Networking tools and application layer protocols

RES 841

October 2013

Introduction

This first lab will help you discover the environment and the tools that we will use in most of the subsequent labs. Then, you will practice with these tools by examining an playing with application-level protocols. Depending on your skill with Unix environments, this lab may be quite long, it is however necessary to understand well its core for the rest of the course. If you do not have enough time during the session, you can either come back to the lab room, or download the virtual machine image (ask Claude Chaudet for the link). You will have to install Virtual Box, a free virtualization player that works on most environments (Windows, Linux, Mac OS X).

The goals of this lab are as follows:

- You will get used to the lab environment (virtualized network). You will use this environment during most of the subsequent labs. You will also get acquainted to the Unix networking toolbox, including network packet analyzers (tcpdump and wireshark).
- You will look at a few application layer protocols and be able to see what are the classical ways to implement such a protocol. For instance, most messages sent to a server, correct or incorrect, trigger the emission of a response code.
- You will notice the encapsulation principle (a message from the application layer is sent to the transport layer, which adds a header before sending it to the network layer, etc.).
- You will learn to use a packet analyzer to understand the behavior of an applicative protocol.

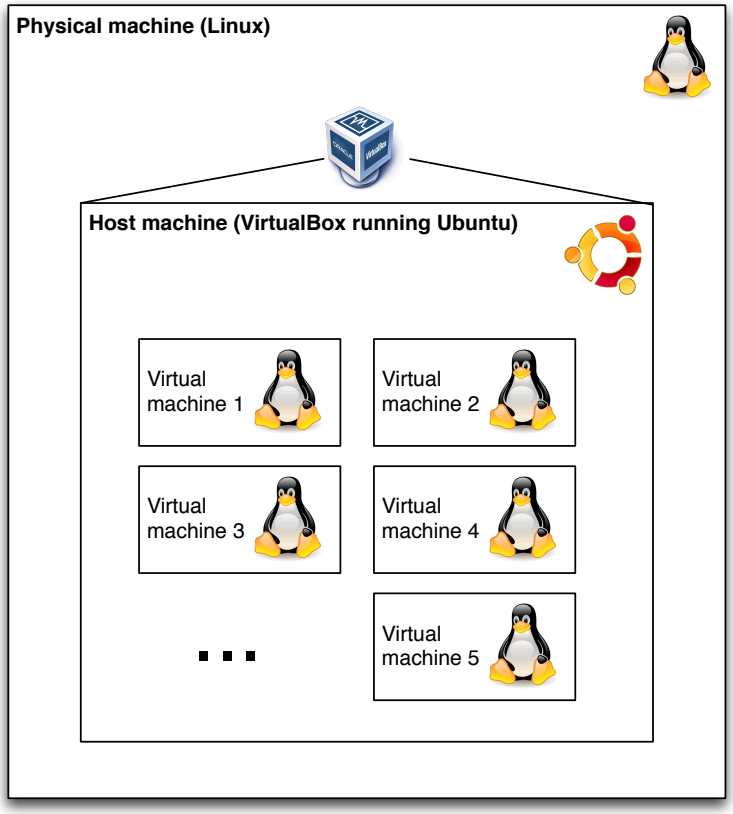
1 Starting the virtual environment

As the lab machines are shared machines connected to the school network, you are not administrator of these machine. You will not be able to work directly on the school network, but all labs will be performed in a virtualized environment. This environment comprises a few servers and network interconnection elements. In this isolated virtual network, you will have full access to the machines and will be able to use administrator tools.

Let us start by launching the virtualized environment:

Start VMWare player by going to the ”Applications” menu (right click on the desktop if you are under xfce, use the ”Activities” menu if you are under Gnome). Then go to the ”VM catalog” submenu and select ”Networking Labs” item. You may have to click on ”Play Virtual Machine” and wait until another Linux environment starts and brings you on a new desktop.

This environment will be called, in the rest of the labs, the *Host* machine. You can switch to full screen, you will not need the initial environment again. The situation is represented on the picture below. Your physical machine runs the host machine on which we installed the required tools to create multiple lightweight virtual machines on which you have administrator rights.



Inside the host machine, you will create a series of new virtual machines, interconnected through the studied network. To create these machines, open a terminal (*Applications* menu, *Accessories* sub-menu, select *Terminal*), get into the TP_1 directory (`cd TP_1`) then type `lstart`.

Warning : getting inside the correct directory before typing `lstart` *is very important. The* `lstart` *script examines the contents of the current directory to create the machines. If you happen to launch machines in the wrong directory, you can stop the machines with* `lcrash`*, then type* `lclean` *to clean up resources (disk space etc.) and restart..*

Four command line windows will open. Two of them will give you access to a virtual machine called `PC1` as administrator, two other to a machine called `PC2`. The two machines are interconnected by a network that hosts a few services.

Networking toolbox

For performance issues (virtual machines share the real machine's memory and thus have a limited capacity), you will only use command line inside the virtual environment (on PC1 and PC2). You will at least have access to the following commands, that you can try:

- `ifconfig` lists the network interfaces of a machine. On PC1 and PC2 you will see that you have two network interfaces (i.e. network cards). `eth0` is the main communication interface, using the Ethernet technology and `lo` is a "false" interface (virtual) that allows a computer to discuss with itself. This interface, called *loopback* is present on every operating system and always has the IP address `127.0.0.1`. It allows to launch a client and a server programs on the same machine and to test the code as if the were on a true IP network. It is very useful to the programmer, to test a service, or when no internet connection is available. You do not need to even have a physical network card to play with this interface.

`ifconfig` gives you the following information on every network interface:

  - The MAC address of the network interface (`HWaddr`, displayed in hexadecimal fomat)
  - The IPv4 address of the network card (`inet addr`) and its IPv6 address(es) (`inet6 addr`, displayed in hexadecimal format)
  - The broadcast address of the local network (`Bcast`) that targets *all*Â machines on the sub-network (i.e. all machines not too far away)

- `ping` allows you to test if a machine (identified by its name or IP address) is alive. It sends a message and waits for the answer, and thus allows to measure the round-trip-time delay (the delay for the answer to come back). Be careful, some machines are configured to ignore these messages and do not send back a reply.

  In some configurations, you can send a request to the broadcast address mentioned above. All machines will then try to answer you, which allows you to identify the machines present on the same network (if every machine answers, you get their IP addresses). Be careful, though some (most) machines are configured *not* to answer these solicitations for security (obfuscation). To send a request to a *broadcast* address, you have to add the `-b` option to the command (`ping -b`). Try to identify the machines present on your network from PC1 or PC2.

- `nslookup` and `dig` allow you to query a DNS database to find the matching between a machine's name (e.g. pc1.res101.telecom-paristech.fr) and its IP address (e.g. 10.0.1.1). This query is, most of the time, performed by the system itself. `dig` gives you the full DNS reply, while `nslookup`, whose usage is deprecated, will summarize this information to make it more readable.

You will manipulate these commands in this lab. In addition, a few files give you additional information on the network configuration. You may disply the contents of these files using the `cat` or `less` commands. In particular:

- `/etc/hostname` displays the name of your machine
- `/etc/resolv.conf` specifies the IP addresses of the DNS servers and the domain name of the network to which your machine is connected. The complete name of your machine over the Internet is composed by the name in `/etc/hostname` followed by this domain (e.g. `PC1.res101.telecom-paristech.fr`). The `search` line in this file also indicates the domain name that will be appended to an incomplete FQDN. For example, when you type `ping pc1`, your system will send a request to `pc1.res101.telecom-paristech.fr`.
- `/etc/services` lists the set of standard TCP / UDP ports to access services, with their names. A port is the number that identifies a service on a machine, which allows the kernel to send the traffic coming from the network to the correct process. For instance, using DNS query tools, note the address of `ftp.res101.telecom-paristech.fr` and `www.res101.telecom-paristech.fr`. What can you notice? Most common port numbers are standardized (it is a common usage, not a standard), therefore we will often find a web server on port 80 (http), the SMTP server on port 25, etc. Nothing forbids to disregard these habits and to use a number that you choose yourself, even though some usages are discouraged.
- `/etc/hosts` contains a list of correspondences between IP addresses and machines names, for common use. If this file is often empty, it may be filled with static entries to avoid interrogating the DNS server every time a request to some machines is emitted. However, this is not without risk, as this file has the priority over the DNS servers. When a machine changes names / IP addresses, your system will not notice this modification.

2 Client-Server dialog

In a classical networking environment [1], a client talks with a server using a set of pre-defined messages to get a service. Displaying a static web page usually requires the following steps:

[1] classical here means non-peer-to-peer to some extent

1. Localize the server with its name or IP address. If the name is used, a DNS query is necessary to get the corresponding IP address.
2. Open the dialog with the server (TCP requires a connection initialization phase for example)
3. Request for a given page
4. Reception of the page contents
5. End of the communication (TCP also requires to tear down the connection when the application is finished for example)

The following exercises consist in talking, by hand, with three servers. The first one implements the HTTP protocol, used to get web pages; the second one uses SMTP for sending e-mails and the last one uses FTP, dedicated to large files downloads.

2.1 HTTP

HTTP (*Hyper Text Transfer Protocol*) is an application level protocol dedicated to fetch web pages. Its most recent specification, described by RFC 2616 is available on the web[2]. For this exercice, we will only use a very limited part of HTTP. Let us start by fetching a simple page with the `telnet` command. This command is used to open a dialog with a server on a specified port and leaves, once the connection has been initiated, the opportunity to the user to discuss with the server by hand.

1. From PC1, display the web page served by `www.res101.telecom-paristech.fr` using the command-line browser `lynx` : `lynx http://www`. Note that you can use the short name of the machine (`www`) or its complete name (`www.res101.telecom-paristech.fr`). The server home page is displayed. You can close the browser by pressing the "Q" key and acknowledging.
2. In a terminal window on PC1 or on PC2, type `telnet www.res101.telecom-paristech.fr 80`
3. Once connected, enter `GET / HTTP/1.1` followed by return, then `Host: www.res101.telecom-paristech.fr` followed by *two* times return. This command asks to the server `www.res101.telecom-paristech.fr` to send you its home page `/`.
4. The text that is displayed is the home page of the site, in HTML language. The browser, at this stage, interprets this code to display the page in a user-friendly form.

2.2 Traffic capture and analysis

Let us now repeat the previous exercise while inserting into the network a spy program that will capture the data traffic. There are multiple programs capable of capturing the traffic, such as `tcpdump`, or `wireshark` (previously called `ethereal`). These tools require an administrator access to the machine for security reasons that we will see later. Therefore, they will not work on the lab machines outside the virtualized environment.

As the virtual machines (PC1 and PC2) do not have a graphical user interface, you will have to capture the traffic on these machines with a command line tool, store it into a PCAP file (which is the standard format for packet capture files) and open it on the host machine to get a graphical display.

- To capture the traffic coming out of PC1, use one of the two windows on this computer and type the command `tcpdump -i eth0 -s 0 -w /hosthome/capture.pcap port 80 or port 53`. This launches the `tcpdump` capture program, to which you give the instruction to listen to everything that passes on ports 80 (HTTP) and 53 (DNS) on the network interface `eth0` (option `-i`) and to save it in the `/hosthome/capture.pcap` file.

[2] `http://www.ietf.org/rfc/rfc2616.txt`

- In the other PC1 window, launch `lynx` and get the previous web page. Once the page is displayed, you can stop `tcpdump` by pressing `Control-C` in its window. `tcpdump` will tell you it captured a few packets (about a dozen).
- On the host machine, launch `wireshark` and open the `capture.pcap` file in the home directory of the `bci` user.You can display the exchange between the browser and the server.
  - In the displayed list, you can notice three types of packets (see the *protocol* column). Some packets correspond to the initial DNS query, some other are marked `TCP` and correspond to the setup and tear down of the connection, and some packets belong to the HTTP exchange. Note that for a simple HTTP exchange, a lot of packets are exchanged besides the applicative protocol ones.
  - Select an HTTP packet in the list. The second part of the window displays the full header of the packets and its contents ; the third part displays the whole packet contents, in hexadecimal and ASCII notations. In the second part you can open the different headers to see their details, decoded. Clicking on a header field will select the corresponding part in the lower frame.
  - By examining this headers list and their relative position in the packet below, you can notice encapsulation. The application (the browser e.g.) builds the message that you can display by selecting the Hypertext Transfer Protocol section. It passes it to the transport layer (TCP) that adds a header containing (among others) the source and destination port. This header is added before the application data. TCP then passes the packet to IP that adds network layer headers, then Ethernet and finally the physical layer headers are added. The reverse operation (decapsulation) is performed at the destination.

    You can note that before filling a header, the relevant layer needs to know all the relevant information. So before filling the IP header, the IP address of the destination needs to be known, hence the DNS exchange happening before the packet is formed.
  - Wireshark also has a set of presentation tools that allow you to analyze a complex traffic. For instance, you can follow a complete (TCP) exchange by right-clicking on a packet and selecting "Follow TCP stream". A new window is opened that displays the full exchange, decoded. When you return on the main window, only a few packets are displayed, as Wireshark automatically filtered the packets that belong to the exchange. The filter expression is displayed in the text box above the the packets list and you can remove this filter (using the *clear* button) to display again the whole capture file contents. Wireshark filters are a powerful tool and several tutorials are available online.
  - Now select the `Flow Graph` command in the `Statistics` menu. In the following dialog, select the following options: `All packets` ; `General Flow` and `Standard source / destination address`. You then get a time graph of the communication.
  - Close the time graph and select the `IO Graph` command in the `Statistics` menu. A new window is opened. You need to change the X-axis resultion by changing the (`tick interval`) parameter. A $1\,ms$ resolution will be precise enough. For the Y-axis you can choose `bytes / tick` and you can change the first graph display style to `FBar`. You will get a timely graph of the flow throughput.

2.3 SMTP

Now, let us analyze the SMTP protocol. SMTP servers usually work on port 25. In this lab the SMTP sever address is `smtp.res101.telecom-paristech.fr`. The applicative protocol to send an e-mail behaves as follows:

- Client opens a connection on port 25
- The server, if the connection is successful, sends back a positive return code (220), followed by some additional information.
- Client then sends the `HELO` command, followed by the client machine name (usually obtained by `hostname`).
- Server answers by a 250 return code, gives its name and says it accepts the exchange.

- Client specifies the sender e-mail address using the `MAIL FROM:` command, followed by the complete sender address (including the @).
- Server answers with a positive return code : 250.
- Client then specifies the destination address with the `RCPT TO:` command. Here, the recipient should be `guest@res101.telecom-paristech.fr`. Server acknowledges again.
- Client enters the DATA command, followed immediately y return. Then, the text of the message is sent, line per line and the input is ended with a single dot (`.`) on a line.
- The message is then enqueued and the server sends back the message ID.
- The client ends the connection with a `QUIT` command ; the server answers with a 221 return code.

This message exchange is fully specified in the RFC 821[3], available on the Web.

1. Capture the corresponding traffic with `tcpdump` and save it in a file in the `/hosthome/` directory.
2. Realize with `telnet` this exhcange
3. Try typing non-existent commands in the middle of the exchange and notice the server answer.
4. To read the e-mail on PC1 or PC2, you can launch the `pine` program who will ask for a password. The password is `guest`. You can select *Message Index* to see the message you just sent. Quit `pine`, with the "Q" key and acknowledge.
5. Open the trace under Wireshark and analyze the message exchange between the client and the server.

2.4 FTP

The third and last protocol that we will study in this lab is FTP. It is described by RFC 959[4] and is dedicated to file transfers. HTTP can perform file transfers, but FTP is more efficient for the server on large files, as it allows to perform each file transfer on a separate port. It avoids blocking the command port this way.

1. You can capture the FTP exchange with `tcpdump`Â by listening, first, to the port 21. You will realize, here, two captures in two separate files: one on PC1 and one on PC2.
2. Open a telnet connection on port 21 of the server `ftp.res101.telecom-paristech.fr`.
3. The server sends you a welcome message, followed by a return code 220, followed again by some additional information.
4. The client is then expected to identify himself. Enter the `USER` command, followed by the `guest`) username to the server that shall answer a 331 return code it it accepts the connection.
5. The server expects the client to send a password, which you can specify with the `PASS` command, followed by the user password (which is `guest` here). Server answers with a 230 return code. The user then accesses the server file system and is able to send commands like in an Unix command line (with a reduced set of commands, though).
6. A first command will display the path, i.e. the location where you are in the file system. This command is identical to the Unix version: `PWD`.

[3] `http://www.ietf.org/rfc/rfc821.txt`
[4] `http://www.ietf.org/rfc/rfc959.txt`

7. Once you know where you are, you may want to list the files in the current directory. Type the `LIST` command. You will receive an error message indicating that no data connection could be established. Indeed, FTP uses, to send such results, a *second* connection on a different port number. To find this port number, **which changes for every command**, type the `PASV` command. You will obtain 6 numbers between brackets. The 4 first numbers are the IP address of the server and the two last ones represent the port number on which the *next* data transfer will occur.

8. To compute the port number, take these two numbers, multiply the first one by 256 and add the second one (you have a calculator in the *Applications* sub-menu of the *Applications* menu). You shall obtain a number between 0 and 65536. Use another command line window and connect on the FTP server using `telnet` on the computed port number ; you may connect from another machine (PC1 or PC2) if you lack command line windows.

9. In the first command window (the one that is connected on port 21), type again the LIST command, you should obtain in the second window the list of the files in the second window. The first window displays the return code that should be 226.

10. To move in the remote filesystem, you can use the `CWD` command (*Change Working Directory*), followed by the name of the targeted directory, described as an Unix path. Go to the `test` directory, you will get the 250 return code in case of success.

11. You can now get the `TP.txt` file with the `RETR` command, followed by the file name. Be careful, the contents of the file will be sent on another connection, you will need to type a `PASV` command again. Get the `README` file this way.

12. Once the transfer is finished, close the connection with the `QUIT` command.

13. Open the trace file captured on PC1 with Wireshark. Find the line that corresponds to the PASS command. What can you infer on the risks related to the use of capture files by regular users?

14. Open the trace file coming from PC2. What can you notice? Does it modify your opinion on the use of capture files by non-administrators?

15. Can you see, in Wireshark, the files list (answer to a LIST command) or a file transfer? Explain your answer.

3 Acting as a server

The purpose of this exercise is to play the role of a server by answering the requests of a web browser. To do this, you will use the `netcat` command, which is the equivalent of `telnet` on a server side.

1. On PC1, type the command `nc -l -p12345`. The `-l`Â option indicates that netcat shall *listen* for incoming connections and the `-p` option specifies the port on which the server will listen. Use `netstat` and find the entry corresponding to your "server", comparing the connections on PC1 and PC2.

2. Type in a window on PC2, `telnet pc1 12345`. Type a few words in the `telnet` and `nc` windows.

3. Make sure that `nc` is still open on PC1, otherwise launch it again. Open `lynx` on PC2 using the following address: `http://pc1:12345`. Comment on the result on both sides.

4. Type in the `nc`Â window the following commands, finishing by a line break (without any space at the beginning of a line). Observe the browser reaction at each step:

```
HTTP/1.1 200 OK
Content-Location: index.html.fr
Accept-Ranges: bytes
Content-Length: 42
```

```
Content-Type: text/html
Content-Language: fr

<html><body><h1>Bravo !</h1></body></html>
```

5. Quit `nc` by pressing `Control-C`.
6. Congratulations, you just played the role of a web server. You can notice that the response that the serve answer contains more than the simple code of the HTML page.