Computer Networks Lab

based on

Mastering Networks - An Internet Lab Manual
by Jörg Liebeherr and Magda Al Zarki

and on

Computer Networking - A Top-Down Approach
by James Kurose and Keith Ross

Adapted for
'Computernetwerken'
by Johan Bergs and Stephen Pauwels

Name 1
Name 2
Group groupID

March 30, 2023

Lab 3

IPv4 and IPv6

What you will learn in this lab:

- Differences between IPv4 and IPv6.
- What the effect is of netmasks.
- Differences in fragmentation.
- How Network Address Translation (NAT) works.
- How Dynamic Host Configuration Protocol (DHCP) works.

3.1 Lab 3

Lab 3 covers the differences between IPv4 and IPv6. This lab is divided in parts where each part compares an IPv4 feature to its IPv6 counterpart. You will need to build a dual stacked setup meaning every host runs both versions of IP.

Part 1. Netmasks

In this part of the lab you explore the effects of changing the netmask of an interface.

Exercise 1: Netmasks

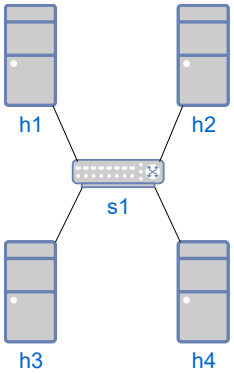

Figure 3.1: Network configuration for part 1.

Linux PC	**IPv6 address**	**IPv4 address**
h1	`fc00::1/48`	`10.0.0.1/8`
h2	`fc00::2/64`	`10.0.0.2/24`
h3	`fc00:0:0:1::3/48`	`10.0.255.3/16`
h4	`fc00:0:0:1::4/64`	`10.1.0.4/24`

Table 3.1: IP addresses and subnet masks

1. Write a python script that creates the topology shown in figure and table 3.1 and save it to `L3-1-1.py`.
2. Open an `xterm` on s1 and start a wireshark session on the `s1` interface. This will capture all packets that go through the switch (from all hosts).
3. Issue the `pingall` command in mininet.
4. Save the Wireshark trace to `L3-1-1.pcapng`. Save the output of the pingall command to `L3-1-1.ping.txt`.

Use the data captured with Wireshark in L3-1-1.pcapng and the output in L3-1-1.ping.txt to answer the questions. Support your answers with the saved Wireshark data.

L3-1-1 Use the output from the pingall and the Wireshark trace file (by referring to packet numbers) to explain what happened for each of the ping commands. Were all pings successful? For all pings (successful or not), explain why. /2

Part 2. Fragmentation in IPv4 and IPv6

Figure 3.2 shows the network setup for this part of the lab. There are two different subnets which are connected by r1. Note that IP addresses are given for both IP versions.

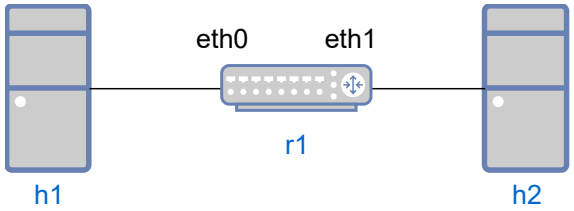

Figure 3.2: Network configuration for part 2.

hosts	Interface eth0	Interface eth1
h1	`fc00:0:0:1::1/64` `10.0.1.1/24`	N/A
r1	`fc00:0:0:1::10/64` `10.0.1.10/24`	`fc00:0:0:2::10/64` `10.0.2.10/24`
h2	`fc00:0:0:2::2/64` `10.0.2.2/24`	N/A

Table 3.2: IP addresses of all PCs for part 2.

In this part of the lab, you observe the differences in IP Fragmentation. In the previous lab, you have already observed how fragmentation has an effect of the overall throughput of a connection. In this lab, you will take a look of how fragmentation is done differently in IPv4 and IPv6.

Exercise 2: UDP and Fragmentation

In this exercise you observe IP fragmentation of User Datagram Protocol (UDP) traffic. In the following exercise, use `iperf` to generate UDP traffic between h1 and h3, across IP router r1.

This is one of the few exercises where you **must use iperf instead of iperf3**. Read up on the man page of iperf as the syntax is slightly different from iperf3!

If you want to redo this exercise, you should stop mininet and restart it, or you have to make sure the path MTU cache is empty on h1. You can clear the cache with `ip (-6) route flush cache`. You can see the cached path MTU with `ip (-6) route get [IP address]`

1. Write a python script to form the network as shown in Figure 3.2 and Table 3.2 and save it to `L3-2-1.py`.
2. Change the MTU on r1-eth1 to 1280 bytes.
3. Execute the following command on h1: `echo 1 > /proc/sys/net/ipv4/ip_no_pmtu_disc`
4. Start Wireshark on the eth0 and eth1 interfaces of r1 and start to capture traffic. Do not set any filters.
5. Use `iperf` to generate UDP traffic **over IPv6** between h1 and h2. Configure the `iperf` client to send 2 seconds of traffic to the server.

6. Repeat the previous step, but this time use IPv4.
7. Stop the traffic captures on r1 and save the Wireshark output to `L3-2-1.eth0.pcapng` and `L3-2-1.eth1.pcapng`.
8. Now, execute the following on h1: `echo 0 > /proc/sys/net/ipv4/ip_no_pmtu_disc`
9. Start Wireshark on the eth0 and eth1 interfaces of r1 and start to capture traffic. Do not set any filters.
10. Use `iperf` to generate UDP traffic **over IPv4** between h1 and h2. Configure the `iperf` client to send 2 seconds of traffic to the server.
11. Stop the traffic captures on r1 and save the Wireshark output to `L3-2-1.eth0.pmtu.pcapng` and `L3-2-1.eth1.pmtu.pcapng`.

Use the data captured with Wireshark in L3-2-1.eth0.pcapng, L3-2-1.eth1.pcapng, L3-2-1.eth0.pmtu.pcapng and L3-2-1.eth1.pmtu.pcapng to answer the questions. Support your answers with the saved Wireshark data. The first questions are only about the first two Wireshark traces. It will be clearly stated when you need to look at the second set of Wireshark trace files.

L3-2-1 Both UDP streams in the first part of the exercise were fragmented. What differences (regarding fragmentation) do you see between IPv4 and IPv6? /1

L3-2-2 Why is it important that the path MTU cache was empty for this experiment? What would be different in the case of IPv6 if we did the experiment twice in a row without clearing the cache? What about IPv4? /1

L3-2-3 Identify which fields in the header(s) are used for fragmentation in IPv4. /1

L3-2-4 Identify which fields in the header(s) are used for fragmentation in IPv6. /1

L3-2-5 Compare the amount of bytes that are needed for fragmentation in both IP protocols. Consider the overhead when fragmentation is needed and when it is not needed. /1

L3-2-6 IPv4 allows fragmentation in intermediate routers where IPv6 forbids this. However, IPv4 can also perform end-to-end fragmentation. This is what happened in the last part of the exercise,

where you saved the second set of Wireshark trace files. Take a look at those last trace files, and compare them to the first set (only look at IPv4). What difference(s) regarding fragmentation do you see when you compare both iperf sessions? /1

L3-2-7 Compare the way IPv4 performs end-to-end fragmentation to the way this is done in IPv6. What are the differences and similarities? /1

L3-2-8 What do you think is the main advantage of not allowing intermediate hosts to fragment IPv4 packets? /1

Part 3. NAT, DHCP and Router Advertisements

In this part we set up NAT, DHCP and router advertisements. These protocols are used to simplify network configurations. DHCP and router advertisements are used to automatically configure clients when they connect to a network. NAT is used to conserve IPv4 addresses.

Dynamic Host Configuration Protocol (DHCP)

The Dynamic Host Configuration Protocol (DHCP) can be used to dynamically set and change configuration parameters of Internet hosts, including IP address, subnet mask, default router, and Domain Name System (DNS) server. DHCP is based on a client-server model. DHCP clients send requests to a DHCP server and the server responds with an allocation of IP addresses and other configuration parameters.

A DHCP client is assigned an IP address for a limited period of time, which is called a lease. Information on current leases is stored at both the client side and the server side.

On a Linux system, a DHCP server is started with the command `dhcpd`. The DHCP server reads the configuration file `/etc/dhcp/dhcpd.conf`. The configuration file contains information on available IP addresses, and other configuration information. The following is an example of a configuration file for a DHCP server:

```
#dhcpd.conf file
default-lease-time 600;

subnet 10.0.0.0 netmask 255.0.0.0 {
        range 10.0.100.0 10.0.255.255;
        option routers 10.0.0.1;
        default-lease-time 120;
}
```

The DHCP client is assigned an IP address for a period of time that is known as a lease. The above configuration file assigns IP addresses for a lease time of 600 seconds (default-lease-time). For requests on network `10.0.0.0/8`, the DHCP server assigns IP addresses in the range `10.0.100.0 - 10.0.255.255`, assigns `10.0.0.1` as the default gateway, and limits the lease of addresses to 120 seconds, thus, overruling the global limit of 600 seconds.

Starting a DHCP client on a host (e.g. h1) is simple. It can be done with one command: `dhclient h1-eth0`.

However, because all the (virtual) hosts created by ipmininet share the same underlying file system, when you have multiple virtual hosts all running `dhclient`, they would all start to use the same leases file and PID file simultaneously. This should be avoided. In order to do so, some optional parameters should be added to the `dhclient` command:

`-lf <leases_file>` to use a different client leases file on each virtual host.

`-no-pid` to disable using a PID file.

The exact same parameters can (and should!) also be used on any host running a `dhcpd` server. This way, it is 100% sure you are performing the experiment without having stale files from previous attempts (or from concurrently running dhcp servers).

Don't forget to add these extra parameters when you are performing the next exercises!

Exercise 3: DHCP Setup

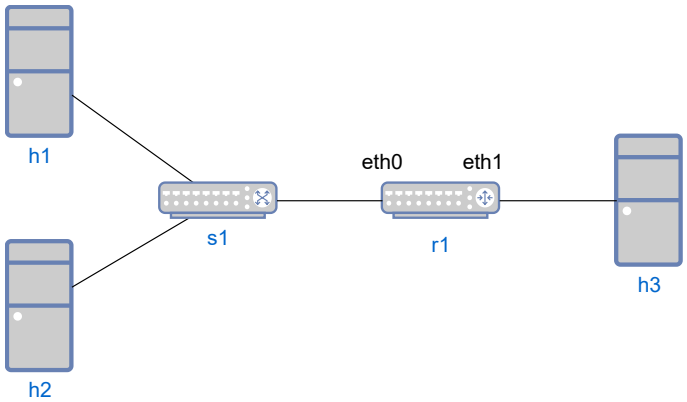

Figure 3.3: Network configuration for part 3.

hosts	Interface eth0	Interface eth1
h1	N/A	N/A
h2	N/A	N/A
r1	`fc00:0:0:1::1/64` `10.0.1.1/24`	`fc00:0:0:2::1/64` `10.0.2.1/24`
h3	N/A	N/A

Table 3.3: IP addresses of all PCs for part 3.

1. Write a python script that results in the topology shown in figure 3.3 and table 3.5 and save it to `L3-3-1.py`.
2. We want to analyze packets used for DHCP configuration. Start to capture traffic with Wireshark on interface eth0 of r1.
3. Configure r1 as a DHCP server:
   (a) On r1, set up a configuration file so that IP addresses are assigned as follows:
   - The default and maximum lease time should be 2 minutes.
   - On network `10.0.1.0/24`, IP addresses are assigned in the range `10.0.1.50 - 10.0.1.100` with default gateway `10.0.1.1` and DNS server `8.8.8.8`.
   - On network `10.0.2.0/24`, IP addresses are assigned in the range `10.0.2.50 - 10.0.2.100` with default gateway `10.0.2.1` and DNS server `8.8.8.8`.

   Save the configuration file to `/home/computernetwerken/dhcpd.conf`
   (b) Make sure the file `/home/computernetwerken/dhcpd.leases` exists. It can be an empty file.
   (c) Open a `xterm` on r1, and start the DHCP server by typing:

```
r1% dhcpd -f --no-pid -cf /home/computernetwerken/dhcpd.conf \\
-lf /home/computernetwerken/dhcpd.leases
```

The DHCP server daemon listens for requests from DHCP clients on all its interfaces. In Linux, the DHCP server must be restarted each time the configuration file is modified. Since only one DHCP server can run at a time, you may need to terminate the current DHCP server process.

4. Start DHCP clients on h1, h2 and h3 with the command:

```
% dhclient hX-eth0 --no-pid -lf /home/computernetwerken/hX.dhclient.leases
```

5. Use the commands `ifconfig` and `route -4 -n` to see if the automatic configuration is successful on all hosts. Verify with a `pingall` all hosts can reach each other.
6. Stop the Wireshark capture. Save it to `L3-3-1.pcapng`.
7. Make sure you include the dhcp server configuration file in the `/traces` folder of your lab report.

Use the data captured with Wireshark in L3-3-1.pcapng, to answer the questions. Support your answers with the saved Wireshark data.

L3-3-1 Which IP addresses are assigned to h1 and h2? /1

L3-3-2 Observe and interpret the different DHCP packets. What is their purpose? /1

L3-3-3 Identify and interpret all option fields in the DHCP packet types that you observe. /1

L3-3-4 Observe the source and destination IP addresses of the packets that are sent between DHCP client and DHCP server. /1

L3-3-5 How is it possible that a host can send and receive DHCP packets, even though it does not have an IP address? /1

L3-3-6 In most client-server applications, the port number of a server is a well-known number (e.g., an HTTP server uses port number 80, an ssh server uses port number 22, etc.), while the client uses a currently available (random) port number. DHCP is different. Here, both the client and the server use a well-known port: UDP port 67 for the DHCP server, and UDP port 68 for the DHCP client. Refer to RFC 2131 and provide an explanation for this protocol design choice. /1

Exercise 4: DHCP leases

In this exercise you will see what happens when a client can't reach the DHCP server anymore. We will simulate this by disabling the DHCP server on r1.

1. Start the script from the previous exercise, and make sure all hosts get an IP address from the DHCP server.
2. Wait until every host has renewed their lease at least once. This cannot take longer than 120 seconds.
3. Start to capture traffic with Wireshark on interface eth0 and eth1 of r1. You can select both interfaces simultaneously in Wireshark so all data is stored in a single `pcapng` file.
4. Stop the DHCP server (Ctrl-C).
5. Watch what the clients do when they try to renew their leases. Wait at least 3 minutes. After this the clients should no longer have an IP address.
6. After the clients have lost their IP address, stop the Wireshark capture and save it to `L3-4-1.pcapng`.
7. Make sure to include a copy of the dhcp server leases file, as well as the dhcp client leases file from all hosts in your report!

Use the data captured with Wireshark in L3-4-1.pcapng, to answer the questions. Support your answers with the saved Wireshark data.

L3-4-1 Why does a DHCP server have a leases file? What is it used for? /1

L3-4-2 Explain the entries in the client lease file. /1

L3-4-3 What attempts did the clients make to renew their leases? You should notice two stages. What happens during those two stages? What happens afterwards (i.e. after the renewal fails)? /1

Network Address Translation (NAT)

Network Address Translation (NAT) refers to a function that replaces the IP addresses (and possibly the port numbers) of IP datagrams. NAT is run on routers that connect private networks to the

public Internet, to replace the IP address-port pair of an IP packet with another IP address-port pair. Generally, the operations of NAT are specified in terms of a set of rules which determines how IP addresses are to be replaced.

Often, a NAT device is referred to as a NAT box. One of the reasons for using NAT is that it conserves IP addresses. NAT allows hosts in a private network to share public IP addresses, or to limit the use of public IP addresses to a small number of hosts in the private network.

Private networks may have IP addresses that are non-Internet routable, as specified in RFC 1918. This means that the Internet routers do not have entries in their routing tables for these addresses.

In the following exercise, we consider a special use of NAT that allows multiple private IP addresses to be mapped to a single public IP address. This use of NAT is called IP masquerading, port address translation (PAT) or Network Address and Port Translation (NAPT). Here, the private network has only a single public IP address, but has multiple hosts in the private network. IP Masquerading modifies the port number of packets so that the single public IP address can be overloaded.

The Linux kernel on all PCs has been built with netfilter, which adds the ability to set IP packet filters in a Linux system. IP packet filters are used to add firewalls as well as NAT functionality to a system. The `iptables` command is used to set up, maintain, and inspect IP packet filter rules to a Linux kernel:

On a Linux system, the configuration of NAT manipulates a set of rules of the netfilter utility, called NAT table. The rules in the NAT table are grouped in so- called chains. Two of the built-in chains are called `PREROUTING` and `POSTROUTING`:

`PREROUTING`
: The rules in this chain are applied to incoming datagrams.

`POSTROUTING`
: The rules in this chain are applied to outgoing datagrams. The main rule is SNAT (Source Network Address Translation), which specifies how the source address of an outgoing IP datagram should be modified.

Commands that manipulate the NAT table start with `iptables -t nat`. The following are some of the most important commands that manipulate the NAT table:

`iptables -t nat -L`
: Displays all rules in the NAT table

`iptables -t nat -F POSTROUTING`
: Deletes all rules in the `POSTROUTING` chain of the NAT table

`iptables -t nat -A POSTROUTING -j SNAT --to IPAddr -s PrivateIPAddr/netmask`
: Adds the following rule to the `POSTROUTING` chain of the NAT table: “In IP datagrams that go to the public network, the IP source address PrivateIPAddr/netmask is changed to IPAddr”.

Example: The source address of outgoing IP datagrams that match "10.0.1.0/24" is changed to 128.195.7.32.

```
iptables -t nat -A POSTROUTING -j SNAT --to 128.195.7.32 -s 10.0.1.0/24
```

Exercise 5: A NAT setup

In this exercise, we use the same topology as in figure 3.3, and r1 will be configured as a NAT router. Configure the topology to use the IPv4 addresses as shown in the table below (note we are only using IPv4 in this exercise).

hosts	Interface eth0	Interface eth1
h1	10.0.1.101/24	N/A
h2	10.0.1.102/24	N/A
r1	10.0.1.1/24	128.66.0.1/24
h3	128.66.0.103/24	N/A

Table 3.4: IP addresses of all PCs for exercise 5.

1. Write a script that sets up the correct topology and save it to `L3-5-1.py`.
2. On r1, add a rule to the NAT table so that the IP source address of outgoing IP datagrams from network `10.0.1.0/24` are set to IP address `128.66.0.1`. By doing this you turn the `10.0.1.0/24` network into a private network that uses r1 to communicate with the public network (`128.66.0.0/24`).
3. On h3, execute the following command: `ip route del default via 128.66.0.1`. This is necessary, as mininet doesn't make a distinction between "private" addresses that will be NATed, and "public" addresses. If h3 would set its default router to r1, packets would be routed normally, as usual, but in this case, this needs to be avoided.
4. Observe traffic at a NAT device:
   (a) To observe the IP address translation, capture packets on both interfaces of r1.
   (b) Start up the `/sbin/sshd` server on h1, h2, r1 and h3.
   (c) Establish a set of ssh sessions. Try to create following sessions (there is no need to log in; you can CTRL-C at the login prompt) and see which ones are successful:
   > h1 → h2
   > h1 → h3
   >
   > h2 → r1 (eth0)
   > h2 → h3
   >
   > h3 → r1 (eth1)
   > h3 → h1

   Note which ssh commands succeed. You will need this information to answer the questions.
   (d) Generate traffic between h1 and h3 and between h2 and h3 at the same time. You can do this by establishing an ssh session between these hosts, logging in (with as username *computernetwerken*, and as password *mvkbj1n*), and maybe type some simple commands.
   (e) Also issue a ping command from h1 to h3, and, simultaneously, from h2 to h3. Make sure you have some pings that are sent in both sessions, and the stop both pings.

5. Save the Wireshark data to `L3-5-1.eth0.pcap` and `L3-5-1.eth1.pcap`.

Use the data captured with Wireshark in L3-5-1.eth0.pcap and L3-5-1.eth1.pcap the questions. Support your answers with the saved Wireshark data.

L3-5-1 For each of the `ssh` commands above, provide an explanation why a command succeeds or fails. /1

L3-5-2 Explain how r1 knows which ssh packets should be forwarded to h1 and which ssh packets should be forwarded to h2. /1

L3-5-3 Explain how r1 knows which ping packets should be forwarded to h1 and which ping packets should be forwarded to h2. /1

L3-5-4 Multiple hosts using the same IP address (i.e. behind a NAT) can be troubling in some situations like IP ban and traceability of suspicious traffic. Explain the impact of using NAT in these scenarios. /1

Router Advertisements

Like DHCP, router advertisements can be used to dynamically set and change configuration parameters of Internet hosts.

On a Linux system, a router advertisement daemon is started with the command `radvd`. The daemon by default reads the configuration file `/etc/radvd.conf`. However, using the `-C` flag, an alternative location of the radvd configuration file can be passed on to the daemon. The configuration file contains information on available IP addresses, and other configuration information. The following is an example of a configuration file:

```
#radvd.conf file

interface eth0 {
        AdvSendAdvert on;
        MaxRtrAdvInterval 30;
        prefix 2001:db8::/64 {
```

```
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr off;
                AdvPreferredLifetime 60;
        };
        RDNSS 2001:4860:4860::8888 {};
};

interface eth1 {
        AdvSendAdvert on;
        MaxRtrAdvInterval 30;
        prefix 2001:db8:0:1::/64 {
                AdvPreferredLifetime 60;
        };
        RDNSS 2001:4860:4860::8888 {};
};
```

This config file will enable router advertisements on eth0 and eth1. The option "AdvSendAdvert" enables this. For each interface a prefix and a DNS server are specified. The configuration for eth0 and for eth1 are identical in this example. In the first case the parameters for the prefix are explicitly specified. For eth1, parameters with default values are omitted. "AdvPreferredLifetime" indicates how long the given prefix should be preferred by the host. When the prefix is not refreshed within this interval, another prefix may become preferred. The prefix remains valid for a period of "AdvValidLifetime" though.

Exercise 6: Router Advertisements on Linux PC

For this exercise, you once again start from the same toplogy as in figure 3.3, but this time with the following IP addresses set:

hosts	Interface eth0	Interface eth1
h1	N/A	N/A
h2	N/A	N/A
r1	`fc00:0:0:1::1/64`	`fc00:0:0:2::1/64`
h3	N/A	N/A

Table 3.5: IP addresses of all PCs for part 3.

1. Write a script that results in the correct topology and save it to `L3-6-1.py`.
2. Start to capture traffic with Wireshark on interface eth0 of r1.
3. Configure r1:
   (a) On r1, create up a `radvd` configuration file so it distributes the correct prefixes and assigns a DNS server as shown in the example. Save it to `L3-6-1.conf`
   (b) On r1, start the router advertisements daemon (run the `radvd -C L3-6-1.conf` command).
4. Use the commands `ip -6 addr show` and `route -6 -n` to see if the automatic configuration is successful on all hosts. Every host should be able to ping all other hosts on their global IPv6 address at this point. Issue a ping from h1 to h3 and from h3 to h2.

5. Bring the interface eth0 of h2 down and back up:
6. Now shut down the router advertisements daemon on r1 with `pkill`
7. Wait at least 60 seconds while watching the IP address and route configuration of eth0 on h1 (hint: use the `watch` command). Once this changes, save the result to `L3-6-5.h1.txt`.
8. Stop the Wireshark capture and save it to `L3-6-1.pcapng`.

Use the data captured with Wireshark in L3-6-1.pcapng to answer these questions. Support your answers with the saved Wireshark data.

L3-6-1 Identify a router advertisement in the Wireshark capture. Which ICMP Options do you see? For each option, explain what it is used for. /1

L3-6-2 What is the destination IP address of a router advertisement? /1

L3-6-3 After resetting interface eth0 of h2 you should have seen a router solicitation packet. What was the destination IP address of this packet? /1

L3-6-4 The router responded to the solicitation with a router advertisement. What destination IP address was used and why? /1

L3-6-5 After 60 seconds, the preferred lifetime of the prefix expires. How do you see this and what does this mean? Use the output saved in L3-6-5.h1.txt. /1

L3-6-6 Explain how hosts get a global IPv6 address based on the information in the router advertisement. /1

L3-6-7 Why doesn't the router advertisement daemon keep track of its hosts like an IPv4 DHCP server? /1

IPv6 Privacy Extension

Privacy was not a pressing matter when the IPv6 standard was proposed in 1995. Later on, a IPv6 privacy extension was proposed in RFC 4941.

Exercise 7: IPv6 privacy extension

This exercise will demonstrate how and when Linux PCs use the privacy extension. They will generate random globally unique IPv6 addresses based on the prefix information in router advertisements.

1. Re-use the previous setup.
2. Change the settings `temp_prefered_lft` and `temp_valid_lft` to 60 seconds on h1, h2 and h3.

   ```
   % echo 60 > /proc/sys/net/ipv6/conf/hX-eth0/temp_prefered_lft
   % echo 60 > /proc/sys/net/ipv6/conf/hX-eth0/temp_valid_lft
   ```

   Normally these timers are much longer. We lower them here to demonstrate what happens when they expire.
3. Enable the privacy extension on h1, h2 and h3 by writing a 2 to the configuration file `use_tempaddr`:

   ```
   % echo 2 > /proc/sys/net/ipv6/conf/hX-eth0/use_tempaddr
   ```

   Writing a 2 to this file enables the generation and use of a random address.
4. Start capturing packets on the eth0 interface of r1.
5. Start the router advertisements daemon on r1.
6. You should see that the hosts each get a new IPv6 address when they receive a router advertisement. Use the command `ip -6 addr show` to verify this.
7. Set up a continuous ping from h2 to h3. Use the global IPv6 address of h3 (*not* the temporary address).
8. At the same time open a ssh session from h2 to h3 (use the same global address). Generate traffic over this session. You can use the watch command for example:

   ```
   % watch date
   ```
9. Wait at least 60 seconds and observe what happens. After that, stop the Wireshark capture and save it to `L3-7-1.pcapng`.


An session that doesn't respond can be terminated with `pkill ssh`.

Use the data captured with Wireshark in L3-7-1.pcapng, to answer the questions. Support your answers with the saved Wireshark data.

L3-7-1 Which of its IP addresses does h2 use as source address for the ping? And which one for the ssh session? /1

L3-7-2 Explain what you observed concerning the ping. /1

L3-7-3 Explain what you observed concerning the ssh session. /1

L3-7-4 What is different between the ssh session and the ping command? /2

L3-7-5 What is the purpose of the privacy extension? What potential privacy violation does it prevent. /1

Acronyms

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

IP Internet Protocol

IPv6 Internet Protocol (IP) version 6

IPv4 IP version 4

NAT Network Address Translation

UDP User Datagram Protocol

MTU Maximum Transmission Unit