ComNet - Lab n°1

Introduction to the networking testbed

This first handout is meant to familiarize you with the testbed environment that we use for the course's laboratory exercises. Throughout the semester, we will be generating and then analyzing real network traffic traces. Today's lab starts with a tutorial introduction to the format that we will use for describing traffic traces by hand (Part 1) and an introduction to the `wireshark` tool for automated analysis of traces (Part 2). Then you learn about the networking testbed on which we will generate our traces (Part 3). It finishes with a practical exercise (Part 4) in which you generate traffic on the testbed and analyze it with `wireshark`, before you clean up the environment at the end of the lab (Part 5). At the end of this handout, you have a reference page that provides the structure of common link-, network-, and transport-layer frames/packets/datagrams, and that will aid you in conducting your analysis.

1 Introduction to trace analysis (without computer assistance)

To study the traffic exchanged in a network, administrators commonly use hardware or software capture tools called **sniffers**. A software sniffer comprises a specialized multi-protocol capture and analysis program, such as `tcpdump` or `wireshark`, running on general-purpose equipment, such as a PC with a network interface card.

1.1 Network traffic traces

Traces are typically captured at the link layer and consist of a sequence of (potentially truncated) frames. Sniffer software receives each frame from the card as a binary dump. The basic way to display the frame is byte by byte, in a three column format:

❶	❷	❸
0000	00 50 7f 05 7d 40 00 10 a4 86 2d 0b 08 00 45 00	.P..}@.. ..-...E.
0010	02 19 17 98 40 00 40 06 6c 14 0a 21 b6 b2 c0 37	....@.@. l..!...7
0020	34 28 84 b3 00 50 b6 94 b0 b8 24 67 89 e9 80 18	4(...P.. ..$g....
0030	16 d0 60 e4 00 00 01 01 08 0a 00 6f a7 32 00 00	..‘..... ...o.2..
0040	00 00 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31	..GET / HTTP/1.1
0050	.. ..	.. ..

❶ the first column consists of 4 hexadecimal digits indicating the **rank** of the first byte of the current line in the frame;

❷ each line of the second column displays 16 **bytes of the frame**, each byte being expressed by two hexadecimal characters, corresponding to a decimal value between 0 and 255;

❸ each line of the third column displays the 16 **ASCII characters** that correspond to the 16 bytes of that line (this can help in interpreting the contents of the frame if the bytes represent printable ASCII characters, which are in the range 0x20-0x7E (decimal 32-126); if they are out of this range then a period '.' is shown.

Important points, prior to analyzing our first frame:

- The sniffer typically captures Ethernet frames. But network adapter cards may limit the information they provide to the kernel. The frames shown here include **neither the preamble nor the CRC.**
- In order to communicate effectively, both in the classroom and in your professional life, you will need to observe the standard conventions on how to write various values, notably:
  - **Ethernet addresses**: colon hexadecimal notation (e.g., `00:50:04:ef:6b:18`)
  - **EtherType**: hexadecimal notation (e.g., 0x0806)
  - **IPv4 addresses**: dotted decimal notation (e.g., `10.1.1.3`)
  - **IPv6 addresses**: compact colon hexadecimal notation (e.g., `2001:db8:abcd:1::1234:5678`)
  - **protocol number** and **port number**: decimal (e.g., 17)
  - ...

1.2 Analyzing a frame by hand

To fully understand how a sniffer analyzes a frame, we analyze **by hand** the beginning of a frame captured on an Ethernet network. Though tedious, this exercise is necessary if you are to acquire a good understanding of encapsulation mechanisms and develop a critical eye that will help you detect potential errors of interpretation by automated tools. The reference sheet on **page 14** will help you decrypt the structures encountered at each protocol layer.

```
0000  00 50 7f 05 7d 40 00 10  a4 86 2d 0b 08 00 45 00   .P..}@.. ..-...E.

0010  02 19 17 98 40 00 40 06  6c 14 0a 21 b6 b2 c0 37   ....@.@. l..!...7

0020  34 28 84 b3 00 50 b6 94  b0 b8 24 67 89 e9 80 18   4(...P.. ..$g....

0030  16 d0 60 e4 00 00 01 01  08 0a 00 6f a7 32 00 00   ..‘..... ...o.2..

0040  00 00 47 45 54 20 2f 20  48 54 54 50 2f 31 2e 31   ..GET /  HTTP/1.1

0050  0d 0a 48 6f 73 74 3a 20  77 77 77 2e 78 69 72 63   ..Host:  www.xirc

0060  6f 6d 2e 63 6f 6d 0d 0a  55 73 65 72 2d 41 67 65   om.com.. User-Age

0070  6e 74 3a 20 4d 6f 7a 69  6c 6c 61 2f 35 2e 30 20   nt: Mozi lla/5.0

0080  .. ..
```

1. Reveal the frame's composition by sketching on the printout the boundaries of each of its constituent structures.
2. What information can you observe concerning the link layer?
3. Sketch on the printout the boundaries of each of the packet fields. What is the size of this packet and what can you deduce from this? Does the packet include options and what is the impact on the structure of the packet? What are the source and destination addresses of the packet?
4. Sketch on the printout the details of the data carried by the packet. Which transport protocol is used? Which port numbers? What do they indicate?
5. *The reference page at the end of this document does not describe the application layer.* Nonetheless, can you deduce anything about this layer in the trace?

2 Frame analysis with `wireshark`

The `wireshark`[1] program conducts both frame capture and protocol analysis. It is capable of capturing frames directly from the LAN to which your computer is connected, through the network interface card. In this lab, however, we will use it purely to analyze a trace that was previously captured and stored as a file.

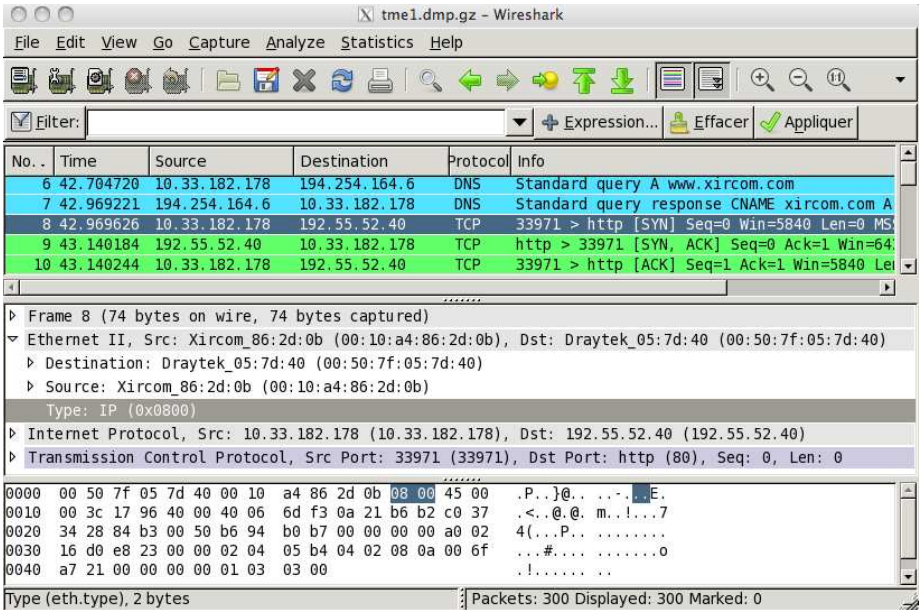


Figure 1: The main window of `wireshark`

Note: To work on these exercises from outside the university, you can copy the trace file during the lab or obtain the trace file from:

`http://www-npa.lip6.fr/~fourmaux/Traces/labV8.html`

2.1 Introduction to `wireshark`

Start by logging into a PC in the computer lab using a GNU/Linux account (either with your own account, or your lab partner does so with their account). Sniffer software requires administrator's rights in order to capture network traffic, which might contain private information. However, in order to maintain a stable and secure shared environment for hundreds of people, the PPTI[2] does not grant students administrator's rights on these machines. Later on, we will see how to capture packets on other, less sensitive, machines. For now, you will use only the protocol analysis capabilities of `wireshark` on prerecorded traces. For the program to work properly, you must explicitly run it in user mode rather than in administrator mode.

Look for "Wireshark" in submenus of the computer's "Applications" menu. Select the version that indicates that the software is run without administrator rights.[3]

Once `wireshark` is launched, a new window appears that is initially empty because no trace has been captured or loaded from a file. To load a pre-captured trace, click the "File" menu at the top of the window and select "Open". A file selection window "Open Capture Files" appears. Choose the file:

`/Infos/lmd/2022/master/ue/MU4IN001-2022oct/tme1.dmp.gz`

Do not specify the "Filter" field (see below). Disable: ❑ "Enable MAC name resolution", ❑ "Enable network name resolution" and ❑ "Enable name resolution transportation". Click on Open. A previously captured trace is loaded and you will be able to analyze it. The application window should be similar to the one shown in Figure 1.

[1] `wireshark` is free software. It is available for many hardware platforms and operating systems, beyond the x86 system with GNU/Linux that you are currently using. You can download it at `http://www.wireshark.org`.

[2] PPTI : Plateforme Pédagogique et Technique d'Informatique

[3] If such an option does not appear, choose whichever version of "Wireshark" is available, and perhaps it will allow you to specify user mode execution. If this fails, generally because default environmental variables start the application in administrator mode, try starting the program from the command line by entering `/usr/sbin/wireshark` in a terminal window.

1. Describe the contents of the three panes displayed in the `wireshark` main window
2. Which format is used to represent data in the third pane?
3. What are the different protocols that you can observe in the captured trace?
4. How many protocols is the `wireshark` version that you are running capable of analyzing?

2.2 Display and coloring filters in `wireshark`

1. Refer to the documentation (click on the "Help" menu and select "Manual Pages") to describe the syntax used by `wireshark`'s display and coloring filters. Do not confuse these filters with the capture filters, which have another syntax that we will not use.
2. Create a filter that selects only frames that contain the application protocol NTP. In the "Analyze" menu, you will find a "Display Filters..." item that opens a window with a list of filters. Click on the +Expression... button to open a window that helps you to create a new filter. Apply your filter. What do you observe?
3. Delete the filter and create a new one to color in purple all frames that contain the NTP protocol.
4. You can combine filters using classical Boolean operators. Filter the display to only show frames that contain the NTP protocol and those that contain the DNS protocol.

2.3 Analyzing HTTP traffic

Following on the manual frame analysis in Part 1.2:

1. In the trace that is loaded into `wireshark`, can you find the frame that you previously analyzed by hand? If yes, check your analysis by comparing it with `wireshark`'s.
2. Select and display **all** frames of the TCP connection that starts at frame 8, and then color in red only those that contain HTTP data.
3. Describe what you observe in the remainder of the trace. If you detect more than one connection, describe the relationship between the connections.
4. Can we simply display the application layer content of a TCP connection using `wireshark`?

3 Introduction to the networking testbed

As mentioned in Part 2.1, students do not have the administrator's rights required to capture real time network traffic from the PCs in the computer lab that are administered by the PPTI. To provide this capability, we have created a separate stand-alone testbed in the computer lab.

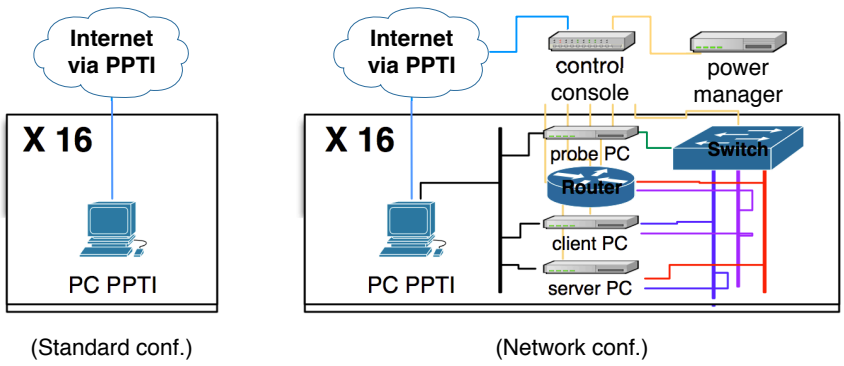
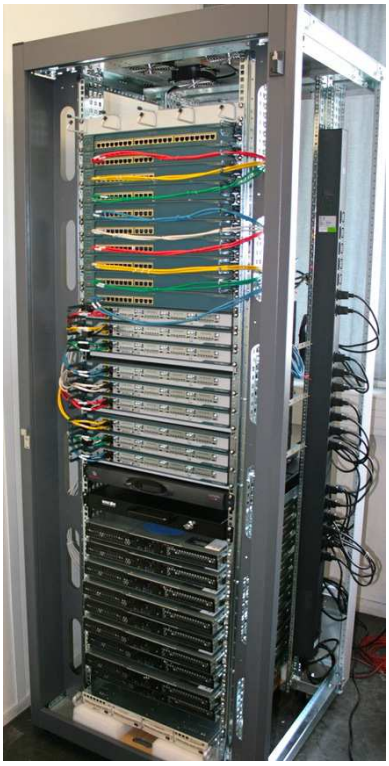


Figure 2: Enhancement of the computer lab resources

The testbed enhances the capabilities of the PPTI's PCs. In addition to the standard services provided by the PPTI (access to user accounts, a range of software, and the internet), students can control the testbed's hosts, switches, and routers. Figure 2 shows how these services have evolved with the arrival of the testbed.

3.1 Hardware components of the networking testbed

The testbed consists of two 42U 19-inch racks containing the following equipment:



- **CISCO Catalyst 2960 switches**, 16 ports Ethernet:
  - control and management (port and VLAN)
  - port mirroring (for frame capture)
- **CISCO 2801 routers**, 2 ports Ethernet, IOS 12.4 with two level of service:
  - **IP Base:** IPv4, RIP, OSPF, IGMP, Netflows, QoS, RSVP, DiffServ, DHCP, NAT, SNMP, RMON, NTP, L2TP, AAA...
  - **Advanced IP Services:** IOS IP Base + IPv6, BGP, Mobile IP, VoIP, SIP, H323, Firewall, IPSEC, VPN, AES...
- **1U rackable PC**, Intel Xeon E3-1230v5, 16 Go RAM, 4 NIC Ethernet, running virtual machines (VM) with:
  - **Debian GNU/Linux** including classical Unix networking environment (Telnet, SSH, FTP, TFTP, SCP, SFTP, HTTP, SMTP, POP, IMAP, Webmail, SNMP, DNS...)

3.2 Student guide to the networking testbed

Each pair of students accesses the testbed via an PPTI-administered PC in the computer lab. This is a classic PC with two network interface cards (NICs). One card allows the usual access to the internet via the PPTI's network, while the other provides direct access to the networking testbed. Indeed, the only way to access the testbed is via the PCs in the lab (room 31-208), which implies that students must either be physically present in the room, or logged in remotely via SSH (`ssh ssh.ufr-info-p6.jussieu.fr` then `ssh ppti-14-503-`**N** and using text oriented command). There is no routing or relaying between the PPTI's network and the testbed, which ensures that the testbed remains isolated and does not raise security concerns.

PPTI-administered PCs `ppti-14-503-01` through `ppti-14-503-08` are connected to rack 1 and `ppti-14-503-09` through `ppti-14-503-16` are connected to rack 2.

We use **N** to designate the last number of the host name of an PPTI-administered PC. PC **N** has direct reserved access to three machines on the testbed:

- the switch **N**
- the router **N**
- the racked host **N**, on which there are three virtual machines (VMs):
  - the "client" VM **N**1
  - the "monitor" VM **N**2
  - the "server" VM **N**3

Figure 3 shows the setup of the experimentation environment, with the three virtual machines.

Your tutor will provide you, when needed, with the logins and passwords for accessing the testbed equipment.

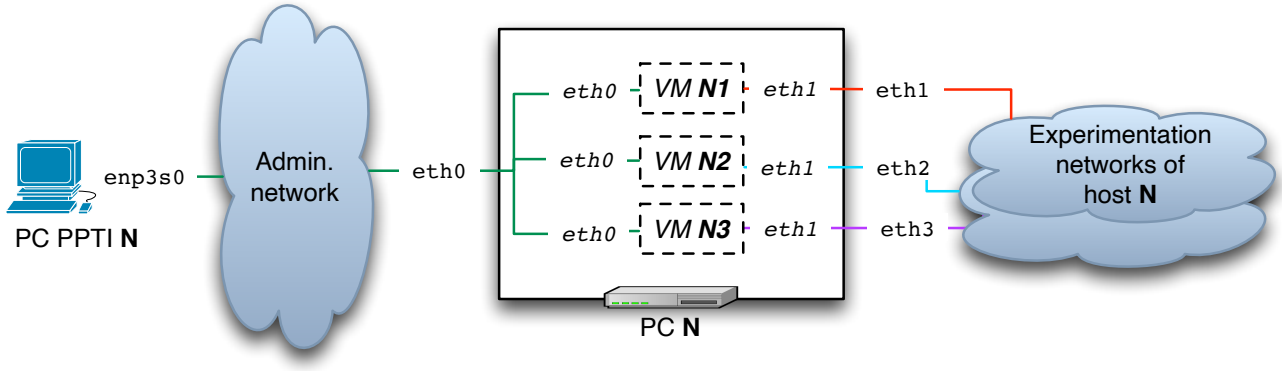


Figure 3: Configuration of the three VMs of the platform

3.3 Available topologies

The testbed allows students to deploy a variety of virtual network topologies. (We use virtual topologies because these are easily configured and reset without physical access to the racks, which are closed to the students.) Figure 4 shows the underlying physical links on which the packets transit.

3.3.1 Topology 1 (no router – first labs)

This first virtual topology consists of two hosts exchanging traffic over a LAN. As shown in Figure 5, the hosts are VM **N1**, on which you will run a client program, and VM **N3**, on which you will start a server. The client-server traffic will go over the experimental network VLAN **N**1. On VM **N2**, you will run a sniffer to capture this traffic. You will control the experiment from the PPTI PC, accessing the virtual machines via the administrative network VLAN 200.

3.3.2 Topology 2 (one router – later labs)

This second virtual topology also consists of the two hosts, VMs **N1** and **N3**, but in this case each one is on a separate LAN, VLAN **N**1 and VLAN **N**2 respectively, which are connected by a router. See Figure 6. In addition to controlling the hosts via the administrative network VLAN 200, you will also be able to configure the router. This configuration allows you to study the behavior of routed traffic, sniffing, as before, from VM **N2**.

3.3.3 Topology 3 (multiple routers – future labs)

The testbed is capable of providing more advanced configurations, involving multi-hop routing (multiple routers running RIP, OSPF, and/or BGP routing protocols). You will explore these in some of the other courses offered by the Networks Masters program.

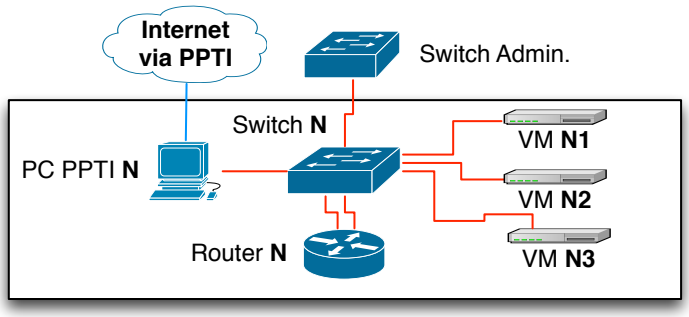


Figure 4: Physical topology associated with an PPTI host

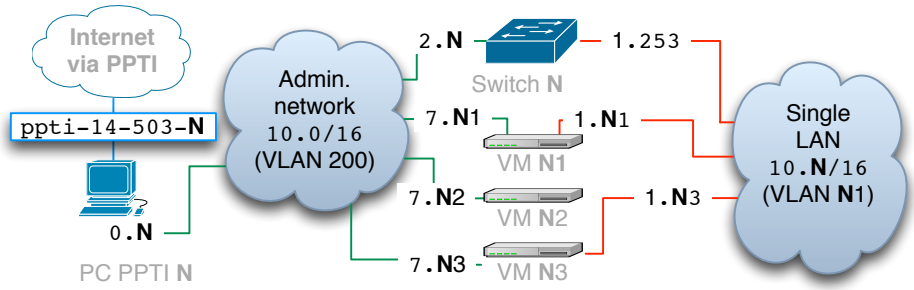


Figure 5: Virtual Topology 1 (one LAN)

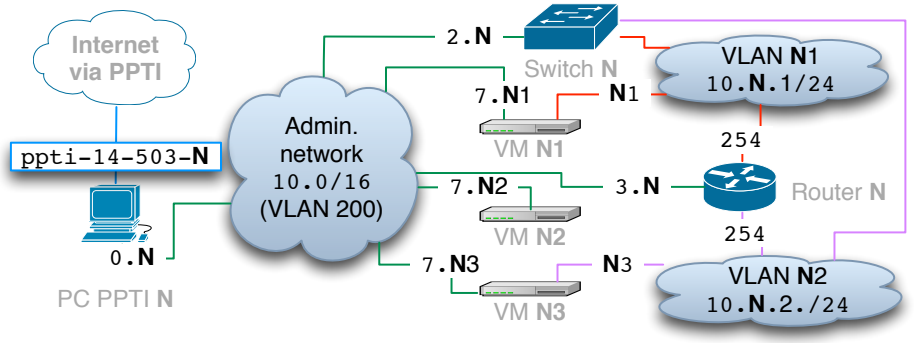


Figure 6: Virtual Topology 2 (two LANs and one router)

3.3.4 IPv4 addressing conventions relative to host N

Mainly two kind of VLAN are used in the networking testbed:

- The management VLAN (VLAN 200) and its management IPv4 addresses for:
  - the PPTI host **N** interface to access to the testbed via the management network: `10.0.0.`**N**
  - the switches: `10.0.2.`**N**
  - the routers: `10.0.3.`**N**
  - the "client" VMs **N**1 (`eth0`): `10.0.7.`**N**1
  - the "monitor" VMs **N**2 (`eth0`): `10.0.7.`**N**2
  - the "server" VMs **N**3 (`eth0`): `10.0.7.`**N**3
- The experimentation VLANs **N**v (with $0 < v$) and their experimentation IPv4 addresses for:
  - the "client" VMs **N**1 of VLANs **N**v (`eth1`): `10.`**N**`.`v`.`**N**1
  - the "server" VMs **N**3 of VLANs **N**v (`eth1`): `10.`**N**`.`v`.`**N**3
  - the routers of VLANs **N**v: `10.`**N**`.`v`.254`

3.4 Generating and capturing traffic

The PPTI host is connected to the the administrative network of the networking testbed through a local interface as shown in Figure 3. You can check the configuration of the interfaces with the `/sbin/ifconfig` Unix command. If the interface with the IPv4 address `10.0.0.`**N** is unknown, restart the interface or reboot your PPTI host.

3.4.1 Remote control of the three virtual machines via SSH and X11 tunneling from the room 503 PCs

For whichever topology you create, you will be controlling the virtual machines remotely via SSH from PPTI PC **N**. Your login on all VMs is the word `etudiant`, and your tutor will provide you the password. To open each session, open a terminal window on host **N** and type the following at the command line:

- for the "client" vm**N**1 (window 1): `ssh -Y etudiant@10.0.7.`**N**1
- for the "monitor" vm**N**2 (window 2): `ssh -Y etudiant@10.0.7.`**N**2
- for the "server" vm**N**3 (window 3): `ssh -Y etudiant@10.0.7.`**N**3

The "`-Y`" option, which redirects the X11 graphical environment from the virtual machine to your local host, can be skipped if a purely textual environment is sufficient for your purposes.

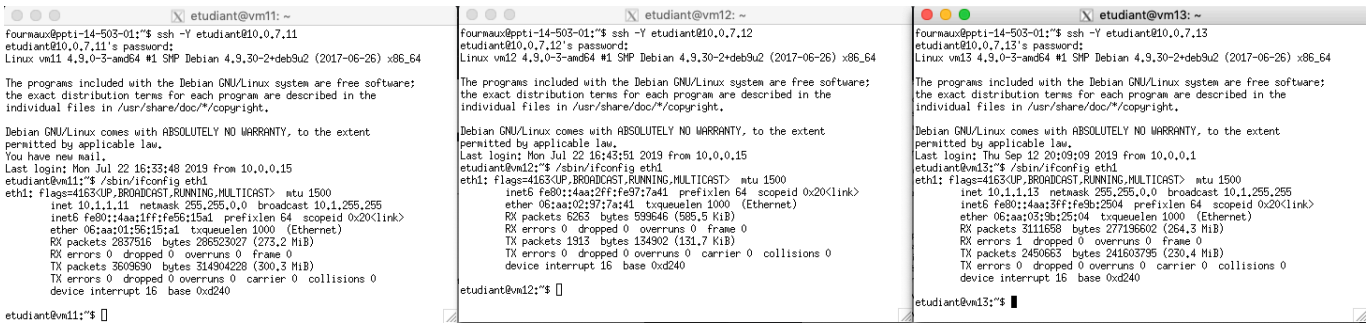


Figure 7: SSH sessions from host `ppti-14-503-01`

The virtual machines each have two (virtual) network interfaces, `eth0`, connected to the administrative network, and `eth1`, connected to the experimental network, as shown in Figure 3. You can check the configuration of each interface on each VM with the `/sbin/ifconfig` Unix command. Run this command in each of the terminal windows, and you should see information similar to that shown in Figure 7.

In the following, all the Labs will be presented with this remote access method. SSH is standard and secure.

3.4.2 Remote control of the three virtual machines via SSH from outside of PPTI

For whichever topology you create, you will be controlling the virtual machines remotely via SSH from outside but thru PPTI PC **N**. Your login on all VMs is the word `etudiant`, and your tutor will provide you the password. To open each session, open a terminal window on your personal computer, and type the following at the command line to go thru PPTI PC **N**:

- for the "client" vm**N**1 (window 1):
  - `ssh <myPPTIlogin>@ssh.ufr-info-p6.jussieu.fr`
  - then `ssh <myPPTIlogin>@ppti-14-503-`**N** (if **N**<0, write `0N`)
  - then `ssh etudiant@10.0.7.`**N**1
- for the "monitor" vm**N**2 (window 2):
  - `ssh <myPPTIlogin>@ssh.ufr-info-p6.jussieu.fr`
  - then `ssh <myPPTIlogin>@ppti-14-503-`**N** (if **N**<0, write `0N`)
  - then `ssh etudiant@10.0.7.`**N**2
- for the "server" vm**N**3 (window 3):
  - `ssh <myPPTIlogin>@ssh.ufr-info-p6.jussieu.fr`
  - then `ssh <myPPTIlogin>@ppti-14-503-`**N** (if **N**<0, write `0N`)
  - then `ssh etudiant@10.0.7.`**N**3

The virtual machines each have two (virtual) network interfaces, `eth0`, connected to the administrative network, and `eth1`, connected to the experimental network, as shown in Figure 3. You can check the configuration of each interface on each VM with the `/sbin/ifconfig` Unix command. Run this command in each of the terminal windows, and you should see information similar to that shown in Figure 7.

In the following, all the Labs will be accessible with this alternate remote access method. Just beware to not collide with other student using the same VM.

3.4.3 Starting a capture

The standard case we study is that of a client running on one VM, the `wireshark` (graphical) or `tshark` (textual) sniffer softwares running on a second VM, and a server on a third VM. The sniffer's VM is connected via its `eth1` interface to a LAN on which the client-server traffic is transiting, and it is able to capture all traffic on that LAN by putting the interface into "promiscuous" mode. Under most operating systems, a network interface card only picks up traffic that is explicitly addressed to its MAC address or to the broadcast MAC address. This reduces processing overhead for the host and it also provides a measure of privacy for others' data that circulate on the LAN. For a networked application such as `wireshark` or `tshark` to put an interface into promiscuous mode, it must be run with rights (already configured).

Graphical capture: `wireshark`

You will know that you have started `wireshark` on the monitor VM because a `wireshark` window will appear on your PC, thanks to X11 redirection. Initially, you should see a new windows as shown in Figure 8. Select the "Capture" menu and click on "Interfaces ...". As shown in Figure 9, a window presenting the different interfaces of the monitor VM will appear. Click on the Options button for interface `eth1`. (Note: this interface only supports IPv6, and so you will see an IPv6 address for this interface rather than an IPv4 address.)

Initiate the capture by clicking on the Start button. In order to see traffic start to appear, ping the server VM from the client VM by typing `ping 10.`**N**`.1.`**N**3 at the Unix command line in the terminal window that you have open for the client VM. Figure 10 shows a ping underway in the client-side terminal window and the ICMP Echo Request packets appearing in the `wireshark` window from the monitor VM.

To stop the pings, type `Ctrl-C` in the client VM terminal window. To stop the capture, click the Stop button in `wireshark`'s capture window.

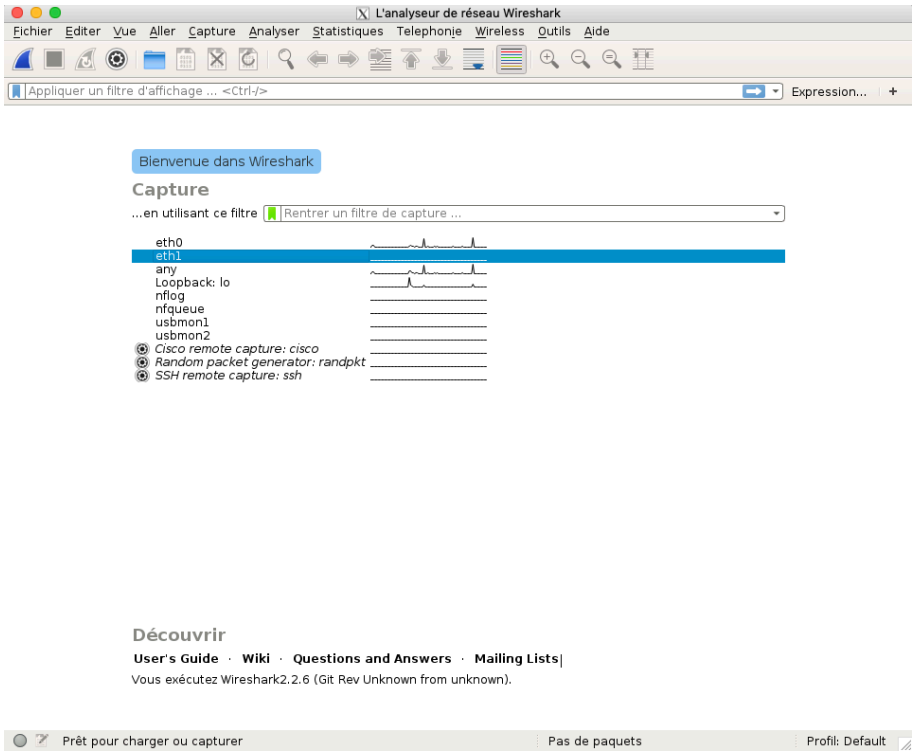


Figure 8: Starting `wireshark`

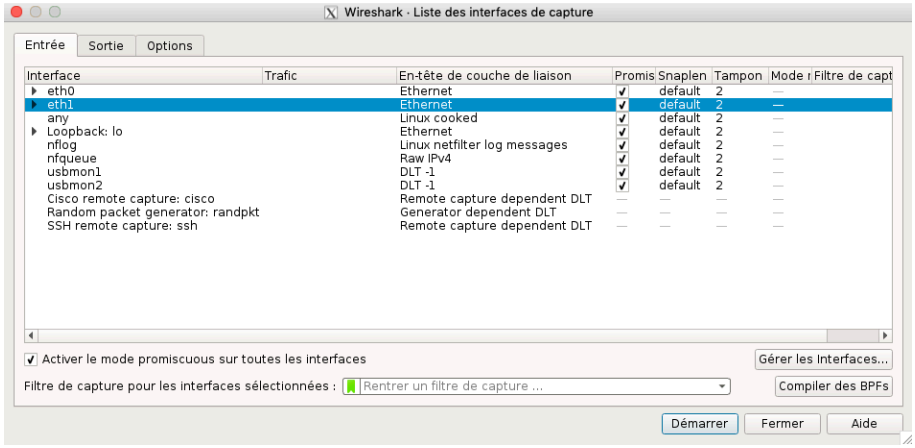


Figure 9: The list of interfaces on the monitor VM

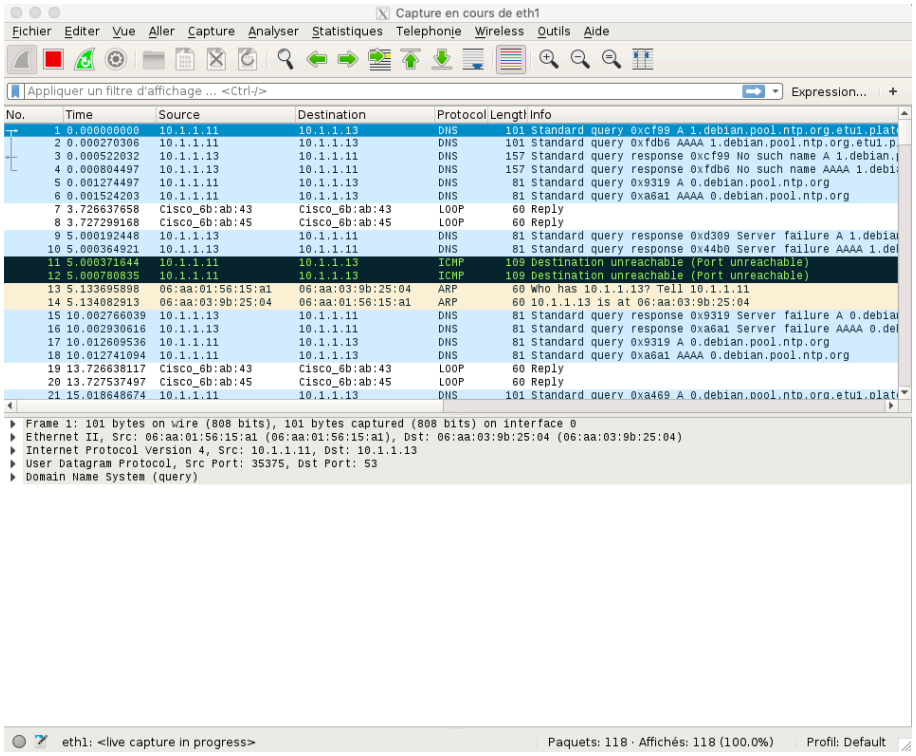


Figure 10: Capturing `ping` traffic (ICMP packets)

Textual capture: `tshark`

Run `tshark -i eth1 -w <myCapture>` on the monitor VM and a new capture of the traffic on interface `eth1` will start immediately.

In order to see traffic start to appear, ping the server VM from the client VM by typing `ping 10.`**N**`.1.`**N**`3` at the Unix command line in the terminal window that you have open for the client VM.

To stop the pings, type `Ctrl-C` in the client VM terminal window.

Then to stop `tshark` and the capture, type `Ctrl-C` in the monitor VM terminal window.

The capture is saved in the file `<myCapture>` on the monitor VM. To be able to use it you must retrieve it on your personal computer:

- repatriate this capture to your PPTI account: `scp <myCapture> <myPPTIlogin>@10.0.0.`**N**`:`
- then on your personal computer: `scp <myPPTIlogin>@ssh.ufr-info-p6.jussieu.fr:<maCapture> .`

You can then locally load the trace captured on the platform with `wireshark -r <myCapture>` then graphically analyze it.

4 Traffic capture and analysis exercise

This exercise uses the simple topology described above, with two hosts exchanging traffic directly over a LAN (see Figure 5). The topology will already be in place for you for this lab section, and an Apache web server will already be running on the server VM. You will start a Firefox web browser (or use the `wget` command) on the client VM and, with the `wireshark` (or `tshark`) sniffer on the monitor VM, you will observe and analyze the web traffic that is generated as you navigate with the browser.

4.1 Capturing web traffic

From PPTI PC **N**:

- If you have not yet connected to the three VMs, do so now, using the login `etudiant` and the password provided by your tutor.

  - access in the "client" vm**N**1 (window 1) with SSH to `etudiant@10.0.7.`**N**1 (use `-Y` if you want to run `firefox`)
  - access in the "monitor" vm**N**2 (window 2) with SSH to `etudiant@10.0.7.`**N**2 (use `-Y` if you want to run `wireshark`)
  - access in the "server" vm**N**3 (window 3) with SSH to `etudiant@10.0.7.`**N**3

- Verify that the web server is running on `10.0.7.`**N**3 (window 3)
  - to double-check the server's IP address, type: `/sbin/ifconfig eth1`
  - look for the web server process, type: `ps aux | grep apache`

- Run the sniffer on `10.0.7.`**N**2 (window 2)
  - if you run a graphical sniffer, type: `wireshark`, then initiate the capture on interface `eth1`, as previously described
  - if you run textual, type `tshark -i eth1 -w <myCapture>`

- Start a HTTP request on `10.0.7.`**N**1 (window 1)
  - run the client, type: `firefox` and open the page la page `http://10.`**N**`.1.`**N**3 (verify there is no proxy use)
  - or, in text mode: `wget -p --no-proxy http://10.`**N**`.1.`**N**3

- The captured trace shown in the `wireshark` window should be similar to that seen in Figure 11, with a browser window on the left, `wireshark` in the middle, and information about the web server appearing in the terminal window on the right. Remember to terminate the capture.

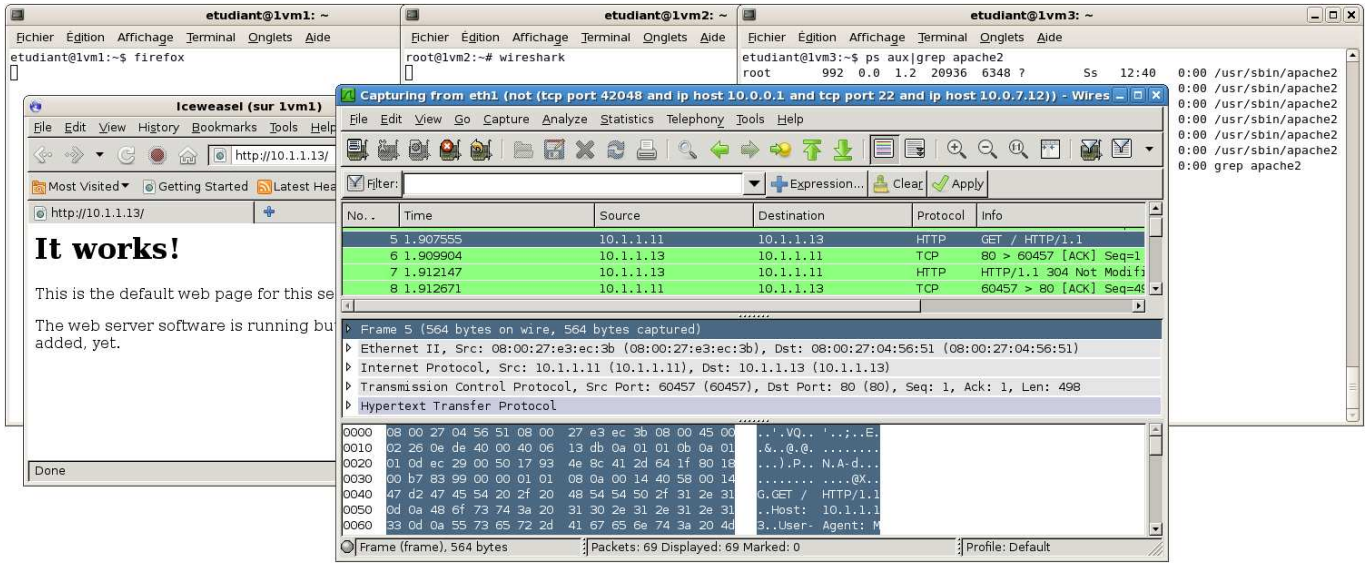


Figure 11: Capturing web traffic

4.2 Analyzing the captured web traffic

Keep `wireshark` open and, with the trace that the sniffer is displaying:

1. Select all frames containing HTTP data.
2. Describe what you observe. If there are multiple connections, explain their relationship.
3. On the client side, use the browser to observe the HTML source code of the page that it is displaying. On the server side, try to find this page in the file system of the VM. Then, from the monitor's perspective, verify in the trace that this content has in fact crossed the network.

5 Before leaving the room

- If you have saved some traces on the monitor VM, do not forget to transfer them back to your PPTI user account. Type the following command on a local terminal of the PPTI host: `scp etudiant@10.0.7.N2:<trace> <dest>`
- Before closing your connections to the virtual machines, be sure to restore them to the state in which you found them, removing any modifications you might have made.

Ethernet frame structure

```
.......+-48bits-+-48bits-+16b-+- - - - -+......
.(Pre.)| dest.  | source |Eth.| data    |(CRC).
.      | addr.  | addr.  |type|         |     .
.......+--------+--------+----+- - - - -+......

Some Eth. types: 0x0200 = XEROX PUP
                 0x0800 = DoD Internet (IPv4)
                 0x0806 = ARP
                 0x8035 = RARP
                 0x86DD = Internet Protocol Version 6 (IPv6)
```

ARP frame structure

```
+16b-+-16b-+8b+8b+16b+-lgHW-+-lgP-+-lgHW-+-lgP-+
|HW  |Proto|HW|P |Op.|AddrHW|AddrP|AddrHW|addrP|
|Type|Type |lg|lg|   |Source|Src. |Rcpt. |Rcpt.|
+----+-----+--+--+---+------+-----+------+-----+

Some types: 0x0001 = Ethernet
            0x0800 = DoD Internet (IPv4)
Operations (Op.): 0x0001 = Request
                  0x0002 = Answer
```

IPv4 packet structure

```
<-----------------32bits----------------------->
<-4b->      <--8bits--->  <--------16bits-------->
+-----+-----+-----------+-----------------------+
| Ver | IHL | TOS       | Total Length (Bytes)  |
+-----+-----+-----------+--+--------------------+
| Identifier              |Fl| FO                 |
+-----------+-----------+--+--------------------+
| TTL       | Protocol  | Checksum (header)     |
+-----------+-----------+-----------------------+
| Source Address                                |
+-----------------------------------------------+
| Destination Address                           |
+-----------------------------------------------+
...     Options
+-----------------------------------------------+
...     Data
+-----------------------------------------------+

Ver = IP Version
IHL = IP Header Length (in 32 bit words)
TOS = Type Of Service
Fl (3 first bits) = Fragmentation Bits
 * 1st = Reserved
 * 2nd = Don't Fragment
 * 3rd = More Fragments
FO (13 following bits) = Fragment Offset
TTL = Time to Live
Some Protocols:                  8 = EGP
        1 = ICMP                11 = GLOUP
        4 = IP (encapsulation)  17 = UDP
        6 = TCP                 46 = RSVP
```

ICMP packet structure

```
<-----------------32bits----------------------->
+-----------+-----------+-----------------------+
|  Type     |  Code     | Checksum (full msg.)  |
+-----------+-----------+-----------------------+
|  Variable (Type dependent)                    |
+-----------------------------------------------+
...     Original Datagram + 8 Bytes
+-----------------------------------------------+

Some ICMP types:   8 = Echo Request
                   0 = Echo Reply
                  11 = TTL Expired
                  12 = Bad IP Header
```

TCP segment structure

```
<-----------------32bits----------------------->
<-4b->         <-6bits-><--------16bits-------->
+-----------------------+-----------------------+
| Source Port           | Destination Port      |
+-----------------------+-----------------------+
| Sequence Number                               |
+-----------------------------------------------+
| Acknowledgment Number                         |
+-----+-------+---------+-----------------------+
| THL |       |  Flag   |  Windows Size         |
+-----+-------+---------+-----------------------+
| Checksum (header)     |  Urgent Pointer       |
+-----------------------+-----------------------+
...      Options
+-----------------------------------------------+
...      Data
+-----------------------------------------------+

THL = TCP Header Length (in 32 bit words)
Flags coded on 6 bits from left to right
 * 1st = Urgent Data (URG)
 * 2nd = Acknowledgement (ACK)
 * 3rd = Flushing data (PSH)
 * 4th = Reset (RST)
 * 5th = Synchronisation (SYN)
 * 6th = Termination (FIN)
Options = list encoded with
 * 1 byte set to 00 = End of Options
 * 1 byte set to 01 = NOP (No OPeration)
 * several bytes with TLV encoding
   T = One Type Byte:
     2 Max Segment Size
     3 Window size increase
     4 Selective acknowledgement
     8 Timestamp
   L = One byte for the total length of the option
   V = value of the option (L-2 bytes)
```

UDP datagram structure

```
<-----------------32bits----------------------->
+-----------------------+-----------------------+
| Port Source           | Port Destination      |
+-----------------------+-----------------------+
| UDP Length            | Checksum (message)    |
+-----------------------+-----------------------+
...      Data
+-----------------------------------------------+
```

Port number associated services

ftp-data	20/tcp		
ftp	21/tcp		
ssh	22/tcp	ssh	22/udp
telnet	23/tcp		
smtp	25/tcp		
domain	53/tcp	domain	53/udp
		tftp	69/udp
www	80/tcp	www	80/udp
kerberos	88/tcp	kerberos	88/udp
pop-3	110/tcp	pop-3	110/udp
		snmp	161/udp
		snmp-trap	162/udp