Computer networks. Experimental Study 2nd Laboratory Practice
HTTP and DNS protocols.

DTe
Departamento de
Tecnología Electrónica

Experimental study

The experimental study of this practice consists of five parts. In each one of them all the steps that the student must carry out are described. If you have any questions, consult the teacher in charge of the practical session. In the case of not completing all parts of the experimental study, before leaving the laboratory you must perform the point 76.

(DO NOT TURN ON THE PC UNTIL INSTRUCTED TO DO SO)

Previous steps

1. Make sure you are connected to the ETSII network.
2. Turn on your PC, restore (if instructed to do so by your teacher) and boot Windows 7. Disable the firewall just as you did in steps 1 to 4 of the first lab session.
3. Disconnect your PC from the ETSII network and connect it to the laboratory's Intranet, specifically to the SWITCH_EUROPA (in G1.31) or SWITCH_SUDAMERICA (in G1.33),similar to how you did steps 40 through 44 in the first lab, just this time you will be connected to a SWITCH and not a HUB**. If there are no free ports in the appropriate SWITCH, please inform the teacher.**
4. How can you be sure you have physical level connectivity between your PC and the SWITCH?

5. On the laboratory intranet, LAB_DTE, the PCs connected to it require a specific TCP / IP configuration that is obtained automatically from a DHCP server. Carry out the appropriate steps to automatically obtain the new TCP / IP configuration of your PC as you did in section 46 of the previous lab. Write the IP address of your PC.
6. Use a command that allows you to check that you have network level connectivity (exchange of R_PDUs) with your border router (Default Gateway). You learned how to do these types of checks in the first practice. Don't continue until you have connectivity.

First Part: Wireshark analyzer

As we already know from the previous lab, Wireshark is a cross-platform free software tool (Windows, Linux and MacOS) that you can download for free from http://www.wireshark.org.
Wireshark is a program that, executed on a end system, is capable of capturing all the network traffic received or sent by itself. It is a very useful tool, as it not only captures traffic but is also capable of analyzing it, showing the user detailed information on the protocols of each of the layers (from the data link layer to the application layer). That is why it is called a protocol analyzer. It is a very complete and complex tool, so we are going to focus on the basic handling of its interface and, at the same time, we will use some fundamental commands and option.

7. Start Wireshark
8. Wireshark is able to use DNS services to always show us host and domain names, instead of showing us the equivalent IP addresses, in the usual xxx.xxx.xxx.xxx numeric format. This feature will be very useful to us in this lab. Enter "Edit" → "Preferences", click "Name Resolution" on the left panel, enable the option "Resolve Network (IP) addresses". Wireshark is also capable of showing us, instead of the TCP and UDP port numbers, the name of the protocol that usually uses that port number. In this specific practice we are not interested in enabling this Wireshark functionality. Enter "Edit" → "Preferences", click "Name Resolution" on the left panel, disable the option "Resolve Transport Name" and click “Ok to apply the changes”
9. Start capturing the traffic coming in and out of your Ethernet network connection, just like you did in the first lab in section 55.
10. Lets generate network traffic by opening the Mozilla Firefox browser and visiting the website http://www.redes.lab/index.html
11. You can see that Wireshark shows you, in the upper panel of its main window, the frames it has captured, the result of data traffic between the client and the web server. Stop capturing traffic when you notice that the loading of the previous page has finished.

12. As you already know from the first practice, in the list of frames we can see a lot of information about each frame, organized in columns, the ones that appear by default are:

    a) The first column is called "No." and shows us the order number in which the frames have been captured, from 1 to N.
    b) The second column is called "Time" and in it Wireshark shows us, in seconds, temporal information of the moment in which that frame was captured. By default this time is measured from the moment the first frame was captured, so in frame number 1 it is 0.000000.
    c) The "Source" column shows information about the equipment that sent the frame (or the one that sent a PDU encapsulated in that frame, it depends on how we have configured Wireshark).
    d) The "Destination" column is analogous to the previous one, showing us information about the destination computer.
    f) The "Protocol" column shows the highest level protocol information encapsulated in that frame and that Wireshark is able to analyze.
    g) The "Length" column shows the number of bytes in the frame. In the last laboratory session we will see which fields (E_PDU) it includes.
    h) The column "info" shows summary information of the protocol of higher level than Wireshark is able to analyze.

13. It is possible to remove and add columns of information to the list of frames, to adapt it to our needs. In this practice, it will be necessary to add two new columns that present information about the source and destination ports of the T_PDUs of the TCP and UDP protocols. Now we can identify the port number used to identify the client and server application process in a frame that encapsulates protocols down to the application level. To do this, you must follow these instructions:

    a) Enter "Edit" → "Preferences", and click on the "Columns" branch (under "User Interface" in the left panel).
    b) Press the "Add" button once to add a new column.
    c) Click on the text "New column" that has appeared, and edit it by typing the text "SrcPort" as the title of the new column and pressing "Enter" on the keyboard.
    d) In the "Field Type" field, select the value "Src Port (unresolved) from the drop-down list.
    e) Repeat steps b), c) and d) to create another column titled "DstPort" and having "Dest Port (unresolved)" of "Field Type".
    f) Click "OK" to close the "Preferences" window.
    g) Observe that the two new columns appear in the list of frames, on the right side (if you cannot see it, scroll to the right). Use the mouse to reorder the columns and place the two new ones in front of the "Info" column, or adjust the width of the "Info" column so that all of them are displayed on the screen without having to scroll.
14. As you know, the main Wireshark window is divided into three panels. We have already reviewed the top panel, the list of frames. The other two panels are closely related to the upper panel, as they show us information about the frame that we have selected in the list of frames.
15. The central panel, "detalles de trama", shows various information about the frame and its content, in an orderly and level structured manner.First, information on the complete frame is displayed and then information on each of the levels is displayed, starting from the link level, then network, transport and application (if all appear, which does not always happen). In each line there is a "+" on the left to display the protocol information associated with each level (once interpreted by the tool). Not all the information that appears for a certain protocol is actually part of that protocol. Sometimes Wireshark adds information that it has determined as a result of an analysis that it has carried out at the global level, in which case this information appears in square brackets []. On the other hand, not everything that appears after a "+" is necessarily a protocol. For example, Wireshark is capable of analyzing different file formats such as GIF, PNG, JPG, etc. and displays them to the right of a "+". Select with the mouse a frame that shows HTTP in the "protocol" column and note the names of the protocols that appear in the middle panel. In case of not finding any, because you have done wrongly capturing the sections 9, 10 and 11, you will need to start the capture again in Wireshark and order Firefox to reload the current page (http: //www.redes.lab/index.html), by clicking on the reload icon (the gray curled arrow to the right of the page's URL). Wait for the upload to finish and stop the capture in Wireshark. Note that pressing F5 in Firefox also reloads the current page as if clicking on the icon of the curled arrow.

16. The bottom panel, "Bytes de la trama", displays a hexadecimal and ASCII dump of the content of the selected frame. Hexadecimal data (on the left side) is presented in 16-byte rows, along with a first column that indicates the relative position (within the frame) of the first octet in the row. If the central panel "clicks" on any of the levels (or on any field within these), the bytes associated with what we

have "clicked" are highlighted with a dark background in the lower panel. The reverse also works, clicking on bytes in the lower panel and seeing in the central panel how the corresponding information field is selected. Click "frame details" under "Hipertext Transfer Protocol" to select the HTTP protocol. What information appears in ASCII in "Bytes de la trama"? Click on the "+" that appears next to "Hipertext Transfer Protocol" in "detalles de la trama", you will see the content of the HTTP_PDU. Click several times on different header lines and see how that information looks in ASCII. How are the control codes' \ r 'and' \ n "displayed in ASCII?

17. As you know, the content shown in the list of frames can be saved in a file (File → Save or clicking on 💾), which can be loaded at any other time (File → Open or clicking 📂). This can be very useful if you lack time to complete this practice, as you could take the capture home and finish the part of the experimental study that you could not finish there.

Regarding the time stamps, shown in the "Time" column of the list of frames, the first frame has by default the mark 0.000000 seconds and the rest of the marks are increasing with respect to this time value. However, it is possible to set a reference mark in any frame so that it is "zero" for all frames after it, which is useful to measure times between frames. To do this, we select the frame that we want to mark as a reference with right-click and choose "Set Time Reference (Toggle)", appearing *** REF *** in that frame and modifying the time of the following frames. If we repeat the operation, the reference of that frame is removed.

18. Note that in the list of frames there are several frames that contain PDUs of the HTTP protocol (note the value in the "Protocol" column). Specifically, you must find a frame that indicates ("Info" column) that it contains a GET request for the HTTP protocol.
19. The GET request has been issued by the client process, which is associated with a specific port on the source host ("source host"). Thanks to the SrcPort column (source port or source port) we can easily find out the port number associated with the client process.
20. Check that the numeric value of the DstPort column of the frame encapsulated by the GET is the usual value used by an HTTP protocol server process. Write it down.
21. Also check that the name of the www.redes.lab server from which we have downloaded the web page does NOT appear in the “Destination” column, but rather a different one. Write it down, we'll explain this later.
22. Now we are going to make the frame with the GET become the "time reference”. To do this, select this frame by "clicking" on it with the right mouse button and select the option "Set Time Reference (Toggle)" from the contextual menu that appears. Notice that now that frame does not have a timestamp in the "Time" column, but instead the text * REF * appears. It is possible to cancel this operation by repeating the same steps that we have taken on that frame.
23. Locate, behind the GET frame, a frame that encapsulates the HTTP response to that GET. In the "Info" column of the response, the status line "HTTP / 1.1 200 OK" or the status line "HTTP / 1.1 304 Not modified" should appear, depending on the circumstances in which the GET was generated with Mozilla Firefox .
24. Check that the same ports are used in the response as in the request, but the source port is now the destination port and vice versa.
25. Check that the same is true for the values in the "Source" and "Destination" columns.
26. Since we have made the GET frame the time reference for all the frames that follow it, it is very easy to measure the time elapsed between issuing the GET and receiving the response.
27. How much is the RTT between your PC and the web server www.redes.lab worth? Call your teacher to check the RTT value.
28. Wireshark is capable of, from any frame that contains a T_PDU of the TCP protocol (or UDP), locate all the other T_PDUs that were transmitted on the same TCP connection as it (or in the case of UDP, the T_PDUs between the same client and server using a certain UDP port pair). Thanks to this, it can show us the flow of bytes transmitted through that TCP connection by the client and server processes (or the exchange of A_PDUs in the case of UDP). Select the GET frame by "clicking" on it with the right mouse button and select the "Follow TCP Stream" tool from the contextual menu that appears. (Note: For UDP it would be "Follow UDP Stream")
29. In the "Follow TCP Stream" window we can see the bytes sent by the client process in pink and the bytes sent by the server process in purple. If the client and the server maintain a "long" dialogue through the same connection (as in persistent HTTP connections), it could be seen how the messages from the client and the server alternate.

Part Two: Display Filters

In the text box labeled "Filter:" that appears below the Wireshark icon bar, you can write a logical expression with a certain syntax, which causes only those that make the logical expression to be true to be displayed in the list of frames. This logical expression is called a "filter" and only affects what is shown in the frame list. It neither eliminates an already captured frame nor is it taken into account when deciding whether a frame should or do not capture. This is only a "display filter" and not a "capture filter".

There are many filter expressions that we can enter for each protocol recognized by Wireshark. You can see all the expressions for each protocol by "clicking" on the "Expression ..." button, but if you know the expression you can type it directly in the "Filter:" text box. Wireshark shows a help with the expressions compatible with what you have already written and even auto-completes them as you type them. If the expression is correct, it will appear with a GREEN background and if it is incorrect, the background will be RED. For Wireshark to use the filter we just wrote, it is necessary to click on "Apply", which we can see to the right of the filter. If we want to see all the frames again, just click on "Clear". Now click on "Clear" if the "Filter:" text box shows a filter.

Some basic expressions are:
a) "ip.addr == 193.1.10.1", which shows frames containing R_PDUs whose source or destination IP address is 193.1.10.1.
b) "tcp.port == 80", to show traffic with source or destination port number 80.
c) "udp.port == 53", which does the same but with UDP.
d) "ip.src" and "ip.dst" are similar to "ip.addr" but only note that the specified IP is either the source or the destination.
e) "tcp.dstport" and "tcp.srcport" are set only on the destination or source port.
f) "http", "dns", "tcp", "udp" and "icmp" are simple expressions that cause only frames that encapsulate PDUs of those protocols to be displayed.

You can build complex logical expressions by combining simple expressions with the logical operators "and", "or", and "not". For example "http or dns" captures frames that make the "combined" logical expression true. That is, those that make "http" true and also those that make "dns" true. It is also possible to use the "contains" operator that allows you to search for strings within the PDU of a protocol, for example, "http contains GET" would allow you to see all the frames that encapsulate HTTP PDUs that contain the word "GET". Another example with this operator would be "dns contains www" which would allow us to see only the frames with DNS PDUs in which the string "www" appears.

Note that if you mark a frame as a temporary reference with * REF * that frame is always displayed in the list of frames, regardless of the display filter applied.

30. If you do not have a capture made that completely fills the list of frames, make a new one generating the necessary network traffic and then stop the capture.
31. Write a simple filter, such as "http", "tcp", "udp", and so on. and apply it.
32. Notice how at the bottom edge of the Wireshark window appears the text "Packets:" indicating the total number of captured frames and the text "Displayed:" indicating the number of frames that have passed the display filter and can be seen in the list of frames.
33. Click "Clear" to clear the display filter and see all captured frames again.
34. Apply a filter that shows only network level traffic originating or going to your own IP. Notify the teacher to check the filter result.

Third part: DNS

Windows operating systems maintain a DNS cache where dns resolutions that have been made previously are saved for a certain time. The idea behind this is not having to ask the DNS server to solve requests that we have made a short period of time before.

35. Use Mozilla Firefox to load the web page http://webserver.af.lab/index.html. You will notice that it shows the same page as in section 9. This is because www.redes.lab is an alias of webserver.af.lab (that is, both are names associated with the same final system). The principal name is known as CNAME (Canonical NAME) in DNS terminology and is the one you wrote down in paragraph 21. Close the Mozilla Firefox browser.
36. Open the command Prompt ("Símbolo de sistema" on the Desktop), and type **ipconfig /displaydns** to display all entries in the DNS cache.

37. Find out, using the output from the command, the IP address of the web server www.redes.lab
38. Clear the DNS cache by running **ipconfig /flushdns** and check with **ipconfig /displaydns** that it has, indeed, been cleared.
39. Start capturing traffic with Wireshark.
40. Make sure that Firefox is closed. This is very important because the browser also saves its own cache with the most recent DNS requests, regardless of the DNS cache of the Windows Operating System.
41. Open the Mozilla Firefox browser and visit the website http://www.redes.lab/index.html
42. Stop capturing traffic when you notice that page has been loaded.
43. Using display filters try to isolate network traffic containing the DNS protocol associated with the loaded page. That is, DNS traffic in which we can see the question ("query") about the name www.redes.lab and the answer ("query response"). Use a simple filter like "dns" first and look how many frames are shown. The filter will most likely show you more than four frames with DNS traffic, so you'd better use a more selective filter that shows you only the DNS traffic associated with page loading.
44. Use the filter "dns contains redes" and you should see only four frames. These are two questions and their corresponding answers. The question we are looking for is the one that has in the "Info" field a text similar to **"Standard query 0xabcd A www.redes.lab".** Please notice that the number **0xabcd** will be a different one, but we must be sure that the Info field contains **"A"** (we are not interested in the one containing **"AAAA"**). Right-click on the frame with the question with the "A" and run the "Follow UDP Stream" tool. Close the "Follow UDP Stream" window that has just been opened. Now, you should only see the DNS traffic that interests us. The reason is that a display filter has been created when using “Follow UDP Stream”. This filter takes the advantage of the fact that a question and its answer are identified by a pair of IP addresses and a pair of UDP ports. Also, notice that the numeric code (**0xXXXX)** that appears in the DNS question is the same code that appears in the associated DNS response
45. Note that it is possible, by analyzing DNS traffic, to find out the IP of the web server www.redes.lab. Where (which field within Wireshark) can you see that information?
46. Does the IP in the previous step match with the one you noted in paragraph 37?
47. Explain which steps must be taken to determine the value of the RTT between your computer and the DNS server. Write down the value of that RTT.
48. Explain how you can find out, using Wireshark, the transport protocol used by DNS clients and servers. Ask the teacher to check your answers.

Fourth part: HTTP[1]

49. The intranet of the laboratory has links between routers with a low bandwidth, compared to those that exist today on the Internet. This causes that even with little network traffic congestion occurs in the routers of the laboratory, greatly affecting the delays appreciated by users and applications. Specifically, while on the Internet it is common for a web server to respond to a connection attempt by a client in a time much less than 250ms, it is relatively normal in the laboratory that these times exceed 250ms. Therefore, it is convenient to configure the Mozilla Firefox browser so that it knows that those delays greater than 250ms are normal and that it should not request a new connection with the web server when 250ms have elapsed without getting a response. We will configure it to wait much longer, for example 12000ms.
50. In Mozilla Firefox type "about:config" in the address bar, accept the "aviso para manazas" and you will see an advanced configuration window. Use "retry-timeout" as the search filter (below the address bar) and double-click on the search result called "network.http.connection-retry-timeout". Delete the current value (250ms),and write 12000.
51. Start a new capture with Wireshark.
52. Using Firefox, load the web page **http://www.redes.lab/lab2/tarta.html.**
53. Stop the capture a few seconds after it finishes loading the page.
54. Use Firefox to display the source code of the HTML base file How many referenced objects does the HTML page have? , Write down the number. How many GET requests should send the browser to load the whole page?

The following questions must be answered using the information obtained with Wireshark:

55. Locate the frame that encapsulates the request PDU of the base page. What version of HTTP does the browser use? In which language(s) have the browser requested the base page?

56. Locate the frame that encapsulates the PDU of the base page (this is the first response that appears in the frame listing). What status code has the web server returned for the home page? Verify if the response HTTP PDU has "body" (HTTP_UD). If the HTTP_PDU has no “body” it is probably because you had a problem with the capture. Ask your teacher to explain you how to repeat the capture by deleting the browser cache first

57. What is the size (in bytes) of the main HTML file (base page)?

58. Did the client set up a persistent connection with the server? If so, using the "Follow TCP Stream" tool, determine how many objects have been requested in this TCP connection with the server. Have all the objects referenced on the base page been requested on this connection? (Remember you wrote down the number in section 54), if not, how will request the browser the missing objects? Ask the teacher to check your answers.

59. To see the different connections established between the browser and the web server you can use a display filter that shows only the http PDUs and then sort the list of frames by the SrcPort column. How many different connections has the client established? Write them down. How can you identify each connection and differentiate it from the others? Ask the teacher to check your answers.

60. Keep using the "http" filter and sort the list of frames by the "No" column (frame number, the default ordering) to see the frames in "chronological" order, as they have been captured.

61. For each TCP connection, determine which objects have been requested by the TCP connection and the order in which they have been requested. Make a time chart showing this information.

62. Within each connection note that a new GET is not made until the response to the previous GET has arrived. Notice, however, that not receiving a GET response on a given connection, does not prevent a new GET from being sent over a different connection. This means that these two connections are working "in parallel". Analyze carefully, starting from the beginning of the capture, the GETs that are being carried out by the different connections and find out at when you find a greater number of connections working in parallel, that is, the moment of time in which there are more GET requests pending to be answered. Write down the maximum number of connections running "in parallel" that you could find, and the time at which it occurred.

63. How long has it taken for the entire page to load (including all objects but excluding DNS resolution)? Write down the value.

64. Start the capture in Wireshark. Make Firefox to reload the current page by clicking on the reload icon (the gray curled arrow to the right of the page URL). Wait a few seconds until you are sure that the reload has ended and stop the capture.

65. Look at the frames that encapsulate HTTP PDUs and check if the response messages from the server have a body (i.e.HTTP_UD).

66. What has changed in the GET request messages so that now the responses do not have a body?

67. What is the total time it took to load the full page this time? Write it down. Is it a longer or shorter time than before in step 63 Why?

68. In Mozilla Firefox you can clear the cache by simultaneously pressing CTRL+Shift+Delete and then clicking the "Limpiar ahora" button. Start a capture with Wireshark. Clear the Mozilla Firefox cache and reload the page. Stop the capture and check, using Wireshark, that the reload after you have cleared the cache is different from the reload you did previously. Do HTTP response messages now have body?

69. In Mozilla Firefox you can change the number of persistent connections in parallel. To do this, type "about:config" in the address bar, accept the "aviso para manazas" and you will see an advanced configuration window. Use the word "persistent" as the search filter (below the addressbar). Look at the preference called "network.http.max-persistent-connections-per-server" and write down the current value. Does that value match the value seen in step 62?

70. Change the current value of the "network.http.max-persistent-connections-per-server" preference by double-clicking it and entering 1 as the new value.

71. Start a new capture with Wireshark.

72. Clear the Firefox cache and reload the same page **http://www.redes.lab/lab2/tarta.html**

73. How many connections have been created with the web server? How many objects have been requested in each of them? How is each of the connections identified?

74. How long did it take to load the full page now? Compare the result with the one you wrote down in step 63. Why are they different?

75. Restore the value of the modified preferences using the "about:config" window (steps 50 and 70 by right-clicking on each preference and selecting "Restablecer".

76. Close all open windows on your PC.
Disconnect your PC from the lab's intranet.
Connect your PC to the ETSII network
Turn off your PC.
Put the LAN cord that you connected to the SWITCH EUROPA or SWITCH_SUDAMÉRICA in place again