets are multicast addresses. (Any IPv6 address that belongs to the prefix `ff00::/8` is a multicast addresse. An address that belongs to `ff02::/16` is a link local multicast addresses, that is, packets with this destination are never forwarded by a router.):
  - `ff02:2:` This address reaches all IPv6 routers.
  - `ff02::16:` This address reaches all IPv6 routers that interpret MLDv2 messages.
  - `ff02::1:ff00:1, ff02::1:ff00:2, ff02::1:ff00:3, ff02::1:ff00:4:` These are *solicited node multicast addresses*. They are constructed by replacing the last three bytes of the address ff02::1:ff00:0 with the last three bytes of an IPv6 address from Table 2.6. When an IPv6 address is added to an interface of a host, the host will listen to multicast IP addresses sent to the corresponding solicited node address.

Step 4: Use the ping6 command to make sure that each PC can exchange messages with every other PC. The command to send a ping from *PC1* to *PC2* is

```
PC1$ ping6 -c2 fd01:2345:6789:1::2
```

Step 5: Stop the traffic capture.

Lab Questions/Report



1. Include the snapshot from Step 2. Describe the source and destination addresses in the IPv6 headers of the saved packets.

Part 6. Neighbor Discovery Protocol (NDP)

In IPv6, the task of resolving the MAC address for a given IP address is performed by the Neighbor Discovery protocol (NDP). The results of the address resolution are stored in the neighbor cache, the same cache that stores results of the ARP protocol. The commands for displaying and manipulating the neighbor cache were discussed in Part 3.

Exercise 6-a. Address Resolution with NDP

Step 1: Verify that the topology of the network is as shown in Figure 2.1 and that the IP configuration of the PCs is as given in Tables 2.1 and 2.6. Use the command `ip addr show eth0' to view the current address configuration on a PC.

Step 2: On *PC1*, delete all entries in the neighbor cache with

```
PC1$ sudo ip neigh flush all
```

Do the same on *PC2*.

Step 3: Start a traffic capture to observe the traffic between *PC1* and *PC2*.

Step 4: Look up the link-local IPv6 address of the *eth0* interface at *PC2*. Then, issue a *ping* from *PC1* to the link-local address of *PC2* with the command

```
PC1$ ping6 -c2 <link-local address of PC2>%eth0
```

- Display the neighbor caches at *PC1* and *PC2* and take snapshots.
- Take a snapshot of the top pane in Wireshark to show all packets that were captured after issuing the ping. Resize the pane if necessary.
- Inspect the source and destination IP addresses in the Neighbor Solicitation/Advertisement messages. Which messages are sent as unicast and which messages are sent as multicast? For multicast destination addresses, identify the type of address.
- Inspect the fields of the ICMPv6 Neighbor Solicitation/Advertisement messages. Compare them to the fields of ARP Request/Reply messages.
- Inspect the relative order of Neighbor Solicitation/Advertisement messages and the Echo Request/Reply messages. Compare this order to the relative order of ARP messages and Echo Request/Reply messages in Part 3. Is the order identical, similar, or different?

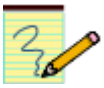
Step 5: Delete all entries of the neighbor cache at both *PC1* and *PC2*. Then, issue a ping from *PC1* to the configured IPv6 address on *PC2* with

```
PC1$ ping6 -c2 fd01:2345:6789:1::2
```

- Display the neighbor caches at *PC1* and *PC2* and take snapshots.
- Take a snapshot of the top pane in *Wireshark* to show all packets that were captured. Resize the pane if necessary.
- Compare the captured packet list to that of Step 4.

Step 6: Stop the traffic capture and save the details of the IPv6 packets as a plain text file.

Lab Questions/Report



1. Provide the snapshots taken in Step 4.
2. Characterize the source and destination IP addresses used in the Neighbor Solicitation and Neighbor Advertisement messages.
3. Compare the snapshot of the packet list with the snapshot of the ARP exchange in Step 4 of Exercise 2-a. Explain differences and commonalities between the NDP exchange and the ARP exchange.
4. Provide the snapshots taken in Step 5.
5. Compare the captured packet list from Step 5 to that of Step 4. Note the additional NDP messages that are exchanged.
6. Provide the message fields of
   - one pair of corresponding ARP Request and ARP Reply packets from the saved data in Exercise 2-a, and
   - one pair of corresponding Neighbor Solicitation and Neighbor Advertisement packets in this exercise.

   Use these packets to point out non-trivial differences between the formats of ARP and NDP packets (You should be able to find five such differences).

Exercise 6-b. Duplicate Address Detection in IPv6

In this exercise, you find an explanation why *PC1* was sending out a Neighbor Solicitation when you configured it with an IPv6 address. This message is used to discover whether an IP address is already used by another network interface. IPv6 always sends a Neighbor Solicitation for its own IP address when an IP address is manually configured. If there is a reply (Neighbor Advertisement) to the solicitation, the IP address already exists and will not be configured on the current node. This process is referred to as *Duplicate Address Detection (DAD).*

Step 1: Delete all entries in the neighbor caches of *PC1* and *PC2*.

Step 2: Open a console window on *PC1*. Display the IP configuration at the *eth0* interface of *PC1* with the command

```
PC1$ ip addr show eth0
```

Direct your attention to the displayed line for the manually configured IPv6 address.

Take a screenshot of the output.

Step 3: Start a traffic capture to observe the traffic between *PC1* and *PC2*.

Step 4: At *PC1*, remove the existing IP address and then configure the IP address of *PC2*, by typing

```
PC1$ sudo ip addr del fd01:2345:6789:1::1/64 dev eth0
PC1$ sudo ip addr add fd01:2345:6789:1::2/64 dev eth0
```

Inspect the NDP messages that are captured by Wireshark. The first message – sent by *PC1* –is a Neighbor Solicitation for the newly configured IPv6 address sent by *PC1*. Since *PC2* already has this IP address, it replies with a Neighbor Advertisement. This advertisement is interpreted by *PC1* to indicate that the IP address is already allocated.

Step 5: Again, display the IP configuration at the *eth0* interface of *PC1* with the command

```
PC1$ ip addr show eth0
```

Direct your attention to the displayed line for the newly configured IPv6 address. Note the output "*dadfailed tentative*" at the end of the line. This indicates that the configuration of the IP address has failed.

Take another screenshot of the output of the command.

Step 6: Perform a set of ping commands. For each execution, observe the traffic capture on *Wireshark* and the output on the console where the command is issued.

- On *PC2*:

```
PC2$ ping6 -c2 fd01:2345:6789:1::2
```

- On *PC3*:

```
PC3$ ping6 -c2 fd01:2345:6789:1::2
```

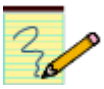
- On *PC1*:

```
PC1$ ping6 -c2 fd01:2345:6789:1::2
```

Record your observation. Interpret the outcomes with respect to avoiding duplicate IPv4 addresses on a subnet.

Step 7: Stop the traffic capture and save the captured IPv6 packets as a text file.

Lab Questions/Report



1. Provide the screen snapshots from Steps 2 and 5.
2. Describe your observations of the outcomes of the *ping* commands in Step 6 and explain the outcomes.