Lab #8: TCP Analysis

Previous reading:

• Unit 4 of the course: Transport Layer. (Kurose: chapter 3)

• Lab #1 of the course: Introduction to Wireshark program.

Lab #8 aim

Lab #8 aim is that the student will be able to understand the workings of TCP transport protocol, studied in the theoretical sessions, through network traffic analysis with the Wireshark analyser.

Introduction

In Lab #8, we are interested in delving into the study of the operation and implementation of TCP without worrying about the data contained in the segments transferred. Therefore, we will indicate to Wireshark that we only want to visualize the TCP segments. The reason is to focus only on the transport information, not on the interpretation that Wireshark can make of the application layer transported by the TCP segments. If you remember the first practices of the course, we analysed HTTP traffic (as we are going to do now) in a comfortable way since the analyser interpreted the data of the TCP segments and indicated on the screen if it was performing, for example, a GET, instead of having to look for that method in the data area of the TCP segment. Now we are not interested in seeing the GET, POST, etc. That is, we are not interested in interpreting the protocol that is used in the data area of a segment at the transport level, but only the format of the TCP segments and the interpretation of their fields. For this reason, in the option **Analyse → Enabled Protocols, we will remove the mark of the HTTP protocol**. In this way, although HTTP is transported in a TCP segment, the analyser will not interpret the HTTP protocol and will only show the TCP segment that transports it.

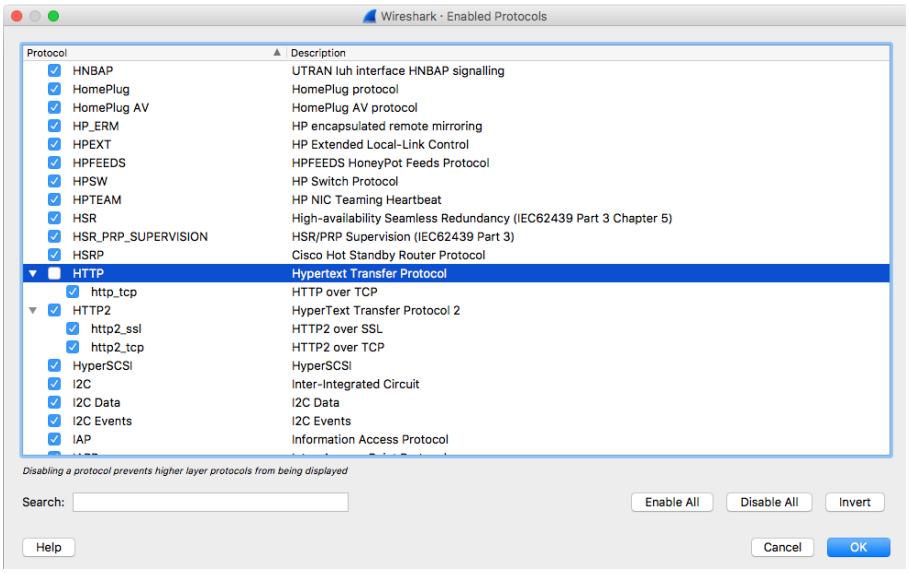

Figure 1. Analyse option -> Enabled Protocols. Uncheck HTTP

TCP Connection Establishment and Termination

In this section, we will analyse the TCP traffic to review the connection establishment and the options included in the SYN segments. Also we will see how each endpoint of the connection use the FIN flag to say that no more data will be sent from that side.

Exercise 1:

Make a capture of the traffic between your browser and the UPV web server (http://www.upv.es) using the Wireshark It may be useful to set a filter to port 80 (from the Capture → Options menu or tcp.port == 80 from the display filter) to facilitate the interpretation of the data.

a) Analyse the connection establishment. Identify the TCP 3-way handshake (SYN,SYN-ACK, ACK) segments. What MSS is chosen to carry out the transfer? What other options establish each end system of the connection?

b) Determines the initial sequence numbers of each end system of the connection. Notice that Wireshark replace the real sequence number by a relative number to make a more comfortable tracking of segments (click on the relative sequence number and you will see the real number that appears in the segment transferred in packets byte pane (the lower window of the main window -contained in hexadecimal). Similarly, by selecting a TCP segment, with the right mouse button (Protocol preferences) we can indicate that we do not want to use relative numbers (although it is more convenient to continue using the relative numbers in the rest of this Lab). (Figure 2)

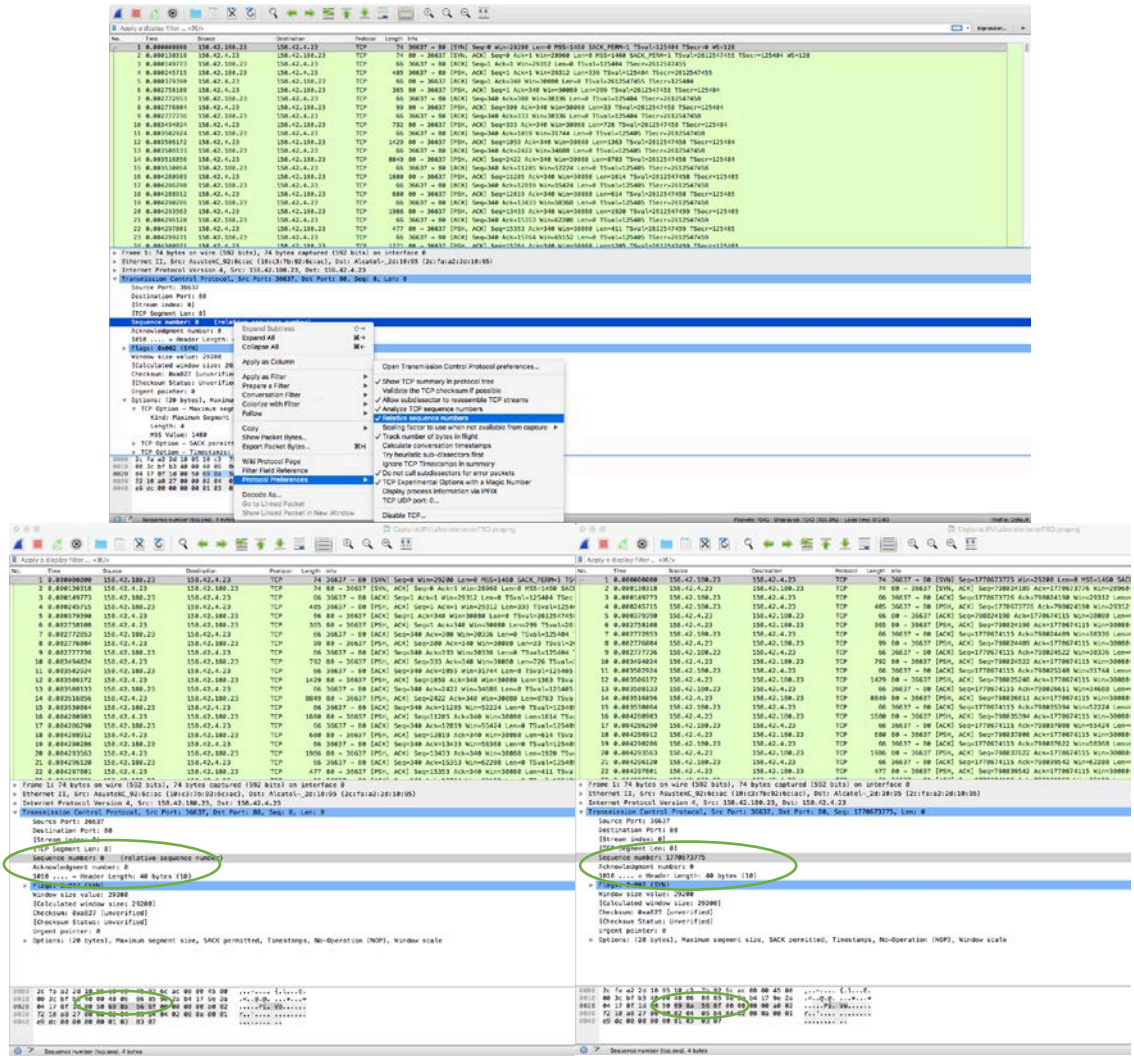

Figure 2. Sequence numbers: relative / absolute

c) Now we want to analyse the connection termination. First we have to select the first TCP flow (the HTML file download flow). You can do it selecting the GET / HTTP/1.1 segment in the packet list and then select the Follow TCP Stream menu item from **Analyze-> Follow TCP Stream** menu (or use the context menu in the packet list). Wireshark will set an appropriate display filter and pop up a window with all the data from the TCP stream laid out in order. Close this window that shows the TCP stream and scroll to the end of the main screen (packet list pane), where you will see the end of the connection. Who takes the initiative? Is it a termination in three or four phases?

Frequency of ACK sendings and Congestion Control Window Size

In this section, we will see how the frequency of ACKs sending can help to the congestion window to grow faster

Exercise 2:

d) Using the same TCP flow as in the Exercise 1.c, execute the option of the **Statistics → Flow Graph** menu. In the new window select the option Show to Display packets (Show: Displayed packets) (Figure 3). What can you see there?

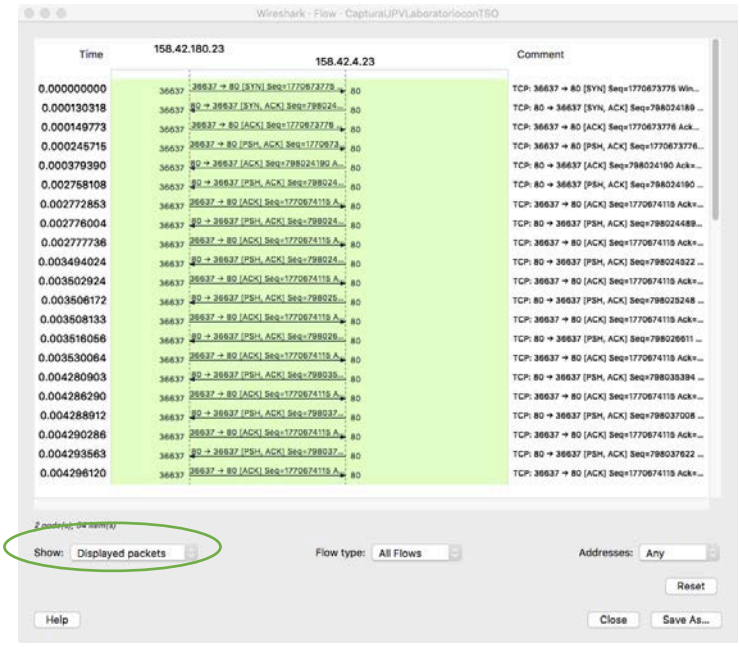

Figura 1.Opción Statistics->Flow Graph

e) Next, analyses the evolution of the acknowledgments (ACK). For this, look at the segments that the web server sends, look its length and its sequence number and observe when the client (your host) sends the ACKs and the acknowledged bytes. This can be seen both in the **Statistics → Flow Graph** window that you have opened in the previous section, and in the main window with the selected flow. Notice that when you select a segment in the **Statistics → Flow Graph** window, it is also selected in the main window. (Note: to know which segment of the server is acknowledged in a client acknowledgment segment, analyse the sequence number of the sender and add the segment length, then look for the ACK that carries that number).

Is there an ACK every two TCP segments? Why? Try to find an explanation (Think about the convenience of using delayed ACK during the first RTT when the slow start algorithm is running).

*During the beginning of a connection, some systems have a mode called **QuickACK**. In this case, the delayed sending of ACK does not start until several segments have been transmitted (for example 16 in some implementations). What do you get with this? The effect is to send an ACK for each segment and then the congestion window grows faster than if the receiver sends an ACK every two segments.*

*However, there are implementations that incorporate the technique called TCP Congestion Control with **Appropriate Byte Counting** (rfc3465). In the systems that implement it, the congestion window is not increased constantly with the reception of each ACK, but it increases as much as the information recognized by the ACK segments. Thus, if only one ACK arrives, but recognizes 2, 3 or 4 segments, the window is increased according to the information recognized. In this way, it is not necessary to send an ACK per segment at the beginning to increase the congestion window faster. These two techniques are not exclusive and we can find implementations that have both (for example, think that a machine can use QuickACK so that the other end -who does not know if it uses Appropriate Byte Counting or not- increase in any case the congestion window faster).*
Finally, it is also interesting to know that in current implementations the number of segments that are sent when a connection is initiated does not always have to be equal to 2 (the rfc5681

standard provides some recommendations). There are studies that determine, in addition, that it would be better to increase the number of segments that are initially transmitted and many systems allow sending up to 10 segments at the beginning of a connection.

RST Flag

In this section, we will analyse the use of the RST flag.

The effect of the RST is to immediately close the connection. This is slightly different from a FIN, which just says that the other endpoint will no longer be transmitting any new data but can still receive some. There are two types of event that cause a RST to be transmitted, one possible event is that the connection is explicitly aborted by the endpoint, e.g. the process holding the socket being killed, and another possible event is that the TCP receives certain kinds of invalid packets, e.g. a non-RST packet for a connection that doesn't exist or has already been closed.

Exercise 3:

f) Make the capture of the traffic generated when your browser requests a web page to the web server of the University of Valencia (http://www.uv.es:81) on port 81. Remember to modify your capture filter to port 81. Analyse the capture made. How does the web server indicate that there is not a service in that port? How is the connection closed? What is the answer of your browser? Does your browser try the connection again? If yes, how many times?

g) Repeat it, but now make the capture of the traffic when your browser requests a web page to the web server of our university: http://www.upv.es:81. Does it happen the same as in the previous case? What do you think is the reasons for this behaviour?

As you can notice they behave differently. Since the port is closed (no service listening on port 81), the server identifies the client connection request as an invalid packet and because of that TCP replies with a RST package. However, ports only actually do this if they are unfiltered. Filtered connections do not reply at all and simply drop the packet. Filtering is usually done by any firewall since it makes attackers jobs harder by providing less information.

Selective Acknowledgement (SACK)

In the connection establishment, we saw that a selective acknowledgment will be use.

The SACK option is used by a receiver to inform the sender that non-contiguous blocks of data have been received and queued. The receiver is awaiting the receipt of data in later retransmissions to fill the gaps in sequence space between these blocks. At that time, the data receiver will acknowledge the data normally by advancing the left window edge in the Acknowledgment Number field of the TCP header.

It is important to understand that the SACK option will not change the meaning of the Acknowledgment Number field, whose value will still specify the left window edge.

Exercise 4:

In this exercise we are going to see how the SACK is implemented. To do this, download the file CapturaPráctica.pcapng that you can find in Poliformat->Practicas and load it in Wireshark.

Analyse the segment number 88 of the main window. What is the value of the ACK? Now, go to the segment 91, independently that Wireshark catalogues it as Window Update, analyses the rest of the information. What is the value of the ACK? Go to the SACK field, What is the receiver indicating?
See the SACK field in the following segments sent from the server to the client (segments 93,95,97,….), and What is going on?