Lab 1: WireShark and other tools

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

You could think of a network packet analyzer as a measuring device used to examine what's going on inside a network, just like a voltmeter is used by an electrician to examine what's going on inside an electric cable (but at a higher level, of course).

Wireshark is available for UNIX, Windows, and MacOSX and it can capture traffic from many different network media types - and despite its name - including wireless LAN as well. Which media types are supported, depends on many things like the operating system you are using.

Here are some examples people use Wireshark for:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

Beside these examples Wireshark can be helpful in many other situations too.

In this lab we will learn:

- How the Wireshark user interface works
- How to capture packets in Wireshark
- How to view packets in Wireshark
- How to filter packets in Wireshark
- … and many other things!

Our aim is to learn how to use Wireshark to learn TCP/IP protocol internals, capturing and analying real network traffic.

Previous Reading: Read sections 1, 2 and 3 before attending the Laboratory session

8.1. WireShark

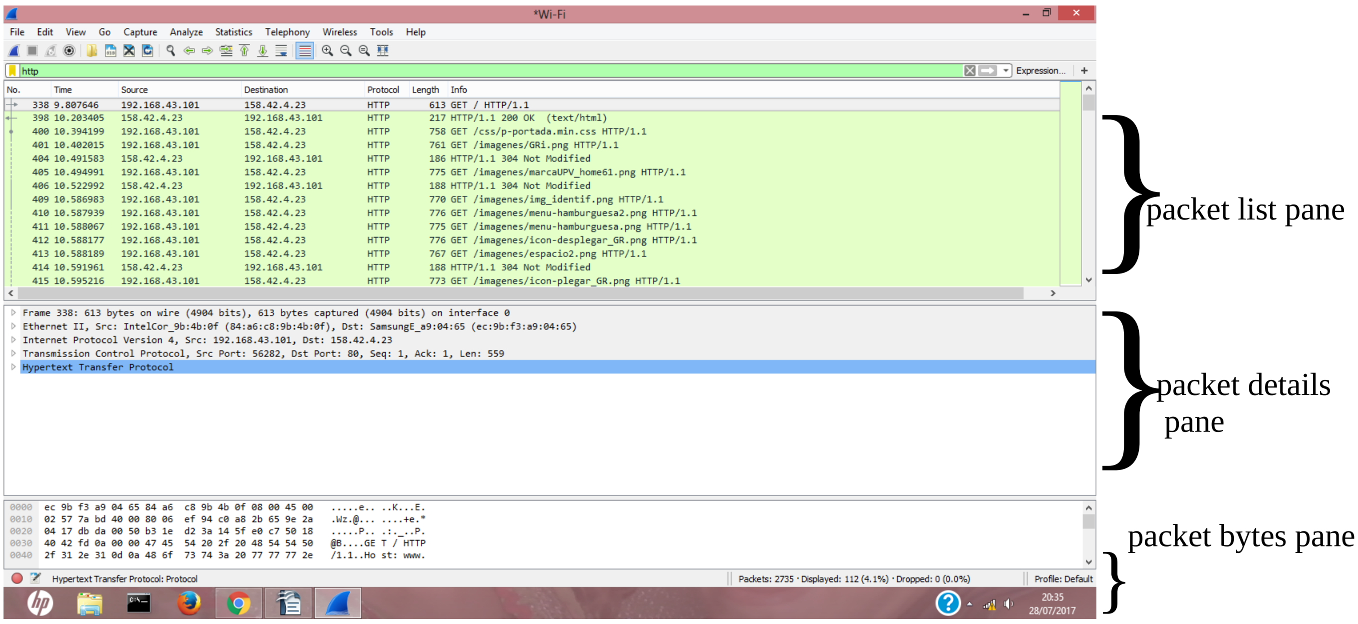
Wireshark is an open source software project, and is released under the GNU General Public License (GPL). You can get the latest copy of the program from the Wireshark website at https://www.wireshark.org/download.html.

Some of the many features Wireshark provides:

- Capture live packet data from a network interface.
- Open files containing packet data captured from a large number of other capture programs as for example tcpdump/WinDump, or NetXRay.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
- Save packet data captured.
- Export some or all packets in a number of capture file formats.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.
- Create various statistics.
- …and a lot more!

1.1. Start Wireshark

You can start Wireshark from your shell or window manager. Once it is started, and some packets are captured, let's look at Wireshark's user interface. Figure 1, "The Main window" shows Wireshark as you would usually see it after some packets are captured


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Expression...
packet list pane
packet details pane
packet bytes pane
Hypertext Transfer Protocol: Protocol
Packets: 2735 · Displayed: 112 (4.1%) · Dropped: 0 (0.0%)
Profile: Default
20:35
28/07/2017

```
No.   Time       Source           Destination      Protocol  Length Info
  338 9.807646   192.168.43.101   158.42.4.23      HTTP       613 GET / HTTP/1.1
  398 10.203405  158.42.4.23      192.168.43.101   HTTP       217 HTTP/1.1 200 OK  (text/html)
  400 10.394199  192.168.43.101   158.42.4.23      HTTP       758 GET /css/p-portada.min.css HTTP/1.1
  401 10.402015  192.168.43.101   158.42.4.23      HTTP       761 GET /imagenes/GRi.png HTTP/1.1
  404 10.491583  158.42.4.23      192.168.43.101   HTTP       186 HTTP/1.1 304 Not Modified
  405 10.494991  192.168.43.101   158.42.4.23      HTTP       775 GET /imagenes/marcaUPV_home61.png HTTP/1.1
  406 10.522992  158.42.4.23      192.168.43.101   HTTP       188 HTTP/1.1 304 Not Modified
  409 10.586983  192.168.43.101   158.42.4.23      HTTP       770 GET /imagenes/img_identif.png HTTP/1.1
  410 10.587939  192.168.43.101   158.42.4.23      HTTP       776 GET /imagenes/menu-hamburguesa2.png HTTP/1.1
  411 10.588067  192.168.43.101   158.42.4.23      HTTP       775 GET /imagenes/menu-hamburguesa.png HTTP/1.1
  412 10.588177  192.168.43.101   158.42.4.23      HTTP       776 GET /imagenes/icon-desplegar_GR.png HTTP/1.1
  413 10.588189  192.168.43.101   158.42.4.23      HTTP       767 GET /imagenes/espacio2.png HTTP/1.1
  414 10.591961  158.42.4.23      192.168.43.101   HTTP       188 HTTP/1.1 304 Not Modified
  415 10.595216  192.168.43.101   158.42.4.23      HTTP       773 GET /imagenes/icon-plegar_GR.png HTTP/1.1
```

```
▷ Frame 338: 613 bytes on wire (4904 bits), 613 bytes captured (4904 bits) on interface 0
▷ Ethernet II, Src: IntelCor_9b:4b:0f (84:a6:c8:9b:4b:0f), Dst: SamsungE_a9:04:65 (ec:9b:f3:a9:04:65)
▷ Internet Protocol Version 4, Src: 192.168.43.101, Dst: 158.42.4.23
▷ Transmission Control Protocol, Src Port: 56282, Dst Port: 80, Seq: 1, Ack: 1, Len: 559
▷ Hypertext Transfer Protocol
```

```
0000  ec 9b f3 a9 04 65 84 a6  c8 9b 4b 0f 08 00 45 00   .....e.. ..K...E.
0010  02 57 7a bd 40 00 80 06  ef 94 c0 a8 2b 65 9e 2a   .Wz.@... ....+e.*
0020  04 17 db da 00 50 b3 1e  d2 3a 14 5f e0 c7 50 18   .....P.. .:._..P.
0030  40 42 fd 0a 00 00 47 45  54 20 2f 20 48 54 54 50   @B....GE T / HTTP
0040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 77 77 77 2e   /1.1..Ho st: www.
```

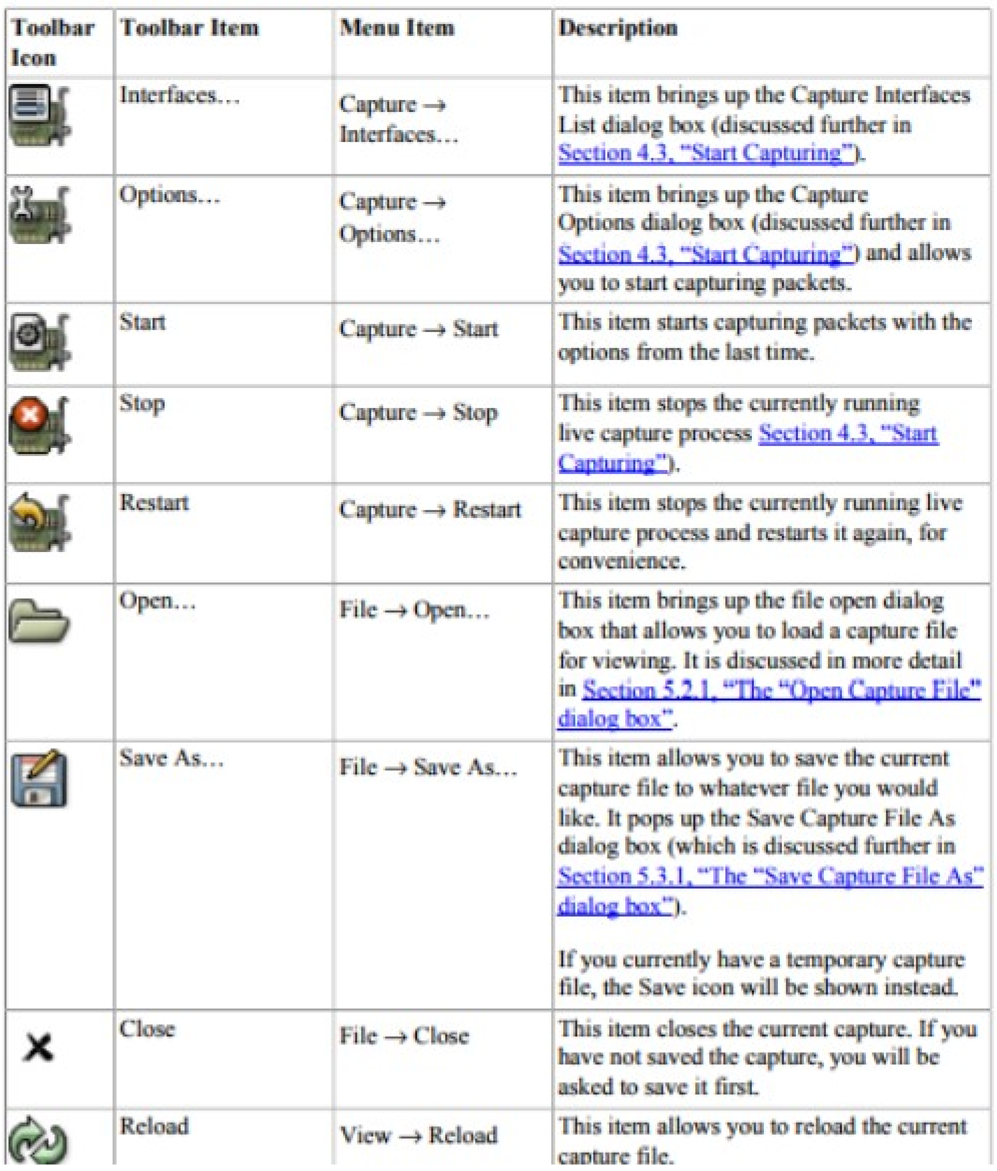
Figure 1. The Main window

Wireshark's main window consists of parts that are commonly known from many other GUI programs.

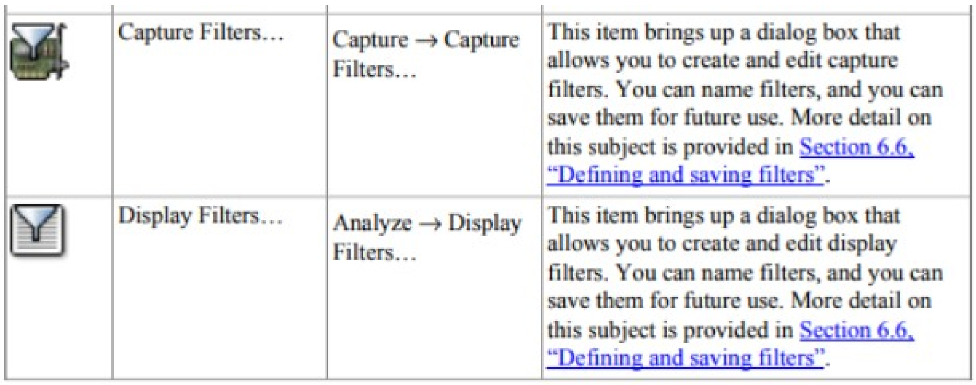
1. The **menu** is used to start actions.
2. The **main toolbar** provides quick access to frequently used items from the menu.

   This toolbar cannot be customized by the user, but it can be hidden using the View menu, if the space on the screen is needed to show even more packet data. As in the menu, only the items useful in the current program state will be available. The others will be greyed out (e.g. you cannot save a capture file if you haven't loaded one).

Table 3.13. Main toolbar items



Toolbar Icon	Toolbar Item	Menu Item	Description
	Interfaces...	Capture → Interfaces...	This item brings up the Capture Interfaces List dialog box (discussed further in Section 4.3, "Start Capturing").
	Options...	Capture → Options...	This item brings up the Capture Options dialog box (discussed further in Section 4.3, "Start Capturing") and allows you to start capturing packets.
	Start	Capture → Start	This item starts capturing packets with the options from the last time.
	Stop	Capture → Stop	This item stops the currently running live capture process Section 4.3, "Start Capturing").
	Restart	Capture → Restart	This item stops the currently running live capture process and restarts it again, for convenience.
	Open...	File → Open...	This item brings up the file open dialog box that allows you to load a capture file for viewing. It is discussed in more detail in Section 5.2.1, "The "Open Capture File" dialog box".
	Save As...	File → Save As...	This item allows you to save the current capture file to whatever file you would like. It pops up the Save Capture File As dialog box (which is discussed further in Section 5.3.1, "The "Save Capture File As" dialog box"). If you currently have a temporary capture file, the Save icon will be shown instead.
✕	Close	File → Close	This item closes the current capture. If you have not saved the capture, you will be asked to save it first.
	Reload	View → Reload	This item allows you to reload the current capture file.



	Capture Filters…	Capture → Capture Filters…	This item brings up a dialog box that allows you to create and edit capture filters. You can name filters, and you can save them for future use. More detail on this subject is provided in Section 6.6, “Defining and saving filters”.
	Display Filters…	Analyze → Display Filters…	This item brings up a dialog box that allows you to create and edit display filters. You can name filters, and you can save them for future use. More detail on this subject is provided in Section 6.6, “Defining and saving filters”.

3. The **filter toolbar** provides a way to directly manipulate the currently used display filter.
4. The **packet list pane** displays a summary of each packet captured. By clicking on packets in this pane you control what is displayed in the other two panes.

   Each line in the packet list corresponds to one packet in the capture file. If you select a line in this pane, more details will be displayed in the “Packet Details” and “Packet Bytes” panes. While dissecting a packet, Wireshark will place information from the protocol dissectors into the columns. As higher level protocols might overwrite information from lower levels, you will typically see the information from the highest possible level only.

   The default columns will show:

   - **No.** The number of the packet in the capture file. This number won’t change, even if a display filter is used.
   - **Time** The timestamp of the packet.
   - **Source** The address where this packet is coming from.
   - **Destination** The address where this packet is going to.
   - **Protocol** The protocol name in a short (perhaps abbreviated) version.
   - **Length** The length of each packet.
   - **Info** Additional information about the packet content.
   - The first column shows how each packet is related to the selected packet. For example, in the image above the first packet is selected, which is a HTTP request. Wireshark shows a rightward arrow for the request itself, followed by a leftward arrow for the response in packet 2. Why is there a dashed line? There are more HTTP packets further down that use the same port numbers.

Wireshark treats them as belonging to the same conversation and draws a line connecting them.

5. The **packet details pane** displays the packet selected in the packet list pane in more detail.

   This pane shows the protocols and protocol fields of the packet selected in the "Packet List" pane. The protocols and fields of the packet shown in a tree which can be expanded and collapsed. There is a context menu (right mouse click) available.

6. The **packet bytes pane** displays the data from the packet selected in the packet list pane, and highlights the field selected in the packet details pane.

   It shows a canonical hex dump of the packet data. Each line contains the data offset, sixteen hexadecimal bytes, and sixteen ASCII bytes. Non-printalbe bytes are replaced with a period ('.').

7. The **statusbar** shows some detailed information about the current program state and the captured data.

   In general, the left side will show context related information, the middle part will show information about the current capture file, and the right side will show the selected configuration profile. Drag the handles between the text areas to change the size

1.2. Capturing Live Network Data

Setting up Wireshark to capture packets for the first time can be tricky.

Here are some common pitfalls:

- You may need special privileges to start a live capture.
- You need to choose the right network interface to capture packet data from.
- You need to capture at the right place in the network to see the traffic you want to see.

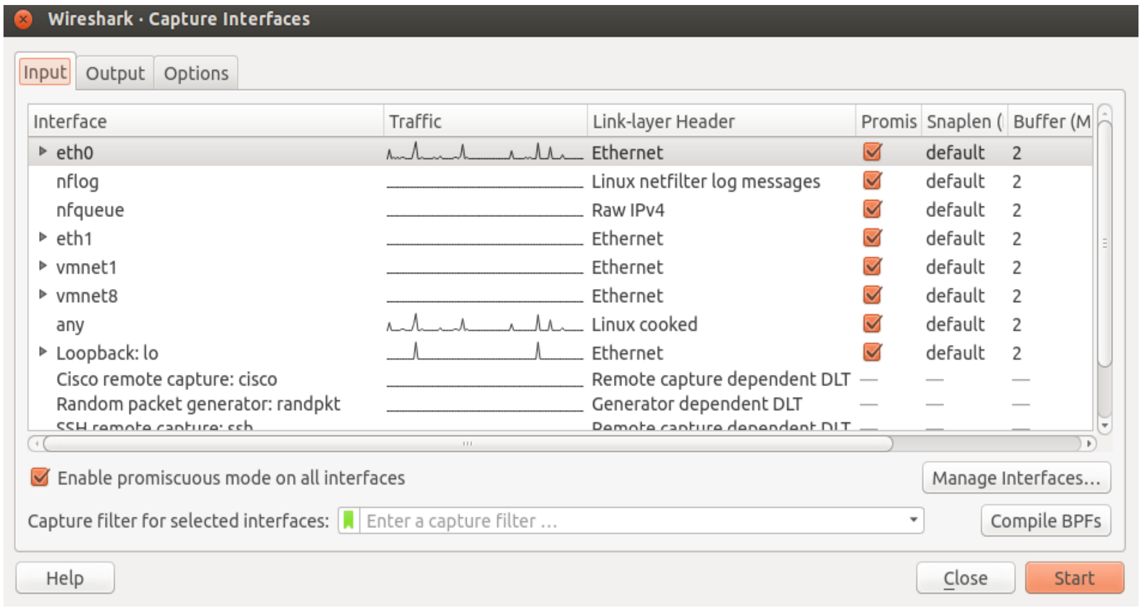
The following methods can be used to start capturing packets with Wireshark:

You can get an overview of the available interfaces using the "Capture Interfaces" dialog box (Capture → Options…).

When you select Capture → Options… from the main menu Wireshark pops up the "Capture Interfaces" dialog box as shown in Figure 2, "The "Capture Interfaces" dialog box on Unix/Linux"

Choose the input tab. there

- choose the right interface to capture from. In the lab computers you should choose the interface which IP address matchs with the IP address in the label of the computer (to see the IP address of *eth0* and *eth1* just move the mouse over on each of them).
- Choose *Capture all in promiscuous mode* to capture traffic destined for your machine, and machines other than your own



Wireshark · Capture Interfaces

Input | Output | Options

Interface	Traffic	Link-layer Header	Promis	Snaplen (	Buffer (M
▸ eth0		Ethernet	✓	default	2
nflog		Linux netfilter log messages	✓	default	2
nfqueue		Raw IPv4	✓	default	2
▸ eth1		Ethernet	✓	default	2
▸ vmnet1		Ethernet	✓	default	2
▸ vmnet8		Ethernet	✓	default	2
any		Linux cooked	✓	default	2
▸ Loopback: lo		Ethernet	✓	default	2
Cisco remote capture: cisco		Remote capture dependent DLT	—	—	—
Random packet generator: randpkt		Generator dependent DLT	—	—	—
SSH remote capture: ssh		Remote capture dependent DLT	—	—	—

✓ Enable promiscuous mode on all interfaces

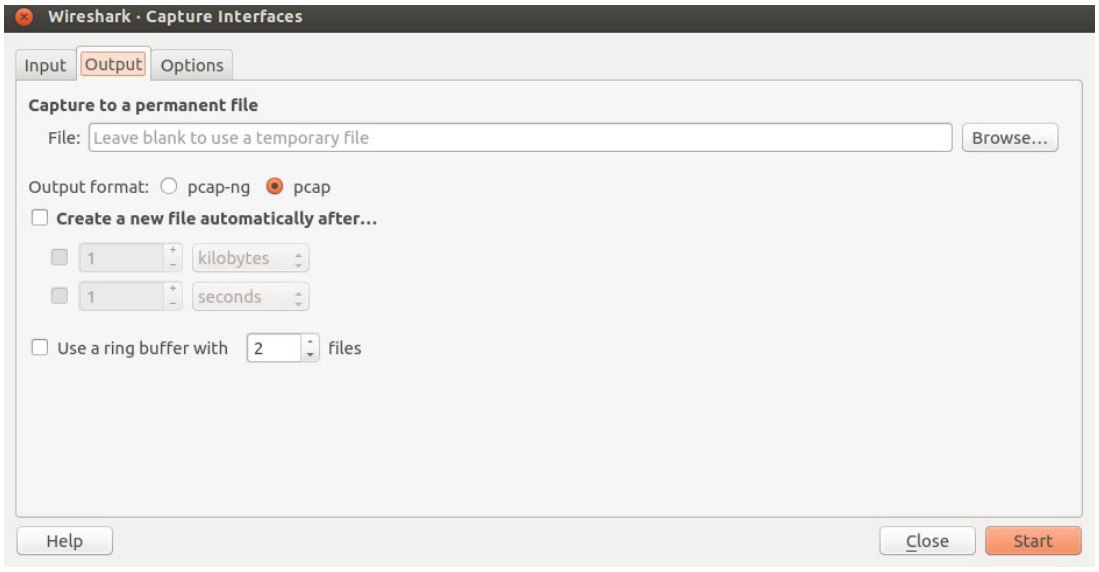
Capture filter for selected interfaces: Enter a capture filter ...

Figure 2. The "Capture Interfaces" dialog box on Unix/Linux . Input tab

Also, *Capture Filter* field allows you to specify a capture filter for all interfaces that are currently selected. Once a filter has been entered in this field, the newly selected interfaces will inherit the filter. Capture filters are discussed in more detail later in this document.

In the Output tab, you can choose to save captured packets (Figure 3). Also, you can do that simply by using the File → Save As… menu item. You can choose which packets to save and which file format to be used.

}



Capture to a permanent file

File: Leave blank to use a temporary file

Output format: pcap-ng [✓ pcap]

☐ **Create a new file automatically after...**

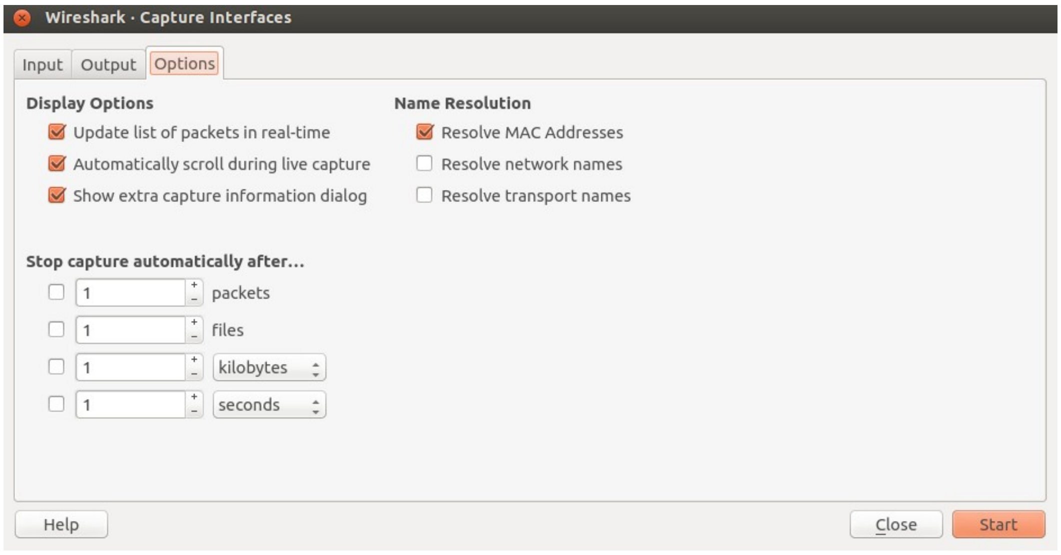
☐ 1 kilobytes

☐ 1 seconds

☐ Use a ring buffer with 2 files

Finally, in the Output tab (Figure 4), we can choose to see in real time the packages that are being captured and we can also choose to perform an automatic vertical scrolling.

In order to finish the capture, we can indicate that it will automatically end when a certain number of packages have been captured, or a certain amount of Kbytes has been captured, or when a certain amount of time has elapsed. If you do not select any of the options, we will have to finalize the capture manually. To start the capture, press the Start button (Input tab).



Display Options

☑ Update list of packets in real-time

☑ Automatically scroll during live capture

☑ Show extra capture information dialog

Name Resolution

☑ Resolve MAC Addresses

☐ Resolve network names

☐ Resolve transport names

Stop capture automatically after...

☐ 1 packets

☐ 1 files

☐ 1 kilobytes

☐ 1 seconds

1.3. Capture filters and display filters

Wireshark has two filtering modes: one used when capturing packets, and one used when displaying packets.

A capture filter is used to select which packets should be saved to disk while capturing. The defined capture filter controls what type of traffic is captured, and disregards all non-matching traffic. Capture filters are set before starting a packet capture and cannot be modified during the capture. You will enter the capture filter into the "*Filter*" field of the Wireshark "*Capture Options*" dialog box (Input tab) (Figure 2).

Display filters on the other hand do not have this limitation and you can change them on the fly. Display filter are used to change the view of a capture file. It controls what is seen from an existing packet capture, but does not influence what traffic is actually captured. When using a display filter, all packets remain in the capture file. The display filter only changes the display of the capture file but not its content! You will enter the display filter into the "*Filter:*" field in the filter toolbar of the Wireshark main window (Figure 1) and press enter to initiate the filter. To remove the display filter, click on the Clear button to the right of the filter field, after it, all captured packet will be shown again.

a) Captures packet filter syntax

The defined capture filter controls what type of traffic is captured (the filter expression is true), and disregards all non-matching traffic.

The filter expression consists of one or more primitives connected by logical operators (and, or, not). Primitives usually consist of an id (name or number) preceded by one or more qualifiers. There are three different kinds of qualifier:

type

type qualifiers say what kind of thing the **id** name or number refers to. Possible types are host, net , port and portrange.

If there is no type qualifier, host is assumed.

The broadcast identifier is also specified, which allows you to reference broadcast addresses.

E.g., `host 158.42.180.62`

captures all packets addressed to or from that IP address.

`net 158.42`

captures all packets addressed to or from that IP network. }

`port 7`

captures all packets addressed to or from port 7, both tcp and udp.

```
portrange 6000-6008
```

captures all packets addressed to or from ports in range 6000-6008, both tcp and udp.

dir

dir qualifiers specify a particular transfer direction to and/or from id. Possible directions are src, dst, src or dst, src and dst.

If there is no dir qualifier, src or dst is assumed.

E.g., `src 158.42.181.18`

```
src port 25
src or dst net 158.42
```

proto

proto qualifiers restrict the match to a particular protocol. Possible protos are: `arp`, `ip`, `icmp`, `tcp`, o `udp.`

If there is no proto qualifier, all protocols consistent with the type are assumed.

Multiple primitives can be combined using the and and or connectors. It is also possible to use the not operator.

E.g., `udp and dst 158.42.148.3`

```
dst 158.42.181.4 or 158.42.181.15
dst 158.42.181.4 and (udp or icmp)
```

This is a brief overview of the syntax. Complete documentation can be found in the online manual typing in a terminal the command `man tcpdump`. Also, you can find a lot of Capture Filter examples at https://wiki.wireshark.org/CaptureFilters.

2. Study of the TCP / IP protocol stack using the Wireshark

In this section, we will use Wireshark to study the TCP/IP protocol stack encapsulation.

Exercise 1.1:

Select the appropriate interface (possibly eth0) and perform a capture of WWW traffic, applying the `port 80` capture filter. This capture filter captures all the HTTP protocol messages that are generated when our browser requests a web page to a server.

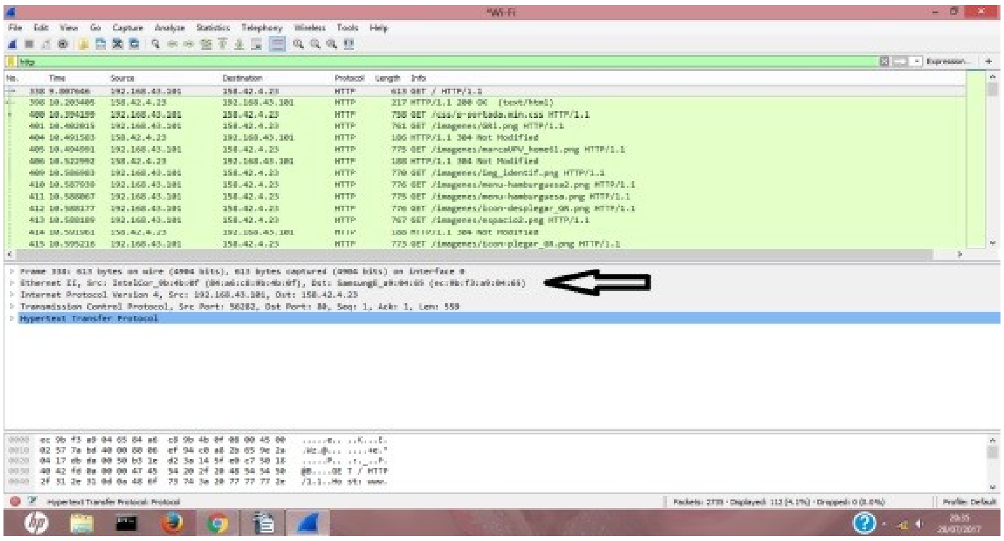
To generate traffic you can run a browser to access www.upv.es web page. Once you get that page, stop capturing.

We should have obtained something similar to Figure 1. We can use a display filter: `http` to see only HTTP protocol packets. Now, select the first captured packet in the packet list pane window. This packet must be the HTTP request message (GET message) generated by our browser when we have requested the web page.

This message is generated in the Application Layer of our computer (HTTP client) and sent to the Web server Application Layer of the UPV (HTTP server). When data moves from upper layer to lower layer of TCP/IP protocol stack (outgoing transmission) each layer includes a bundle of relevant information called a header along with the actual data (encapsulation). The data package containing the header and the data from the upper layer then becomes the data that is repackaged at the next lower level with lower layer's header. Header is the supplemental data placed at the beginning of a block of data when it is transmitted. This supplemental data is used at the receiving side to extract the data from the encapsulated data packet.

Once it arrives at the destination the reverse process of encapsulation (or decapsulation) occurs. As the data moves up from the lower layer to the upper layer of TCP/IP protocol stack (incoming transmission), each layer unpacks the corresponding header and uses the information contained in the header to deliver the packet to the exact network application waiting for the data.

Now we will analyze in more detail each of the layers that appear in this first selected package. To do this, we focus on the packet details pane.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Expression...

```
No.   Time         Source           Destination      Protocol  Length  Info
  338 9.887646     192.168.43.101   158.42.4.23      HTTP      613 GET / HTTP/1.1
  398 10.203405    158.42.4.23      192.168.43.101   HTTP      217 HTTP/1.1 200 OK  (text/html)
  400 10.394159    192.168.43.101   158.42.4.23      HTTP      758 GET /css/e-portada.min.css HTTP/1.1
  401 10.402015    192.168.43.101   158.42.4.23      HTTP      761 GET /imagenes/G01.png HTTP/1.1
  404 10.491503    158.42.4.23      192.168.43.101   HTTP      186 HTTP/1.1 304 Not Modified
  405 10.494091    192.168.43.101   158.42.4.23      HTTP      775 GET /imagenes/marcaUPV_home81.png HTTP/1.1
  406 10.522992    158.42.4.23      192.168.43.101   HTTP      188 HTTP/1.1 304 Not Modified
  409 10.586083    192.168.43.101   158.42.4.23      HTTP      770 GET /imagenes/img_identif.png HTTP/1.1
  410 10.587930    192.168.43.101   158.42.4.23      HTTP      776 GET /imagenes/menu-hamburguesa2.png HTTP/1.1
  411 10.588067    192.168.43.101   158.42.4.23      HTTP      775 GET /imagenes/menu-hamburguesa.png HTTP/1.1
  412 10.588177    192.168.43.101   158.42.4.23      HTTP      776 GET /imagenes/icon-desplegar_0R.png HTTP/1.1
  413 10.588189    192.168.43.101   158.42.4.23      HTTP      767 GET /imagenes/espacio2.png HTTP/1.1
  414 10.591961    158.42.4.23      192.168.43.101   HTTP      188 HTTP/1.1 304 Not Modified
  415 10.595218    192.168.43.101   158.42.4.23      HTTP      773 GET /imagenes/icon-plegar_0R.png HTTP/1.1

> Frame 338: 613 bytes on wire (4904 bits), 613 bytes captured (4904 bits) on interface 0
> Ethernet II, Src: IntelCor_0b:4b:0f (84:a6:c8:9b:4b:0f), Dst: SamsungE_a9:04:65 (ec:9b:f3:a9:04:65)
> Internet Protocol Version 4, Src: 192.168.43.101, Dst: 158.42.4.23
> Transmission Control Protocol, Src Port: 56282, Dst Port: 80, Seq: 1, Ack: 1, Len: 559
> Hypertext Transfer Protocol

0000  ec 9b f3 a9 04 65 84 a6  c8 9b 4b 0f 08 00 45 00   ......e.. ..K...E.
0010  02 57 7a bd 40 00 80 06  ef 94 c0 a8 2b 65 9e 2a   .Wz.@... ....+e.*
0020  04 17 db da 00 50 b3 1e  d2 3a 14 5f e0 c7 50 18   .....P.. .:._..P.
0030  40 42 4e 0a 00 00 47 45  54 20 2f 20 48 54 54 50   @B....GE T / HTTP
0040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 77 77 77 2e   /1.1..Ho st: www.

Hypertext Transfer Protocol: Protocol        Packets: 2738 · Displayed: 112 (4.1%) · Dropped: 0 (0.0%)        Profile: Default
```

}

Figura 5. Packet details pane

In the packet details pane we can observe 5 lines. Each of them corresponds to a layer of the TCP / IP protocol stack. The first line corresponds to the Physical Layer and the last one to the Application Layer.

The protocols and fields of the packet shown in a tree which can be expanded and collapsed. There is a context menu (right mouse click) available. In this Pop-up menu of the "Packet Details" pane" we can find "Copy→ Value" , that can be used to copy the contents of some lines to a document. The Ctl + Shift + V key combination also allows the same effect.

Exercise 1.2:

We now focus on the fifth line of the packet details pane that corresponds to the Application Layer. If we expand it we can see the complete HTTP request message that our machine send to the web server of the UPV.

What does the Host field indicate? What is the field that indicates what type of client software is being used?

Exercise 1.3:

Expand now the fourth line corresponding to the Transport Layer. What protocol appears at this layer? What other protocol might appear? What is the fundamental difference between the two protocols? What identifiers appear at this layer to identify the source and the destination process? Take note of them.

The Transport Layer is responsible for collecting the message generated by the application and preparing it to be sent according to the requirements of the application. In the destination the message is delivered to the corresponding process running in the Application Layer. The port numbers identify the source and destination processes at the Application Layer.

Look at the source and destination ports that appear at this layer of wireshark. Are random numbers chosen by the operating system? What is the difference between client and server port?

Exercise 1.4:

We now focus on the third line of the packet details pane. If we expand it, we can see in detail the fields belonging to the header that has added by Network Layer. Remember that this layer is in charge of routing the package through the Internet to reach its destination. For this it is important to perfectly identify both, the source and the destination host.

What protocol appears at this layer? What kind of addresses are used to identify at this layer the source and destination host? Whose addresses are these? Write them down.

Another interesting point is to be able to see the value of each of the fields that appears in the packet details pane, in hexadecimal and in ASCII. To do this, simply select the field packet details pane and see the values that are highlighted in the packet bytes pane.

Indicates the corresponding values of the source and destination IP addresses in hexadecimal.

Exercise 1.5:

Let's now analyze the second line of the packet details pane that corresponds to the Link Layer. Link Layer is responsible for sending packets to devices that are directly connected or in the same LAN (Local Area Network).

Therefore, the addresses that appears as source and destination will always belong to the same LAN.

What kind of addresses appear as origin and destination in this layer? Whose addresses are these? Do any of them belong to the web server of the UPV? Write them down.

When and where will the UPV Web server physical address appear in the link layer header? What network node will add that header? In the link layer header of that packet, whose source and destination physical addresses are these?

Finally, the first line of the packet details pane corresponds to the Physical Layer. If we display this line we can see details related to this layer. Physical layer deals with the actual ones and zeroes that are sent over the network. It will send HTTP message, with all the headers, through the physical medium. At this layer we can see, among other things, the identifier of the network interface through which the information is transmitted.

Exercise 2: }

Let's work with the DNS application protocol, so we have to remove the “http” display filter (attention, once deleted continue acting until you press the <Enter> key. Modify the capture filter to "port 53" and run a new traffic capture. Generate traffic loading a new web page.

This filter should capture all messages in the DNS application layer protocol. This protocol allows you to find out the IP address corresponding to a host name.

Select the first DNS packet, it will correspond to the DNS query that your host sends to the DNS server, asking about the IP address of the web server name where is allocated the web page that you want to download. Analyze in detail the headers of each level. Can you find protocol differences related to the protocols acting in each layer between the DNS and HTTP capture? Is there any header in which source and destination addresses are identical with those in the HTTP capture? Why?

3. SSH Application.

SSH (Secure Socket Shell) is an application that provides a secure way to access a remote computer as we were sitting there. The application protocol that define it is also called SSH. Usually when the client connects to the remote server, it asks to be identified by a username and password.

The way to use ssh is:

$ ssh user_name@server_host_name

Exercise 3:

Open a terminal and use the command *`ls -la`* to get a list of the current working directory.

Open a new terminal and connect via ssh to the server that we have available in the Network lab: *zoltar.redes.upv.es*. (Your teacher will provide you the username and password).

Run the `ls -la` command again. Which host is the file list from? Which user is the owner? Do your colleagues get the same file list?

You can close the remote session using the `exit` command.

As we can see, the SSH application allows several users to work on the same remote machine.

Exercise 4:

Remove previous the display and capture filters, and write as new capture filter `port 22` (we do that because ssh server is allocated to port 22) to capture ssh traffic.Connect to **zoltar** again to generate ssh traffic.

What transport layer protocol is used by ssh?

In the captured ssh messages look for the username and password used in the connection. Are you able to read them? Do you consider ssh as a secure protocol?

From the previous exercise we can conclude that the ssh protocol works on the TCP transport protocol. Therefore, a TCP connection is established between client and ssh server. In addition SSH uses encryption techniques that makes the information that travels through the TCP connection goes in a non-readable way, preventing third parties to get the user and password of the connection and what is written during the entire session.

We have also found out that this connection is established between a random client port and server port 22.

4. NETCAT Application

Netcat (often abbreviated to `nc`) is a computer networking utility for reading from and writing to network connections using TCP or UDP.

We can use netcat to establish a TCP connection with a particular server. To do this we will have to identify the destination server appliaction with which we want to connect to. We will do it indicating the server host name, and its port number. We will use netcat as follows:

```
$ nc server_host_name port_number
```

Exercise 5:

Try to connect using netcat to echo server (port 7) on the zoltar server. Check that the dialog with the server works correctly. Note that in this case your netcat client is talking to an echo server and therefore will return everything you send (whatever you type in you keyboard is sent to the server and it sends it back to your screen) . You can cut the communication with the key combination: <Ctrl> <C>.

You can also try connecting to the daytime server (port 13). }

When we (as client) connects to a specific server, client and serve must follow the the protocol of the corresponding application layer. For example, if we make a netcat to the web server of the UPV (`$ nc www.upv.es 80`) the web application protocol, HTTP, will govern the communication between client and server.

Netcat allow us to open a port and start a daemon to listen on this port. To do it, we will use netcat as follows:

```
$ nc -l port_number
```

Netcat also allow us to connect with this daemon specifying the name or IP address of the host where we have opened the port in listen mode, and the port number:

```
$ nc server_name port_number
```

Exercise 6:

Connect with a partner to make a chat between two PCs[1] using netcat. In one of the computers you will make free port in listen mode, being careful not to use a port already used. Use, for example as a port number 9999. On the other you will use netcat as client to connect to that port. Once the connection is made, check that everything written on one computer is received by the other.

Remember that port in listen mode must be ready before the client tries to connect to it.

By default netcat uses the TCP protocol as transport protocol. If you want to use UDP protocol, we need to specify it, adding -u option:

```
$ nc -u -l port_number
$ nc -u server_name port_number
```

Exercise 7:

Repeat exercise 6 but this time using UDP ports.

Could we keep both chats (TCP and UDP) active using the same port number? Justify your answer.

Also, netcat allows to test and debug network applications on the machine itself, To do this we simply have to tell the client that the `server_name` is `localhost`.

You can test it running both, client and server of exercise 6 in your host.