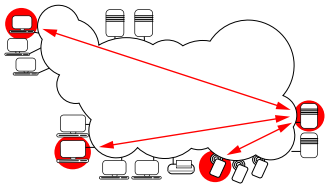
ComNet - Lab n°2

Application Layer (1): Telnet, SSH, FTP, TFTP and Web

In Lab n°1, you learned how to use the networking testbed and you used it to generate, capture, and analyze a simple application layer trace involving web traffic. In Lab n°2, you will explore the application layer in much greater detail, examining the following protocols: TELNET, SSH, FTP, SFTP, TFTP, and HTTP. For each, you will generate real traffic and capture it and analyze it using the `wireshark/tshark` tool. You will also use the IETF RFC (the formal specification) of one of these protocols, FTP, to help you understand its traffic.



1 Warm-ups exercices (without computer assistance)

1. What is an application layer protocol?

2. Which network-enabled programs do you regularly use? Do you know which are the application layer protocols they use?

3. Which communications model do classical applications tend to employ? Which terms describe the roles of the participants in such communications?

4. Describe the main categories of network-enabled applications. For each category, indicate typical needs in terms of bandwidth, the ability to tolerate variation in available bandwidth, sensitivity to losses, and delay constraints.

2 Remote login

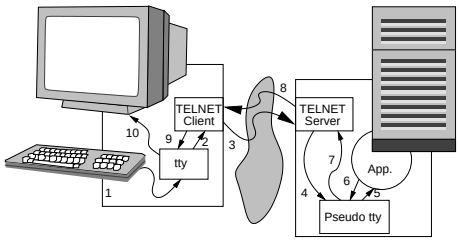
2.1 Reminders

1. What is the purpose of remote login applications?

2. What kinds of information do these applications exchange?

3. What impediments must such applications overcome? Give examples.

4. What kinds of services do these applications require from lower networking layers?



2.2 The TELNET protocol

TELNET enables remote login. It is one of the oldest protocols in the TCP/IP protocol suite. (RFC 854 was published in 1983.) It was designed to work with a large preexisting base of machines and terminal types, and so it allows for negotiation of many optional parameters in order to accommodate needs at both ends. Typically, these parameters are negotiated at the start of a TELNET session.

To overcome the potential heterogeneity across hosts involved in a networked exchange, TELNET defines a standard service called NVT (Network Virtual Terminal), with a character encoding that is close to the 7-bit ASCII encoding for printable characters. NVT's encoding is also used for conveying text in other protocols of the TCP/IP suite.

TELNET uses the TCP transport layer in order to ensure reliability. The TCP port number **23** has been allocated for the server side of TCP connections. TELNET uses *in-band control*, meaning that parameter negotiation and other aspects of signalling take place in the same connection as is used for data transfer. Like most of the early TCP/IP protocols, it lacks security mechanisms. For instance, it does not ensure confidentiality.

In following, you will analyze the two characteristic phases of a TELNET connection: negotiation, and data exchange.

2.2.1 Capturing TELNET traffic

First, you will capture some TELNET traffic in order to understand its basic characteristics. Topology 1 (with client and server on the same LAN, as described in Lab n°1) is available to you on the networking testbed. Capture traffic using `wireshark` or `tshark`, as follows:

- From PPTI PC **N**, connect to the three corresponding VMs of the testbed from three separate terminal windows
  - 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 GUI client)
  - 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 TELNET is running on the server VM (window 3)

  - look for the TELNET server process, which will either be `telnetd` (stand-alone) or `inetd` (a larger network services process, which runs TELNET if configured to do so in `/etc/inetd.conf`): `ps aux | grep telnetd` (or `inetd`)
  - query the VM about its network interfaces, look at the information regarding the interface to the experimental LAN (`/sbin/ifconfig eth1`), and verify that its IP address is `10.`**N**`.1.`**N**3

- 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 TELNET client on the client VM (window 1)
  - type `telnet 10.`**N**`.1.`**N**3, which should establish a TELNET connection from the client VM to the server VM
  - using the TELNET connection, log in to the server VM as `etudiant` (with the corresponding password), type a few UNIX commands, and then end the TELNET session (log off with the `exit` command)
- Observe the trace captured in the `wireshark` window
- **Filter the traffic to keep only TELNET** (filter = `telnet`). Save the filtered trace for later reuse. Keep the application running in order to conduct the following analysis.

2.2.2 Analyzing TELNET negotiation traffic

Parameter negotiation principally takes place at the beginning of a TELNET connection. Exchanges consist of commands that can be sent either from the client to the server or in the other direction. A special one-byte **escape character**, `IAC`=0xff, meaning "interpret as command", is used to signal that the following bytes are part of a command rather than data. (Note that `IAC` is not to be confused with the ASCII escape character 0x1b, which can be part of the regular data stream. Note also that if 0xff happens to appear in the data stream, it needs to be doubled, as 0xff 0xff, in order to be interpreted unambiguously as data by the receiver.) A one-byte **command code** follows `IAC`. Four of the commands have to do with negotiating options, and they are each followed by a one-byte **option code** (see table). These commands are: `WILL`=0xfb (indicates that an entity is prepared to apply the specified option, or confirms that the entity has applied it) `WON'T`=0xfc (the entity will not apply the option), `DO`=0xfd (requests the other entity to apply the specified option), `DON'T`=0xfe (requests them not to). For example:

`IAC`, `DO`, `24`

A typical negotiation might consist of an entity signalling `WILL` to indicate that they are ready to apply an option and the other entity responding by `DO`, followed by more details regarding how to apply the option. These details are specified in **sub-options**, which are signalled by the following sequence: `IAC`, `SB`=0xfa, the byte indicating the option, a byte indicating either "value required" (0x01) or "value supplied" (0x00), several bytes containing the value, `IAC`, `SE`=0xf0. For example:

`IAC`, `SB`, `24`, `0`, `'V'`, `'T'`, `'2'`, `'2'`, `'0'`, `IAC`, `SE`

$(Value)_{10}$	**Option name**
1	Echo
3	Suppress Go Ahead
5	Status
6	Timing Mark
24	Terminal Type
31	Negotiate About Window Size
32	Terminal Speed
33	Remote Flow Control
34	Linemode
35	X Display Location
36	Environment variables
39	New Environment Option
...	

Using `wireshark`, focus your attention only on the hexadecimal values in the trace, and try to discern which parameters are being negotiated.

1. Describe which options and sub-options appear in the exchanges.

2. How much time does the negotiation phase take?

2.2.3 Analyzing the TELNET data exchange

Look beyond the first few negotiation frames.

1. At what point does data start to appear in the TELNET connection?

2. What traffic do you see on the network that relates to the user typing characters on the keyboard as part of the TELNET session?

3. What do you think of the effectiveness of the protocol?

4. What degree of interactivity does the protocol support?

5. What application level information is conveyed in the data flow?

2.2.4 Wide area TELNET trace (*optional... you can tackle this on your own if you are well ahead of the other students)*

From a record containing one hour of long distance traffic between the *Lawrence Berkeley Laboratory* and the rest of the world in January 1994, find examples of TELNET communications.

These traces, initially in `tcpdump` format (the standard trace format that `wireshark/tshark` uses as well), have been "anonymized", meaning that IP addresses have been renumbered and packet contents have been deleted. The remaining information has been recoded in a purely ASCII format.[1].

Here's an excerpt:

```
8.430376 22 21 23 33281 1
8.437539 3 4 3930 119 47
8.442644 4 3 119 3930 15
8.454895 26 11 4890 23 1
8.459398 5 2 14037 23 0
8.469004 4 23 4464 119 512
```

The first column contains a timestamp relative to the beginning of the capture (expressed in seconds), the two following columns are the source and destination addresses renumbered in order of appearance, then there are the port numbers, and finally the data size (in bytes).

Load the trace `tme2-lbl.txt.gz`, either from the directory `/Infos/lmd/2022/master/ue/MU4IN001-2022oct`, or from the web page `http://www-npa.lip6.fr/~fourmaux/Traces/labV8.html`, to a **local** directory (eg. `/tmp`).[2] Then, rather

[1] The trace lbl-pkt-4 ran from 14:00 to 15:00 on Friday, January 21, 1994 (times are Pacific Standard Time) and captured 1.3 million TCP packets, the dropping about 0.0007 of the total. The tracing was done on the Ethernet DMZ network over which flows all traffic into or out of the Lawrence Berkeley Laboratory, located in Berkeley, California. The raw trace was made using tcpdump on a Sun Sparcstation using the BPF kernel packet filter. Timestamps have microsecond precision. The trace has been "sanitized" using the sanitize scripts. This means that the host IP addresses have been renumbered, and all packet contents removed. The trace was made by Vern Paxson (vern@ee.lbl.gov). The trace may be freely redistributed.

[2] The size of the trace is quite large. If you put it in your home directory, which is not on the individual PC that you are using but rather is mounted via NFS from the PPTI's servers, you will get very poor response time and risk putting a strain on the system.

than using `wireshark/tshark`, which is unable to process this ASCII trace format, use standard UNIX file processing tools (`awk`, `perl`, `sed`, ...) to isolate a TELNET stream and identify its typical characteristics. **If you are not yet familiar with UNIX file processing tools:** We expect you to learn on your own how to use one of these tools, taking advantage of resources available over the web and texts in the library. Your tutor is available to answer your questions that relate to computer networking, but might not be familiar with the given tool that you choose to employ.

1. Sketch a chronogram of some of the TELNET exchanges included in the trace. What can you say about their level of interactivity?
2. Is the monitor that is capturing the trace close to the sender?
3. Can you infer anything about the type of information that is exchanged?

2.2.5 Without the testbed...

If you have difficulty accessing the networking testbed, or you would simply prefer to work from another machine, you can download the trace `tme2-tel.dmp` (similar to the one previously captured) either from the directory `/Infos/lmd/2022/master/ue/MU4IN001-2022oct`, or from the web page `http://www-npa.lip6.fr/~fourmaux/Traces/labV8.html`, and then analyze it with `wireshark` (without needing administrator rights).

2.3 The SSH protocol

These days, people typically use SSH for remote login, rather than TELNET, because it provides security mechanisms: authentication, confidentiality, and the integrity of communications. The TCP port number **22** is reserved for SSH servers to receive connections.

People take advantage of SSH's security mechanisms for much more than just remote login. An SSH connection can be used as a secure transport layer for applications. You can, for example, create an SSH connection between your residential host and the university's access server, and redirect all the traffic between a local client application at home and and a remote server in the computer center of the university. It is possible to multiplex multiple application streams over a single SSH connection.

In the following exercise, you will study SSH by attempting to generate the same user level interactions as you did for TELNET.

2.3.1 Capturing SSH traffic

The aim of this third traffic capture is to understand SSH. On the networking testbed, again using Topology 1 (client and server on the same LAN), capture SSH traffic using `wireshark/tshark`:

- From PPTI PC **N**, if you do not already have three terminal windows open, connected to the client, monitor, and server VMs, establish these connections now.
- Verify that the SSH server (`sshd`) is running on `10.0.7.`**N**3 (window 3)
- Start the capture by running the sniffer on interface `eth1` of host `10.0.7.`**N**2 (window 2)
- Start an SSH client on `10.0.7.`**N**1 (window 1)
  - invoke the client at the command line, establishing a connection to the server by typing: `ssh 10.`**N**`.1.`**N**3 (this is the IPv4 address of the server on the experimental LAN)
  - log in as `etudiant`, with the corresponding password, and then type the same UNIX commands that you ran previously during the TELNET traffic captures
- View the `wireshark` window to see the traffic captured
- **Filter the trace to keep only the SSH traffic** (filter = `ssh`). Save the filtered trace for later reuse. Do not quit the `ssh` session, so that you can continue to use it during the following analysis.

2.3.2 Analyzing the SSH exchange

1. What do you observe at the beginning of the exchange?

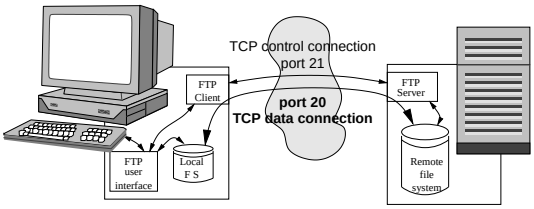
2. While you have kept the user level interaction identical to the ones for TELNET, what differences do you notice between SSH and these protocols at the application data level?

2.3.3 Wide area SSH trace (Optional. . . you can tackle this on your own if you are well ahead of the other students)

1. Identify the SSH communications in the same anonymized trace from Lawrence Berkeley Laboratory that you examined for the optional TELNET exercises, `tme2-lbl.txt.gz`.

2.3.4 Without the testbed. . .

If you have difficulty accessing the networking testbed, or you would simply prefer to work from another machine, you can download the trace `tme2-ssh.dmp` (similar to the one previously captured) either from the directory `/Infos/lmd/2022/master/ue/MU4IN001-2022oct`, or from the web page `http://www-npa.lip6.fr/~fourmaux/Traces/labV8.html`, then analyze it with `wireshark` (without needing administrator rights).



3 File Transfer

3.1 The FTP protocol

3.1.1 Studying RFC 959 (without computer assistance)

In order to understand application-level protocols in these labs, we have so far taken the purely bottom-up approach of looking at actual application traffic. For our examination of FTP, we start top-down by reading RFC 959, the official standard.[3] Start by downloading the document:

- start a browser and go to the web page `http://www.rfc-editor.org/`
- click on RFC SEARCH and search for "FTP"
- select **RFC 959** from the search results

Open this document and take a quick look at its contents. Then answer the following questions:

1. What can you say about the form of the document? And regarding its structure, what are the different sections of the document?

2. Describe FTP's communication architecture. What does it mean when we say that the information flow control is "out of band"?

3. What are the different commands available to the client?

4. Can you name the different types of errors that can be signalled in FTP? How is error information communicated?

[3] Standards relating to the Internet are published by the **IETF** *(Internet Engineering Task Force)* in documents called **RFC**s. RFC means *Request For Comments*, implying that the documents are still subject to comment, and therefore to change, which many of them indeed are.

3.1.2 Capturing FTP traffic

The aim of this fourth traffic capture is to understand FTP. On the networking testbed, again using Topology 1 (client and server on the same LAN), capture FTP traffic using `wireshark/tshark`:

- From PPTI PC **N**, if you do not already have three terminal windows open, connected to the client, monitor, and server VMs, establish these connections now.
- Verify that the FTP server (`ftpd`) is running on `10.0.7.`**N**3 (window 3)
- Start the capture by running the sniffer on interface `eth1` of host `10.0.7.`**N**2 (window 2)
- Start an FTP client on `10.0.7.`**N**1 (window 1)
  - invoke the client at the command line, establishing a connection to the server by typing: `ftp 10.`**N**`.1.`**N**3 (this is the IPv4 address of the server on the experimental LAN)
  - log in as `etudiant`, with the corresponding password
  - navigate through the FTP server's file system, using the FTP client's `pwd`, `cd`, and `dir` commands
  - choose a file and download it to the client machine, using the FTP client's `get` command
  - terminate the exchange, using the FTP client's `quit` command
- Observe the trace captured in the `wireshark` window
- **Filter the traffic to keep only FTP** (filter = `ftp or ftp-data`). Save the filtered trace for later reuse. Keep the application running in order to conduct the following analysis.

3.1.3 Analyzing the FTP control connection

Figure out how messages exchanged over network, on the FTP control connection, correspond with what is shown to the user of the FTP client application.

1. By definition, who initiates the communication between client and server? Can we observe this in the trace?

2. Which FTP client command does the user employ in order to identify him- or herself? What corresponding activity do you see in the trace?

3. Which FTP client command authenticates the user? Does the password appear in clear on the network?

4. What is the purpose of the command that follows authentication?

5. What purpose does the `PORT` command serve? Analyze its parameters. Why is it sent at this point in the exchange?

6. The `LIST` command is used to list the files that are in the current directory on the server. Why is it followed by two messages sent by the server?

7. What other commands that you can observe? What are they?

8. When do file transfers take place?

3.1.4 Analyzing the FTP data connection

1. What information is exchanged over the data connection?

2. Which port numbers are used for the data?

3. What do you observe regarding synchronization of messages between the connection control and the data connection?

3.1.5 Wide area FTP trace (Optional... you can tackle this on your own if you are well ahead of the other students)

1. Identify the FTP communications in the same anonymized trace from Lawrence Berkeley Laboratory that you examined for the optional TELNET, RLOGIN, and SSH exercises, `tme2-lbl.txt.gz`. Be sure to identify both the `FTP` and `FTP-DATA` connections.
2. Draw the chronogram.
3. What can you say about the interactivity of FTP as compared to TELNET?

3.1.6 Without the testbed...

If you have difficulty accessing the networking testbed, or you would simply prefer to work from another machine, you can download the trace `tme2-ftp.dmp` (similar to the one previously captured) either from the directory `/Infos/lmd/2022/master/ue/MU4IN001-2022oct`, or from the web page `http://www-npa.lip6.fr/~fourmaux/Traces/labV8.html`, then analyze it with `wireshark` software (without needing administrator rights).

3.2 The SCP and SFTP protocols

There are several protocols for secure transmission of files. The `scp` application is client software that is part of the SSH suite, enabling transfer of files to an `sshd` server. From the user's perspective, `scp` functions similarly to `rcp` (the UNIX remote file copy command, which is part of the `r*` family of commands). There is also an `sftp` client that functions similarly to FTP, while connecting through an SSH tunnel to its dedicated counterpart, the `sftp-server`.

3.2.1 Capturing SCP and SFTP traffic

The aim of this fifth traffic capture is to understand secure file transfer. On the networking testbed, again using Topology 1 (client and server on the same LAN), capture either SCP or SFTP traffic using `wireshark/tshark`:

- From PPTI PC **N**, if you do not already have three terminal windows open, connected to the client, monitor, and server VMs, establish these connections now.
- Verify that the SCP server (`sshd`) or SFTP server (`sftp-server`) is running on `10.0.7.`**N**3 (window 3)
- Start the capture by running the sniffer on interface `eth1` of host `10.0.7.`**N**2 (window 2)

- Start an SCP or SFTP client on `10.0.7.N1` (window 1) and retrieve from the server the file previously transferred as part of the FTP exercise

  SCP type `scp etudaint@10.N.1.N3:<remote_file> <local_file>` and then authenticate yourself

  SFTP type `sftp 10.N.1.N3` and then authenticate yourself and type the same commands you previously executed during the FTP traffic capture

- Observe the trace being captured in the `wireshark` window
- **Filter traffic to keep only the secure file transfer traffic** (filter = `ssh`). Save the filtered trace for later reuse. Keep the application running in order to conduct the following analysis.

3.2.2 Analyzing the SCP or SFTP traffic exchange

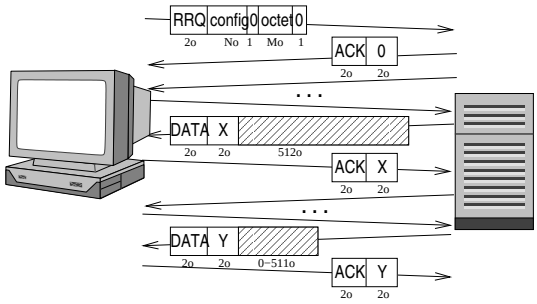
1. Recap how the SCP and SFTP protocols work.

2. Can you establish a correspondence between the commands that are typed by the user and the frames that are exchanged over the network?

3. What differences do you see with FTP?

3.2.3 Without the testbed...

If you have difficulty accessing the networking testbed, or you would simply prefer to work from another machine, you can download the trace `tme2-scp.dmp` (similar to the one previously captured) either from the directory `/Infos/lmd/2022/master/ue/MU4IN001-2022oct`, or from the web page `http://www-npa.lip6.fr/~fourmaux/Traces/labV8.html`, and then analyze it with `wireshark` (without needing administrator rights).



3.3 The TFTP protocol

3.3.1 Capturing TFTP traffic

The aim of this last traffic capture is to understand TFTP (Trivial File Transfer Protocol). On the networking testbed, again using Topology 1 (client and server on the same LAN), capture TFTP traffic using `wireshark/tshark`:

- From PPTI PC **N**, if you do not already have three terminal windows open, connected to the client, monitor, and server VMs, establish these connections now.
- Verify that the TFTP server (`tftpd`) is running on `10.0.7.N3` and the directory is configured for the transfers (window 3). Look for the files inside this directory.
- Start the capture by running the sniffer on interface `eth1` of host `10.0.7.N2` (window 2)
- Start a TFTP on `10.0.7.N1` (window 1)
  - invoke the client at the command line by typing: `tftp 10.N.1.N3` (this is the IPv4 address of the server on the experimental LAN)

  - download a file to the server with the `get` command
  - terminate the exchange with the `quit` command

- View the `wireshark` window to see the traffic captured
- **Filter the trace to keep only the TFTP traffic** (filter = `tftp`). Save the filtered trace for later reuse. Do not quite the `tftp` session, so that you can continue to use it during the following analysis.

3.3.2 Analyzing the TFTP exchange

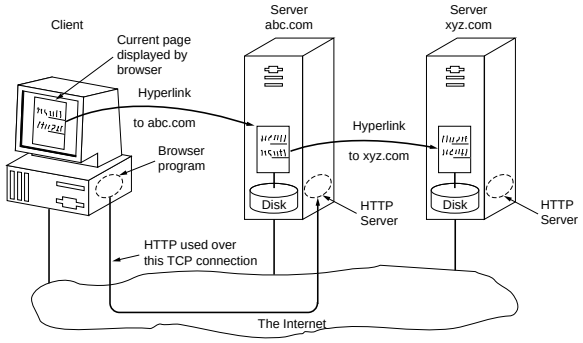
1. Recap how the TFTP protocol works.

2. Can you establish a correspondence between the commands that are typed by the user and the frames that are exchanged over the network?

3. What differences do you see with FTP?

3.3.3 Without the testbed...

If you have difficulty accessing the networking testbed, or you would simply prefer to work from another machine, you can download the trace `tme2-tft.dmp` (similar to the one previously captured) either from the directory `/Infos/lmd/2022/master/ue/MU4IN001-2022oct`, or from the web page `http://www-npa.lip6.fr/~fourmaux/Traces/labV8.html`, and then analyze it with `wireshark` (without needing administrator rights).



4 Web traffic

4.1 Exercices (without computer assistance)

1. Describe the steps involved in downloading a webpage. Suppose you want to retrieve a page composed of an HTML file and two small-sized objects stored on the same server. Excluding the transmission time for these objects, specify the delay necessary to obtain the page. Illustrate your answers with chronograms.

2. What optimizations introduced with HTTP 1.1 do today's webservers employ to reduce exchange latency? For the example provided above, illustrate your answer with a chronogram.

3. Another way to reduce the response time is through cache memory. Describe where this mechanism is employed and for which kinds of objects it makes sense.

4.2 The HTTP protocol

4.2.1 Capturing HTTP traffic

The aim of this last traffic capture is to understand HTTP. On the networking testbed, again using Topology 1 (client and server on the same LAN), capture HTTP traffic using `wireshark/tshark`:

- From PPTI PC **N**, if you do not already have three terminal windows open, connected to the client, monitor, and server VMs, establish these connections now.
- Verify that the HTTP server (`apache2`) is running on `10.0.7.`**N**3 (window 3)
- Start the capture by running the sniffer on interface `eth1` of host `10.0.7.`**N**2 (window 2)
- Use an HTTP client 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
  - or, in text mode: `wget -p --no-proxy http://10.`**N**`.1.`**N**3
- View the `wireshark` window to see the traffic captured
- **Filter the trace to keep only the HTTP traffic** (filter = `http`). Save the filtered trace for later reuse. Keep the client running so that you can continue to use it during the following analysis.

4.2.2 Analyzing the HTTP request

1. Are you able to observe the means by which the page described in our example, above, is retrieved?

2. What parameters are negotiated between client and server?

3. What optimizations are implemented to accelerate the download of web pages?

4. Can you display the webpage based on the trace?

4.2.3 Making a new web page (optional. . . tackle this only if you are ahead of the other students)

To learn more about HTTP, you can study traces based upon additional webpages that you create, to be served from the `etudiant` account of the server VM.

To enable webpages to be served from this account, create the `public_html` directory with appropriate rights in the account's home directory. Use the following UNIX command: `cd ; mkdir ~/public_html ; chmod 755 ~/public_html`

In the `public_html` directory, you will place files that the browser will be able to access at the following URL: `http://10.`**N**`.1.`**N**`3/~etudiant/`.

Create a very simple web page composed of:

- 1 very simple HTML file (if you are not already familiar with basic HTML, you can easily find examples on the web)
- 3 small images (you can transfer these via SCP from the PPTI host to the `public_html` directory)

Repeat the traffic capture from section 4.2.1 by using the browser to access the new webpage that you just created. Then answer the questions from section 4.2.2 for this new capture.

4.2.4 Without the testbed...

If you have difficulty accessing the networking testbed, or you would simply prefer to work from another machine, you can download the trace `tme2-htt.dmp` (similar to the one previously captured) either from the directory `/Infos/lmd/2022/master/ue/MU4IN001-2022oct`, or from the web page `http://www-npa.lip6.fr/~fourmaux/Traces/labV8.html`, and then analyze it with `wireshark` (without needing administrator rights).

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.