Preparation for Lab1

Getting started with Wireshark and nslookup

Romaric Duvignau

January 26, 2023

Computer Communication Labs
with contributions and support from Hans-Martin Heyn, Roman Melnik, Ali Salehson and Marina Papatriantafilou. Some parts adapted from **Wireshark Lab from J.F. Kurose and K.W. Ross**.

CHALMERS

This document provides a preparation to the first lab about "Wireshark' and includes 3 preparation tasks to perform at home. The main command-lines are provided only for Windows and Unix-based (Linux, macOS, etc) operating systems so for other variants please refer to online documentation. The screenshots are from Wireshark Version 3.2.3, installed on a KDE-based system, your own installation might differ slightly in look.

Main Task: You'll get acquainted with Wireshark, and make some simple packet captures and observations. You will also learn using the program `nslookup`.

Prerequisite: You should be acquainted with the content covered by **Chapter 1** and **Chapter 2** in your course book (specifically **section 2.2** about HTTP protocol and **section 2.4** about the DNS protocol). It could also be helpful to have read the "Wireshark Lab" at the end of **Chapter 1** in the course book.

Organization: The first part consists of two sections: getting **starting with Wireshark** (§ 1.1) and its installation on your computer, and some easy **test tasks** (§ 1.2) to perform to warm up. The second part covers experimenting with the HTTP protocol and using the `nslookup` command. You should account for about 1-2h to cover these preparation instructions. This preparatory lab is intentionally very guided, feel free to skip some sections if you feel confident using the tools.

Contents

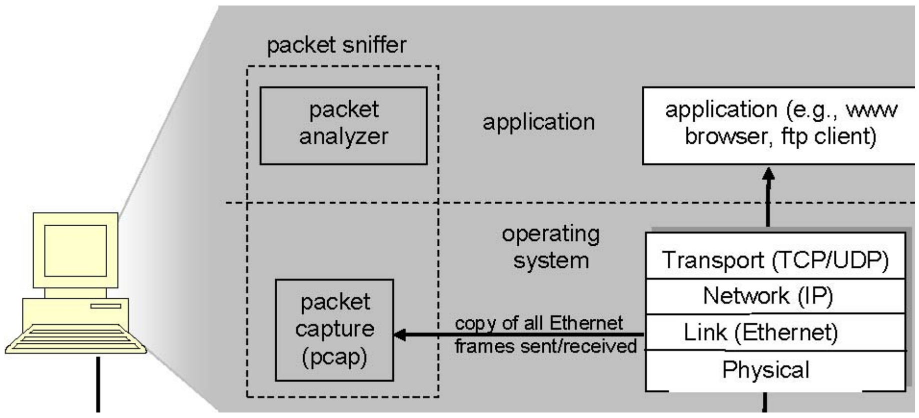


Figure 1: Packet sniffer structure.

1 Introduction to Wireshark

One's understanding of network protocols can often be greatly deepened by "seeing protocols in action" and by "playing around with protocols", observing the sequence of messages exchanged between two protocol entities, delving down into the details of protocol operation, and causing protocols to perform certain actions and then observing these actions and their consequences. This can be done in simulated scenarios or in a "real" network environment such as the Internet. In the Wireshark labs you'll be doing in this course, you'll be running various network applications in different scenarios. You'll observe the network protocols "in action", interacting and exchanging messages with protocol entities executing elsewhere in the Internet. Thus, you (and the computer that you use) will be an integral part of these "live" labs. You'll observe, and you'll learn, by doing.

The basic tool for observing the messages exchanged between executing protocol entities is called a **packet sniffer**. As the name suggests, a packet sniffer captures ("sniffs") messages being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a *copy* of packets that are sent/received from/by applications and protocols executing on your computer.

Figure 1 shows the structure of a packet sniffer. At the right side of Figure 1, there are protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on your computer. The packet sniffer, shown within the dashed rectangle in Figure 1 is an addition to the usual software in your computer, and consists of two parts. The **packet capture library** receives a copy of every link-layer frame that is sent from or received by your computer. Recall from the discussion from **Section 1.5.2** in the course book (**Figure 1.24**) that messages exchanged by higher layer protocols such as HTTP, FTP, or DNS, are transported by TCP or UDP and sent as IP packets which are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In Figure 1, the assumed physical media is an Ethernet, and so all upper-layer protocols are eventually encapsulated within Ethernet frames. Capturing all link-layer frames thus gives you all messages sent/received from/by all protocols and applications executing in your computer.

The second component of a packet sniffer is the **packet analyzer**, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must "understand" the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Then, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string "GET", "POST", or "HEAD", as shown in **Figure 2.8** in the course book.

We will be using the Wireshark packet sniffer [`http://www.wireshark.org/`] for these labs, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer). Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac computers. It's an ideal packet analyzer for our labs. Wireshark is stable, has a large user base and well-documented support that includes:

- user-guide (`http://www.wireshark.org/docs/wsug_html_chunked/`),
- man pages (`http://www.wireshark.org/docs/man-pages/`), and
- detailed FAQ (`http://www.wireshark.org/faq.html`),

along with rich functionality that includes the capability to analyze hundreds of protocols, and a well-designed user interface. It operates in computers using Ethernet, serial (PPP and SLIP), 802.11 wireless LANs, and many other link-layer technologies.

1.1 Starting with Wireshark

Downloading and installing Wireshark

In order to run Wireshark, you will need to have access to a computer that supports both Wireshark and the libpcap or WinPCap packet capture library. If it is not installed within your operating system, the libpcap software will be installed for you when you install Wireshark. See `http://www.wireshark.org/download.html` for a list of and download sites. So now, if you haven't done already, it's time to **download and install** the Wireshark software: go to `http://www.wireshark.org/download.html`, download and install the Wireshark binary for your computer using the version for your operating system. The Wireshark FAQ has a number of helpful hints and interesting tidbits of information, particularly if you have trouble installing or running Wireshark.

Running Wireshark

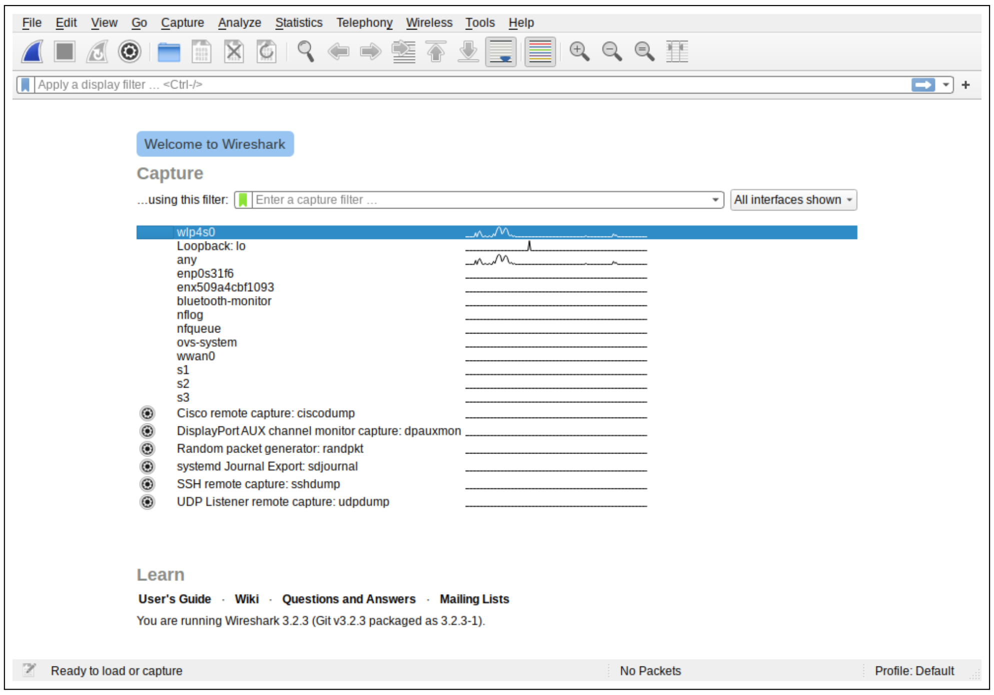
Linux

In Unix, you must run wireshark as superuser for packet capturing: `sudo wireshark`.

When you run the Wireshark program, you should get a startup screen, as shown in Figure 2.

Take a look at the upper left hand side of the screen, you'll see an "Interface list". This is the list of network interfaces on your computer. Once you choose an interface, Wireshark will capture all packets on that interface. In this example, there are several active interfaces among which a wifi interface (active interfaces show a small network traffic graph next to them).

If you now click on one of the active interfaces to start packet capture (i.e. for Wireshark to begin capturing all packets being sent to/from that interface), a screen like the one of Figure 3 will be displayed, showing information about the packets being captured. Once you start packet capture, you can stop it by using the Capture pull-down menu and selecting Stop, or simply by clicking the red square in the shortcut bar. To restart packet capture after it has been stopped, click the "blue Wireshark fin", the leftmost icon in the bar.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists

```
wlp4s0
Loopback: lo
any
enp0s31f6
enx509a4cbf1093
bluetooth-monitor
nflog
nfqueue
ovs-system
wwan0
s1
s2
s3
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
Random packet generator: randpkt
systemd Journal Export: sdjournal
SSH remote capture: sshdump
UDP Listener remote capture: udpdump
```

You are running Wireshark 3.2.3 (Git v3.2.3 packaged as 3.2.3-1).

Ready to load or capture | No Packets | Profile: Default

Figure 2: Initial Wireshark Screen.

The Wireshark GUI has five major components:

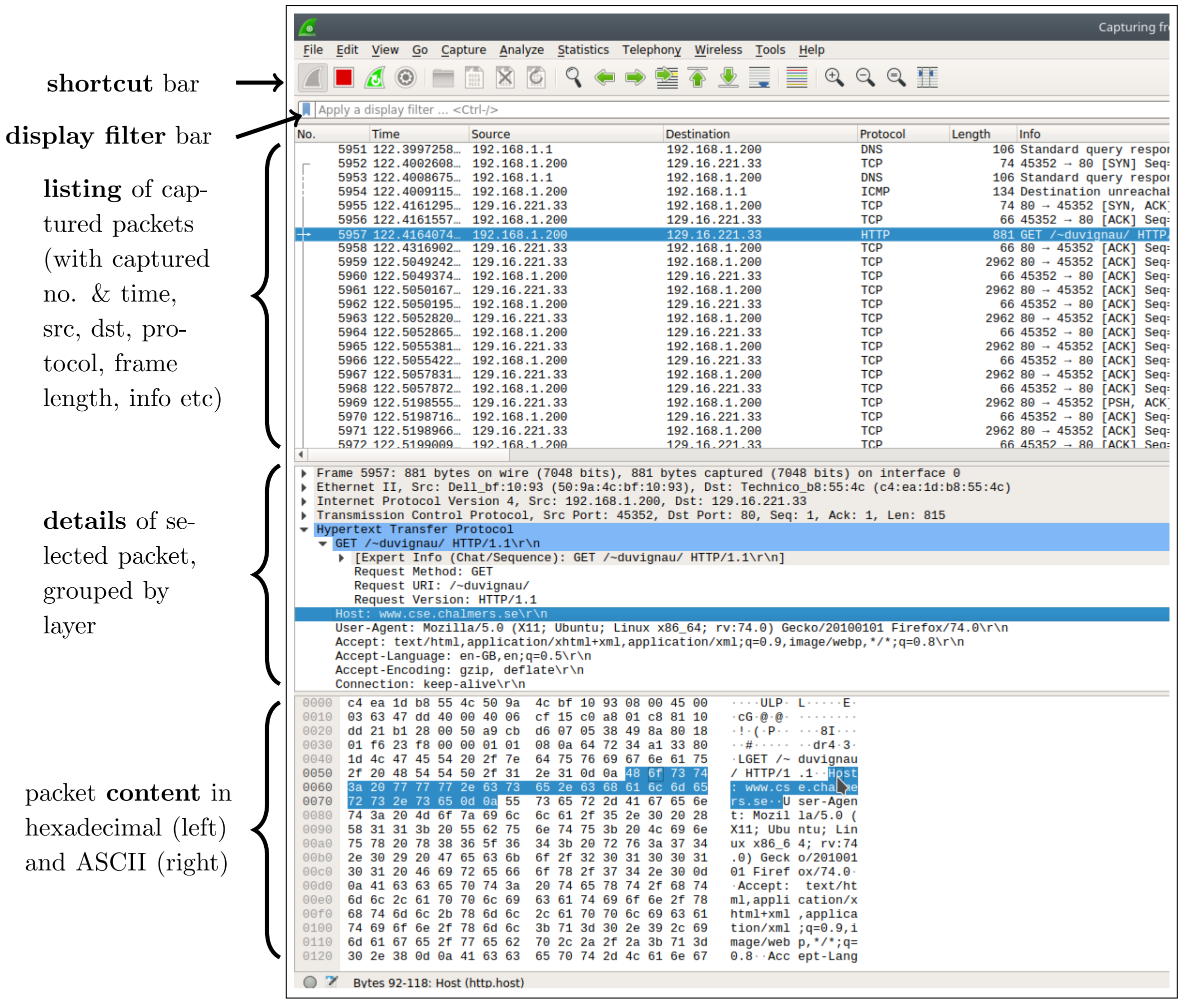
- The **command** menus are standard pull-down menus located at the top of the window. Of interest to us now are the *File* and *Capture* menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data, and exit the Wireshark application. The Capture menu allows you to begin packet capture.

- The **packet-listing pane (window)** displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is *not* a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest-level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.

- The **packet-header details pane** provides details about the packet selected (highlighted) in the packet-listing pane. These details include information about the Ethernet frame (assuming the packet was sent/received over an Ethernet interface) and IP datagram that contains this packet. The amount of Ethernet and IP-layer details displayed can be expanded by clicking on the ▶ to the left of the Ethernet frame or IP datagram line in the packet-details pane (click ▼ to minimize). If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are also provided.

Tips

To select a packet in the packet-listing, place the cursor over the packet's one-line summary in the packet-listing pane and click with the left mouse button.


shortcut bar
display filter bar
listing of captured packets (with captured no. & time, src, dst, protocol, frame length, info etc)
details of selected packet, grouped by layer
packet **content** in hexadecimal (left) and ASCII (right)
Capturing fr
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.       Time             Source          Destination     Protocol  Length Info
     5951 122.3997258…  192.168.1.1     192.168.1.200    DNS          106 Standard query respo
     5952 122.4002608…  192.168.1.200   129.16.221.33    TCP           74 45352 → 80 [SYN] Seq
     5953 122.4008675…  192.168.1.1     192.168.1.200    DNS          106 Standard query respo
     5954 122.4009115…  192.168.1.200   192.168.1.1      ICMP         134 Destination unreacha
     5955 122.4161295…  129.16.221.33   192.168.1.200    TCP           74 80 → 45352 [SYN, ACK
     5956 122.4161557…  192.168.1.200   129.16.221.33    TCP           66 45352 → 80 [ACK] Seq
     5957 122.4164074…  192.168.1.200   129.16.221.33    HTTP         881 GET /~duvignau/ HTTP
     5958 122.4316902…  129.16.221.33   192.168.1.200    TCP           66 80 → 45352 [ACK] Seq
     5959 122.5049242…  129.16.221.33   192.168.1.200    TCP         2962 80 → 45352 [ACK] Seq
     5960 122.5049374…  192.168.1.200   129.16.221.33    TCP           66 45352 → 80 [ACK] Seq
     5961 122.5050167…  129.16.221.33   192.168.1.200    TCP         2962 80 → 45352 [ACK] Seq
     5962 122.5050195…  192.168.1.200   129.16.221.33    TCP           66 45352 → 80 [ACK] Seq
     5963 122.5052820…  129.16.221.33   192.168.1.200    TCP         2962 80 → 45352 [ACK] Seq
     5964 122.5052865…  192.168.1.200   129.16.221.33    TCP           66 45352 → 80 [ACK] Seq
     5965 122.5055381…  129.16.221.33   192.168.1.200    TCP         2962 80 → 45352 [ACK] Seq
     5966 122.5055422…  192.168.1.200   129.16.221.33    TCP           66 45352 → 80 [ACK] Seq
     5967 122.5057831…  129.16.221.33   192.168.1.200    TCP         2962 80 → 45352 [ACK] Seq
     5968 122.5057872…  192.168.1.200   129.16.221.33    TCP           66 45352 → 80 [ACK] Seq
     5969 122.5198555…  129.16.221.33   192.168.1.200    TCP         2962 80 → 45352 [PSH, ACK
     5970 122.5198716…  192.168.1.200   129.16.221.33    TCP           66 45352 → 80 [ACK] Seq
     5971 122.5198966…  129.16.221.33   192.168.1.200    TCP         2962 80 → 45352 [ACK] Seq
     5972 122.5199009…  192.168.1.200   129.16.221.33    TCP           66 45352 → 80 [ACK] Seq
```

```
▸ Frame 5957: 881 bytes on wire (7048 bits), 881 bytes captured (7048 bits) on interface 0
▸ Ethernet II, Src: Dell_bf:10:93 (50:9a:4c:bf:10:93), Dst: Technico_b8:55:4c (c4:ea:1d:b8:55:4c)
▸ Internet Protocol Version 4, Src: 192.168.1.200, Dst: 129.16.221.33
▸ Transmission Control Protocol, Src Port: 45352, Dst Port: 80, Seq: 1, Ack: 1, Len: 815
▾ Hypertext Transfer Protocol
  ▾ GET /~duvignau/ HTTP/1.1\r\n
    ▸ [Expert Info (Chat/Sequence): GET /~duvignau/ HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /~duvignau/
      Request Version: HTTP/1.1
    Host: www.cse.chalmers.se\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:74.0) Gecko/20100101 Firefox/74.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-GB,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
```

```
0000  c4 ea 1d b8 55 4c 50 9a  4c bf 10 93 08 00 45 00   ····ULP· L·····E·
0010  03 63 47 dd 40 00 40 06  cf 15 c0 a8 01 c8 81 10   ·cG·@·@· ········
0020  dd 21 b1 28 00 50 a9 cb  d6 07 05 38 49 8a 80 18   ·!·(·P·· ···8I···
0030  01 f6 23 f8 00 00 01 01  08 0a 64 72 34 a1 33 80   ··#····· ··dr4·3·
0040  1d 4c 47 45 54 20 2f 7e  64 75 76 69 67 6e 61 75   ·LGET /~ duvignau
0050  2f 20 48 54 54 50 2f 31  2e 31 0d 0a 48 6f 73 74   / HTTP/1 .1··Host
0060  3a 20 77 77 77 2e 63 73  65 2e 63 68 61 6c 6d 65   : www.cs e.chalme
0070  72 73 2e 73 65 0d 0a 55  73 65 72 2d 41 67 65 6e   rs.se··U ser-Agen
0080  74 3a 20 4d 6f 7a 69 6c  6c 61 2f 35 2e 30 20 28   t: Mozil la/5.0 (
0090  58 31 31 3b 20 55 62 75  6e 74 75 3b 20 4c 69 6e   X11; Ubu ntu; Lin
00a0  75 78 20 78 38 36 5f 36  34 3b 20 72 76 3a 37 34   ux x86_6 4; rv:74
00b0  2e 30 29 20 47 65 63 6b  6f 2f 32 30 31 30 30 31   .0) Geck o/201001
00c0  30 31 20 46 69 72 65 66  6f 78 2f 37 34 2e 30 0d   01 Firef ox/74.0·
00d0  0a 41 63 63 65 70 74 3a  20 74 65 78 74 2f 68 74   ·Accept:  text/ht
00e0  6d 6c 2c 61 70 70 6c 69  63 61 74 69 6f 6e 2f 78   ml,appli cation/x
00f0  68 74 6d 6c 2b 78 6d 6c  2c 61 70 70 6c 69 63 61   html+xml ,applica
0100  74 69 6f 6e 2f 78 6d 6c  3b 71 3d 30 2e 39 2c 69   tion/xml ;q=0.9,i
0110  6d 61 67 65 2f 77 65 62  70 2c 2a 2f 2a 3b 71 3d   mage/web p,*/*;q=
0120  30 2e 38 0d 0a 41 63 63  65 70 74 2d 4c 61 6e 67   0.8··Acc ept-Lang
```

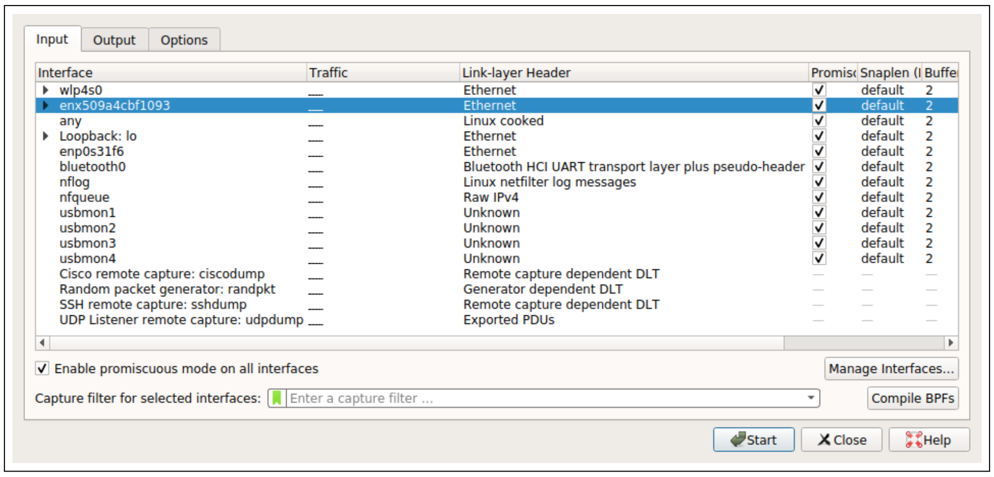
```
Bytes 92-118: Host (http.host)
```

Figure 3: Wireshark Graphical User Interface, during packet capture and analysis.

- The **packet-contents pane** displays the entire content of the captured frame, in both ASCII and hexadecimal formats.
- Towards the top of the Wireshark graphical user interface, is the **display filter** field, into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing (and hence the packet-header and packet-contents). In the example below, we'll use the display filter field to have Wireshark hide (not display) packets except those that correspond to HTTP messages.

1.2 Taking Wireshark for a test run

The best way to learn about any new piece of software is to try it out! We'll assume that your computer is connected to the Internet via a wired Ethernet interface (in the text, it is assumed that Ethernet is used at the link layer) but the output is similar when capturing on a wireless interface. To perform the tasks, you need at least one interface that is connected to the internet: Ethernet, Wifi, Bluetooth, USB, etc.



Interface	Traffic	Link-layer Header	Promisc	Snaplen (I	Buffe
wlp4s0	___	Ethernet	✓	default	2
enx509a4cbf1093	___	Ethernet	✓	default	2
any	___	Linux cooked	✓	default	2
Loopback: lo	___	Ethernet	✓	default	2
enp0s31f6	___	Ethernet	✓	default	2
bluetooth0	___	Bluetooth HCI UART transport layer plus pseudo-header	✓	default	2
nflog	___	Linux netfilter log messages	✓	default	2
nfqueue	___	Raw IPv4	✓	default	2
usbmon1	___	Unknown	✓	default	2
usbmon2	___	Unknown	✓	default	2
usbmon3	___	Unknown	✓	default	2
usbmon4	___	Unknown	✓	default	2
Cisco remote capture: ciscodump	___	Remote capture dependent DLT	—	—	—
Random packet generator: randpkt	___	Generator dependent DLT	—	—	—
SSH remote capture: sshdump	___	Remote capture dependent DLT	—	—	—
UDP Listener remote capture: udpdump	___	Exported PDUs	—	—	—

✓ Enable promiscuous mode on all interfaces

Capture filter for selected interfaces: Enter a capture filter ...

Figure 4: Selecting a network interface.

Tips

For capturing packets at home, it might be easier to analyse what you capture if you close all unnecessary applications (web browser tabs, email client, instant chat, etc) during the execution of the lab. Doing so will help you to find what you want as the quantity of packets being captured and displayed will be much smaller.

Do the following tasks in steps:

1. Start up your favorite web browser, which will display your selected homepage.

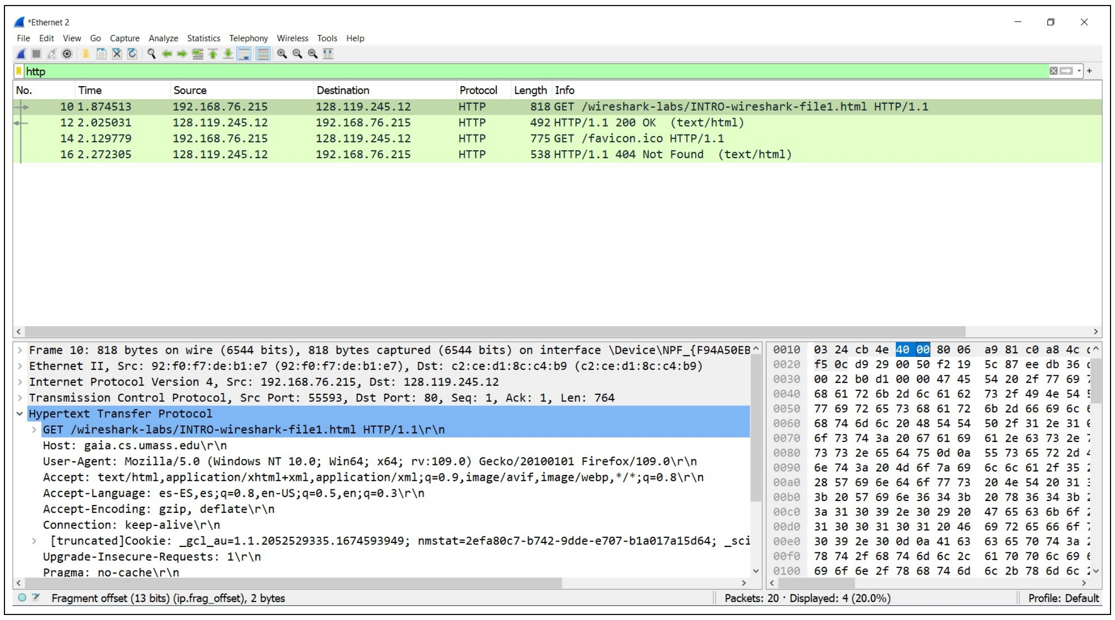
2. Start up the Wireshark software. You will initially see a window similar to the one shown in Figure 2. Wireshark has not yet begun capturing packets, but will show all available interfaces. Next to each interface, you will notice a small chart indicating the number of packets being received on the corresponding interface; this can help you to select an interface currently receiving traffic.

3. To begin packet capture, click on the interface you are currently connected to the Internet. If later on, you would like to change the capturing interface, go *Capture* → *Options...* This will cause the "Wireshark: Capture Interfaces" window to be displayed, as shown in Figure 4, then click on *Start* for the interface on which you want to begin packet capture. Packet capture will now begin. Wireshark is now capturing all packets being sent/received from/by your computer!

   Tips

   Ethernet interfaces are often named *eth0* and wireless interfaces *wlan0*.

4. Once you begin packet capture, a window similar to that shown in Figure 3 will appear. This window shows the packets being captured (live data that contains all protocol messages exchanged between your computer and other network entities). By selecting *Capture* pull-down menu and selecting *Stop*, you can stop packet capture. But don't stop packet capture yet. Let's capture some interesting packets first. To do so, we'll need to generate some network traffic. Let's do so using a web browser, which will use the HTTP protocol (that we will study in detail later in the labs) to download content from a website. Open the following link in your web browser: `http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html` After opening the link, you can stop capturing packets.


*Ethernet 2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

```
No.      Time       Source            Destination       Protocol  Length Info
      10 1.874513   192.168.76.215    128.119.245.12    HTTP        818 GET /wireshark-labs/INTRO-wireshark-file1.html HTTP/1.1
      12 2.025031   128.119.245.12    192.168.76.215    HTTP        492 HTTP/1.1 200 OK  (text/html)
      14 2.129779   192.168.76.215    128.119.245.12    HTTP        775 GET /favicon.ico HTTP/1.1
      16 2.272305   128.119.245.12    192.168.76.215    HTTP        538 HTTP/1.1 404 Not Found  (text/html)
```

```
> Frame 10: 818 bytes on wire (6544 bits), 818 bytes captured (6544 bits) on interface \Device\NPF_{F94A50EB
> Ethernet II, Src: 92:f0:f7:de:b1:e7 (92:f0:f7:de:b1:e7), Dst: c2:ce:d1:8c:c4:b9 (c2:ce:d1:8c:c4:b9)
> Internet Protocol Version 4, Src: 192.168.76.215, Dst: 128.119.245.12
> Transmission Control Protocol, Src Port: 55593, Dst Port: 80, Seq: 1, Ack: 1, Len: 764
v Hypertext Transfer Protocol
  > GET /wireshark-labs/INTRO-wireshark-file1.html HTTP/1.1\r\n
    Host: gaia.cs.umass.edu\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/109.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
  >  [truncated]Cookie: _gcl_au=1.1.2052529335.1674593949; nmstat=2efa80c7-b742-9dde-e707-b1a017a15d64; _sci
    Upgrade-Insecure-Requests: 1\r\n
    Pragma: no-cache\r\n
```

```
0010  03 24 cb 4e 40 00 80 06  a9 81 c0 a8 4c
0020  f5 0c d9 29 00 50 f2 19  5c 87 ee db 36
0030  00 22 b0 d1 00 00 47 45  54 20 2f 77 69
0040  68 61 72 6b 2d 6c 61 62  73 2f 49 4e 54
0050  77 69 72 65 73 68 61 72  6b 2d 66 69 6c
0060  68 74 6d 6c 20 48 54 54  50 2f 31 2e 31
0070  6f 73 74 3a 20 67 61 69  61 2e 63 73 2e
0080  73 73 2e 65 64 75 0d 0a  55 73 65 72 2d
0090  6e 74 3a 20 4d 6f 7a 69  6c 6c 61 2f 35
00a0  28 57 69 6e 64 6f 77 73  20 4e 54 20 31
00b0  3b 20 57 69 6e 36 34 3b  20 78 36 34 3b
00c0  3a 31 30 39 2e 30 29 20  47 65 63 6b 6f
00d0  31 30 30 31 30 31 20 46  69 72 65 66 6f
00e0  30 39 2e 30 0d 0a 41 63  63 65 70 74 3a
00f0  78 74 2f 68 74 6d 6c 2c  61 70 70 6c 69
0100  69 6f 6e 2f 78 68 74 6d  6c 2b 78 6d 6c
```

```
Fragment offset (13 bits) (ip.frag_offset), 2 bytes    Packets: 20 · Displayed: 4 (20.0%)    Profile: Default
```

Figure 5: Capturing HTTP packets.

5. Type in “http” (without the quotes, and in lower case, because all protocol names are in lower case in Wireshark) into the display-filter field at the top of the main Wireshark window, then hit “enter” or click *Apply* (to the right of where you entered “http”). This will cause only HTTP messages to be displayed in the packet-listing pane[1].

   Tips

   You can also select a field like an IP address and apply it as filter by right clicking on the field, then selecting **Apply as Filter** → **Selected**.

6. Find the **HTTP GET** message that was sent from your computer to the **gaia.cs.umass.edu** web server. When you select the HTTP GET message, the Ethernet frame, IP datagram, TCP segment, and HTTP message header information will be displayed in the packet details. By clicking on ▶ and ▼ right-pointing and down-pointing arrowheads to the left side of the packet details pane, you will *minimize* the amount of Frame, Ethernet, Internet Protocol, and Transmission Control Protocol information displayed, or *Maximize* the amount information displayed about the HTTP protocol. Your Wireshark window should now look roughly as the equivalent example in Figure 5.

   Hint

   Look for a packet that shows “GET” followed by the URL that you entered in “Info” column and remember that if you are also web browsing in the same time as capturing packets, many other HTTP packets may have been captured.

2 Preparatory Tasks

The goal of these introductory tasks is only to prepare you for the first lab so there is nothing to handle in at the end of this preparatory lab.

[1]To remove any filters, erase the entire filter bar and hit enter.

2.1 Capturing HTTP traffic

Let's continue a little bit first with Wireshark and explore further some of its basic capabilities.

- Start Wireshark packet capture (blue shark fin symbol[2]).
- Make sure no filter is in place (i.e., erase any filter in the filter bar and hit enter).
- While Wireshark is running, open the following URL in a web browser: `http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html`.
- Once the page is displayed in your browser, then **stop Wireshark packet capture** (red square symbol).

In order to display this page, your browser will contact chalmers' HTTP server and exchange HTTP messages with the server in order to download the webpage, as discussed in **Section 2.2** of the course book. The frames containing these HTTP messages (as well as all other frames passing through your adapter) will be captured. The HTTP messages exchanged with the `gaia.cs.umass.edu` web server should appear somewhere in the listing of packets captured. But there will be many other types of packets displayed as well (see for example the many different protocol types shown in the Protocol column). Even though the only action you have taken is to download a web page, there were evidently many other protocols running on your computer that are unseen by the user. You'll learn much more about these protocols as you progress through the course! For now, you should just be aware that there is often much more going on than "meets the eye"!

Preparatory Task 1. (a) Find 3 protocols that appear in the protocol column in the unfiltered packet listing and indicate to which layer each protocol belongs.

Apply now a `http` filter (i.e., type "http" lowercase without quotes in the filter bar and press enter) and find in the packet list the HTTP GET request and response sent and received in order to retrieve the webpage.

(b) Measure how long time did it take from when the HTTP GET message was sent until the HTTP OK response was received.

(c) What is the Internet IPv4 address of `gaia.cs.umass.edu`

(d) What is the Internet IPv4 address of your computer?

Tips

By default, the value of the **Time column** in the packet-listing panel is the amount of time, in seconds, since Wireshark packet capture began. To display the Time field in time-of-day format, select the Wireshark *View* pull-down menu, then select *Time Display Format*, then select *Time of Day (01:02:03.123456)*.

2.2 Nslookup

In this part, we will learn how to use the small program `nslookup` whose job is to convert domain names into IP addresses by sending DNS requests to your local DNS server or any DNS server. For example, running the following command-line:

[2] Every time a new capture is started, Wireshark will ask if the previous capture should be saved. For the preparatory tasks, you can discard all captures. To remove the annoying pop-up, go to Edit → Preferences... → Appearance → Confirm unsaved capture files.

```
nslookup www.chalmers.se
```

will provide us with the IP addresses of chalmers' webserver, that are **129.16.71.10** (IPv4) and **2001:6b0:2:200c::7110** (IPv6).

Starting the Command-Line Interface (CLI) To perform the preparatory tasks, start first your command-line interpreter following the below tips if needed.

Windows

In Windows, you need to execute **cmd** in the task bar or look for **cmd** program in search field. To find help about commands, use the help parameter `/?` in order to find out what commands do, for example:

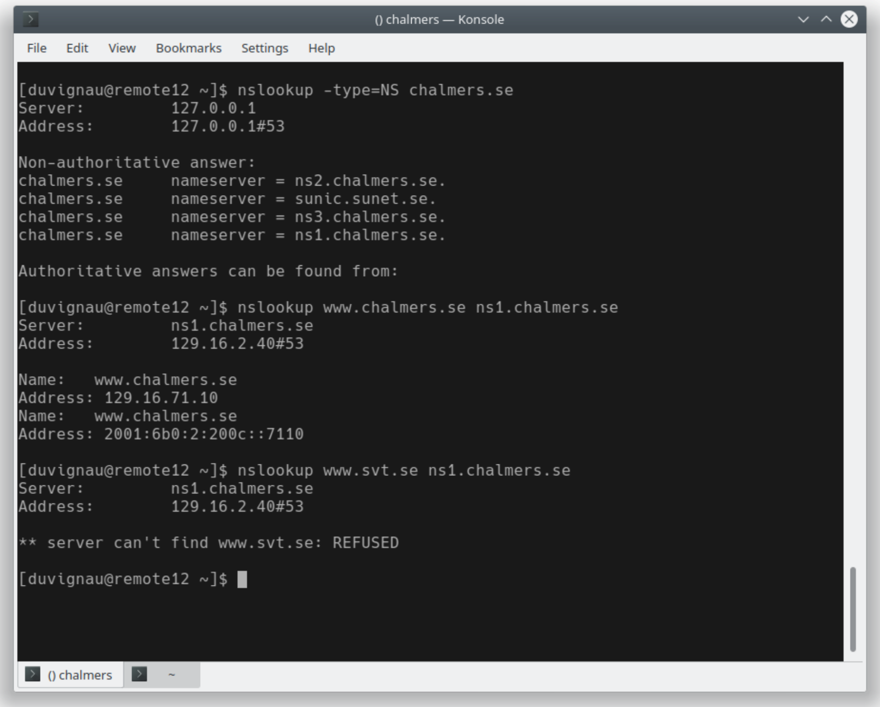
```
C:\>nslookup /?
```

Linux/MacOS

In Unix systems, the graphical program is usually called **Terminal** (eg ubuntu, macOS) or Konsole (kde-based distros). If you need help about a command, you can ask the system in Linux by using the man command in a terminal window, for example:

```
man nslookup
```

Getting started with nslookup In its most basic operation, the `nslookup` tool allows the host running the tool to query any specified DNS server for a DNS record. The queried DNS server can be a **root** DNS server, a **top-level-domain** DNS server, an **authoritative** DNS server, or an intermediate DNS server (see the course book for definitions of these terms). To



```
[duvignau@remote12 ~]$ nslookup -type=NS chalmers.se
Server:         127.0.0.1
Address:        127.0.0.1#53

Non-authoritative answer:
chalmers.se     nameserver = ns2.chalmers.se.
chalmers.se     nameserver = sunic.sunet.se.
chalmers.se     nameserver = ns3.chalmers.se.
chalmers.se     nameserver = ns1.chalmers.se.

Authoritative answers can be found from:

[duvignau@remote12 ~]$ nslookup www.chalmers.se ns1.chalmers.se
Server:         ns1.chalmers.se
Address:        129.16.2.40#53

Name:   www.chalmers.se
Address: 129.16.71.10
Name:   www.chalmers.se
Address: 2001:6b0:2:200c::7110

[duvignau@remote12 ~]$ nslookup www.svt.se ns1.chalmers.se
Server:         ns1.chalmers.se
Address:        129.16.2.40#53

** server can't find www.svt.se: REFUSED

[duvignau@remote12 ~]$ █
```

Figure 6: Nslookup: A utility to send DNS queries.

accomplish this task, `nslookup` sends a DNS query to the specified DNS server, receives a DNS reply from that same DNS server, and displays the result. The basic syntax of running the program is

```
nslookup [-type=t] name [server]
```

where *t* is an optional DNS record type (written in uppercase or lowercase), *name* is a domain name that we want to find the IP address and *server* is an optional DNS server to inquire instead of the default one. Let us make a couple of examples to illustrate how to use the tool.

Figure 6 shows the results of three independent `nslookup` commands. In this example, the client host is located at the Chalmers campus. When running `nslookup`, if no DNS server is specified, then `nslookup` sends the query to the default DNS server, which in this case is the **localhost** (it looks then first in the local cache). Consider the first line of command of Figure 6:

```
nslookup -type=NS chalmers.se
```

In the above line, the user has provided the option "`-type=NS`" for the domain "`chalmers.se`". This causes `nslookup` to send a query for a type NS record to the default local DNS server. In words, the query is saying "*please send me the hostname of the authoritative DNS servers for the domain `chalmers.se`*" to the queried DNS server. When the **–type** option is not used, `nslookup` uses the default one, which is to query for **type A** records. The answer in Figure 6 first indicates the DNS server that is providing the answer (which is here **127.0.0.1**, the localhost) along with four Chalmers name servers. Each of these servers is indeed an authoritative DNS server for the hosts on the Chalmers campuses. However, `nslookup` also indicates that the answer is "non-authoritative," meaning that this answer has come from the cache of the local server rather than directly from an authoritative DNS server.

Now consider the second command-line of Figure 6:

```
nslookup www.chalmers.se ns1.chalmers.se
```

In this example, the user has sent a "`-type=A`" query (default when no type is specified) for the domain "**www.chalmers.se**" and sent the query to DNS server "**ns1.chalmers.se**". In words, the query is saying "*please send me the IP address associated with the hostname **www.chalmers.se***". In this example, the user has indicated that the query should be sent to the authoritative DNS server **ns1.chalmers.se** rather than to the default DNS server. Thus, the query and reply transaction take place directly between the querying host and **ns1.chalmers.se**. As shown in the screenshot, 2 responses for this command were given: (1) the hostname and IPv4 address for **www.chalmers.se** and (2) the hostname and IPv6 address for **www.chalmers.se** (that is, a second "`-type=AAAA`" query was also generated by `nslookup`).

Finally consider the third command of Figure 6:

```
nslookup www.svt.se ns1.chalmers.se
```

In this example, the **ns1** DNS server cannot provide the IP address of the host **www.svt.se**, which is a web server on a different domain (`svt.se`), and **ns1.chalmers.se** has no responsibility (authority) for such translations.

In summary, `nslookup` can be run with zero, one, two or more options with the general syntax being:

```
nslookup -option1 -option2 name_to_find dns_server
```

And as you have seen in the above examples, the `dns_server` is optional as well; if it is not supplied, the query is sent to the default local DNS server.

Understanding nslookup's different types For each possible DNS record type, nslookup sends a DNS request to the queried DNS server for that type and the DNS response displayed by nslookup must be understood accordingly. In summary, the different possibilities are:

Type	**Input of** `nslookup` (DNS Name)	**Output of** `nslookup` (DNS Value)
A	a hostname	IPv4 address(es) associated with the hostname (default)
AAAA	a hostname	IPv6 address(es) associated with the hostname (default)
NS	a domain	Name(s) of authoritative DNS server(s) in the domain
MX	a domain	Name(s) of mail servers in the domain
CNAME	a hostname	Canonical name associated with the hostname (if any)

You should make sure to always use a domain (e.g. **chalmers.se**, **google.com**, **svt.se**) and not the hostname of a web server (e.g. **www.chalmers.se**, **www.google.com**, **www.svt.se**) when looking for domain names (`type=NS`) as well as names of mail servers[3] (`type=MX`). This is a very common mistake so check again the previous examples if needed.

Preparatory Task 2. Pick your favorite domain name and run the command `nslookup` so that to retrieve the following information:

- one **mail server** for the domain.
- the **IP address X** of the web server (www.) associated with the domain name.
- one authoritative **name server Y** (IP and hostname) for the same domain.
- the command-line in order to **ask directly** the server **Y** for the value of **X**.

Tips

Refer to your book for finding more information about the different type of records to look for (MX, NS, etc) and in particular **Section 2.4** page **162**, and remember to remove the host part when looking for records of type MX and NS.

Linux

In addition to the `nslookup` command, you can also experiment with the **host** command (with almost identical syntax, just the option "type=NS" becomes "-t NS").

2.3 Capturing DNS traffic

Let's assume one is interested in retrieving the IP address of the webserver of the Fiji's government and thus execute the command:

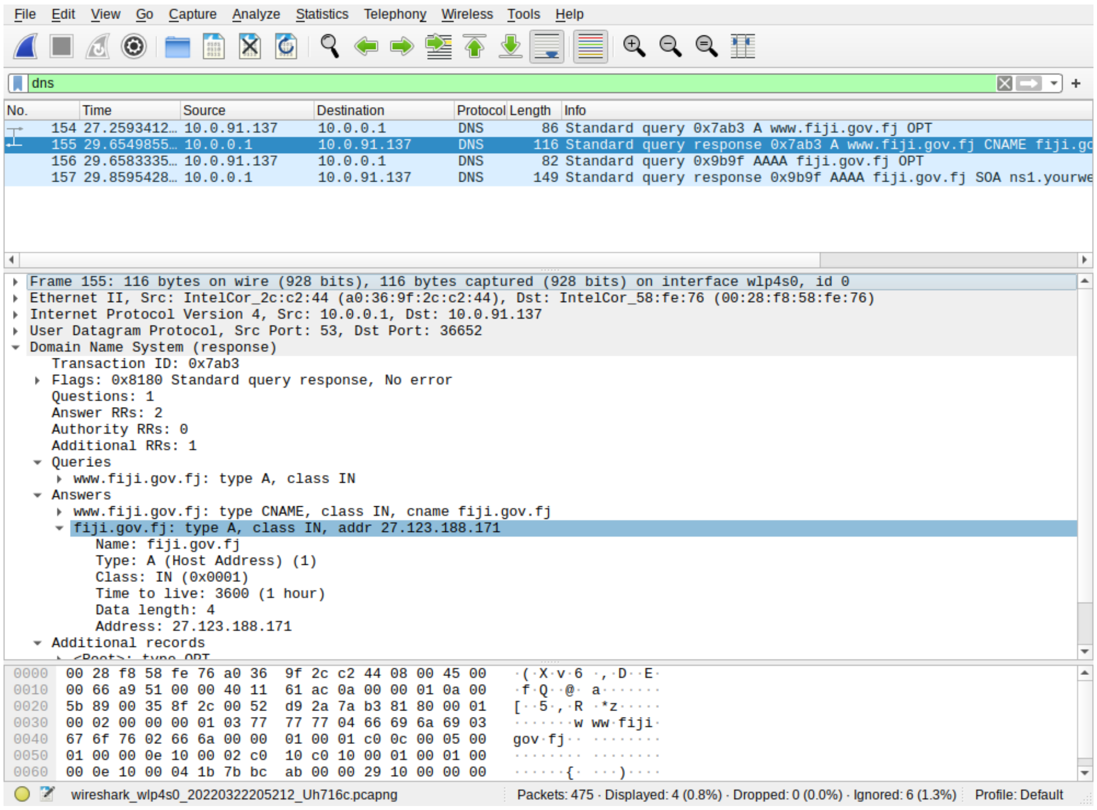
```
nslookup www.fiji.gov.fj
```

Nslookup's answer in this case is

```
Non-authoritative answer:
www.fiji.gov.fj canonical name = fiji.gov.fj.
Name:   fiji.gov.fj
Address: 27.123.188.171
```

That means, even though we were looking for a type A record (the IP address associated with the hostname), we receive as answer a CNAME record with the canonical name of the webserver. On top of that, `nslookup` also provides us with its IP address, but was this address retrieved by sending yet another DNS query? Let us figure that out in Wireshark! Figure 7 presents a screenshot that is obtained when using Wireshark to capture the packets when `nslookup` is used to resolve `www.fiji.gov.fj`. Inspecting the response from the local DNS in Wireshark, we can see that in the section "Answers" both the CNAME and the type A record are provided, and

[3]More precisely, `nslookup` will retrieve the canonical name of mail servers that have been registered in the DNS database as having the targeted domain or hostname as an alias.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns

```
No.   Time          Source        Destination   Protocol Length Info
  154 27.2593412… 10.0.91.137   10.0.0.1      DNS        86 Standard query 0x7ab3 A www.fiji.gov.fj OPT
  155 29.6549855… 10.0.0.1      10.0.91.137   DNS       116 Standard query response 0x7ab3 A www.fiji.gov.fj CNAME fiji.go
  156 29.6583335… 10.0.91.137   10.0.0.1      DNS        82 Standard query 0x9b9f AAAA fiji.gov.fj OPT
  157 29.8595428… 10.0.0.1      10.0.91.137   DNS       149 Standard query response 0x9b9f AAAA fiji.gov.fj SOA ns1.yourwe
```

```
▸ Frame 155: 116 bytes on wire (928 bits), 116 bytes captured (928 bits) on interface wlp4s0, id 0
▸ Ethernet II, Src: IntelCor_2c:c2:44 (a0:36:9f:2c:c2:44), Dst: IntelCor_58:fe:76 (00:28:f8:58:fe:76)
▸ Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.91.137
▸ User Datagram Protocol, Src Port: 53, Dst Port: 36652
▾ Domain Name System (response)
    Transaction ID: 0x7ab3
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 2
    Authority RRs: 0
    Additional RRs: 1
  ▾ Queries
    ▸ www.fiji.gov.fj: type A, class IN
  ▾ Answers
    ▸ www.fiji.gov.fj: type CNAME, class IN, cname fiji.gov.fj
    ▾ fiji.gov.fj: type A, class IN, addr 27.123.188.171
        Name: fiji.gov.fj
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 3600 (1 hour)
        Data length: 4
        Address: 27.123.188.171
  ▾ Additional records
```

```
0000  00 28 f8 58 fe 76 a0 36  9f 2c c2 44 08 00 45 00   ·(·X·v·6 ·,·D··E·
0010  00 66 a9 51 00 00 40 11  61 ac 0a 00 00 01 0a 00   ·f·Q··@· a·······
0020  5b 89 00 35 8f 2c 00 52  d9 2a 7a b3 81 80 00 01   [··5·,·R ·*z·····
0030  00 02 00 00 00 01 03 77  77 77 04 66 69 6a 69 03   ·······w ww·fiji·
0040  67 6f 76 02 66 6a 00 00  01 00 01 c0 0c 00 05 00   gov·fj·· ········
0050  01 00 00 0e 10 00 02 c0  10 c0 10 00 01 00 01 00   ········ ········
0060  00 0e 10 00 04 1b 7b bc  ab 00 00 29 10 00 00 00   ······{· ···)····
```

```
wireshark_wlp4s0_20220322205212_Uh716c.pcapng    Packets: 475 · Displayed: 4 (0.8%) · Dropped: 0 (0.0%) · Ignored: 6 (1.3%)    Profile: Default
```

Figure 7: Wireshark capturing DNS messages.

thus only 1 DNS query was sent. We can however see another DNS query afterwards. You have to remember that by default if no specific type is specified, `nslookup` wants to retrieve both a type A and a type AAAA records for the hostname. Hence the second “type=AAAA” query is sent, but this time nslookup uses directly the CNAME that it just learned!

Preparatory Task 3. Do a similar packet capturing when running `nslookup` for a mail server (type=MX) in the domain of your choice. Locate the *Name*, *Type* and *Time to Live* (TTL) in the DNS *response* message.

Hint

To only display DNS messages, you can use the filter `dns`.

Have a look at the different parts in the query and response messages, including the header and the DNS information in the message sections (Queries, Answers, Authoritative name servers and Additional records). In each section, there may be zero or more *Resource Records* (RR). Learn about the format of the RR (specifically the *Name*, *Type*, *TTL* and *Value* as explained in your course book **page 162**).

Tips

By performing the different tasks, you might have realized that no DNS traffic was sometimes captured. This is due to your host using addresses already stored in its **DNS cache**. Simply pick a different domain name in this case, or reboot your machine :)