EL4012 Lab 3a: Wireshark

Learning Outcomes

1. Implement Filters and apply any packet field as a filter element
2. Personalize and customize Wireshark
3. Build complex filters and chase channels of transmissions

In this lab, we will dive into the Wireshark app properties to be able to sort out the traffic and make it recognizable in an efficient way.

## Task 1: Explore filters

The modern high-speed internet transmission made possible for every application to send an enormous amount of information across the web making the inbound/outbound traffic difficult to differentiate. Many applications transmit updating messages to their servers of the origin or perform broadcasting messages for updates. Although Wireshark would perform well-capturing data up to 10Mbits/s, it is the user to decide what particular section he/she is after.

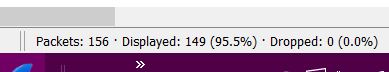
Since the packets are categorized and have many bits fields, Wireshark allows us to visualize the certain type of data in different colours/columns or entirely filtering it out.

1. Once you have started a capture you can filter out information by typing an expression with logical parameters into the text console “Apply a display filter”. The logical expression follows the common programming language syntax and is identical to the C language. Please, see the cheat sheet accompanying this lab for an exhaustive list of possibilities. Start simple by typing:

ip.addr == 10.100.129.31 or your\_machine\_ip

The filter will automatically display only the packets related to this IP address. Don’t forget that your router ARP table updates your machines’ IPs occasionally and can shuffle addresses therefore double check the local IPs each session.

1. The remaining number of packets after filtration is written on the right at the very bottom of your screen.

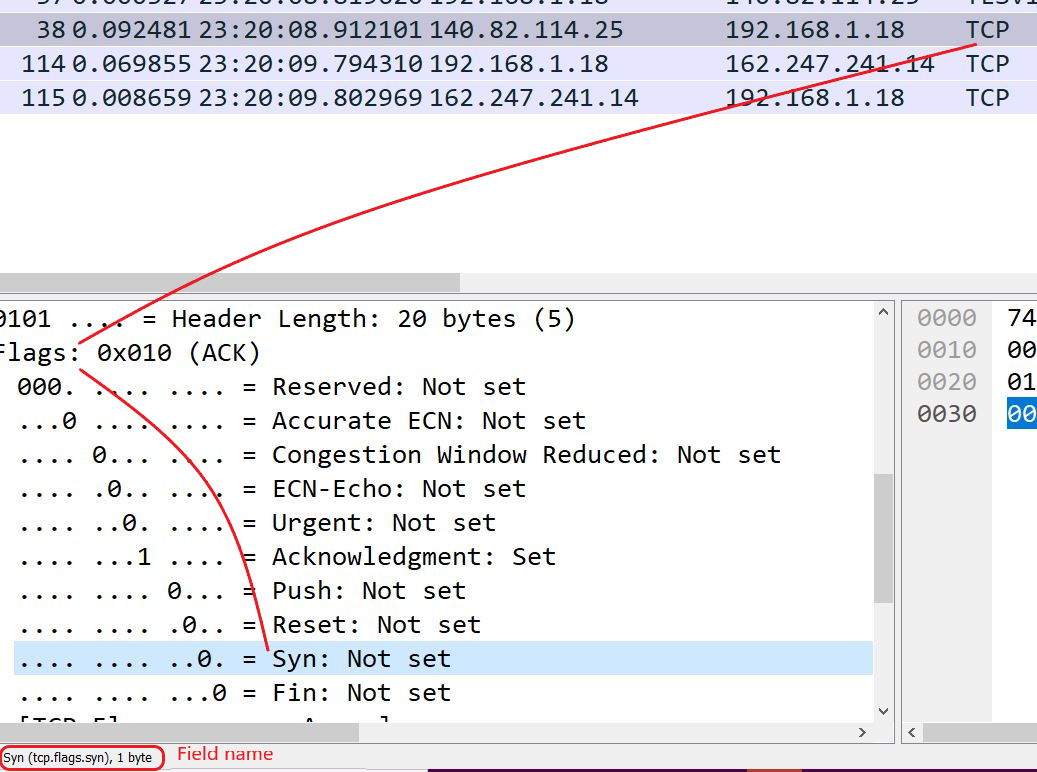


Still, a big chunk of data is visible and can be narrowed down by expanding the filter with ‘and’ && to a particular field of the packet. For example:

Ip.addr == 10.100.129.31 and (tcp)

1. A natural question can be raised how can we know what filtering fields are available? Wireshark makes sure to follow the TCP/IP model although not every connection strictly falls into these categories. If a certain connection has protocols with undefined type Wireshark will make the best effort to fit them into the TCP/IP model. Therefore, the packet details would have similar identical categories but particular subcategories.

Click on Transmission Control Protocol and observe at the very bottom left of the app its field name appears in brackets. Expand the TCP and find the subcategory Flags and expand it further. Find the Syn flag, press it and observe how the app is showing you the field name as an object of a tcp.flags.syn.



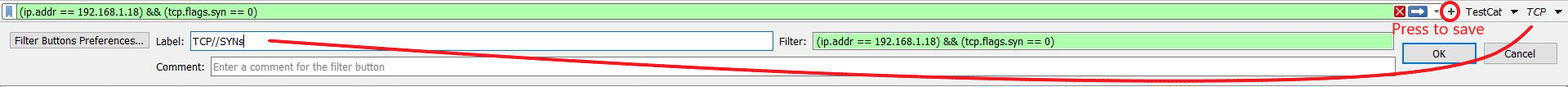
1. Add the new filter to the already existing one with the ‘and’ operator again.

ip.addr == 192.168.1.18 and (tcp) and (tcp.flags.syn == 1)

1. Since the method of typing filters is prone to errors the app can add them for you by right-clicking the field -> Apply as Filter -> … and Selected
2. A multicomponent filter can be reduced and returned back by deleting the section that is not used anymore. The activation of tcp.flags.syn == 1 logic has inherited TCP connections only therefore the (tcp) filter is redundant. Remove the (tcp) part to the final stage of the filter as follows:

ip.addr == 192.168.1.18 and (tcp.flags.syn == 1)

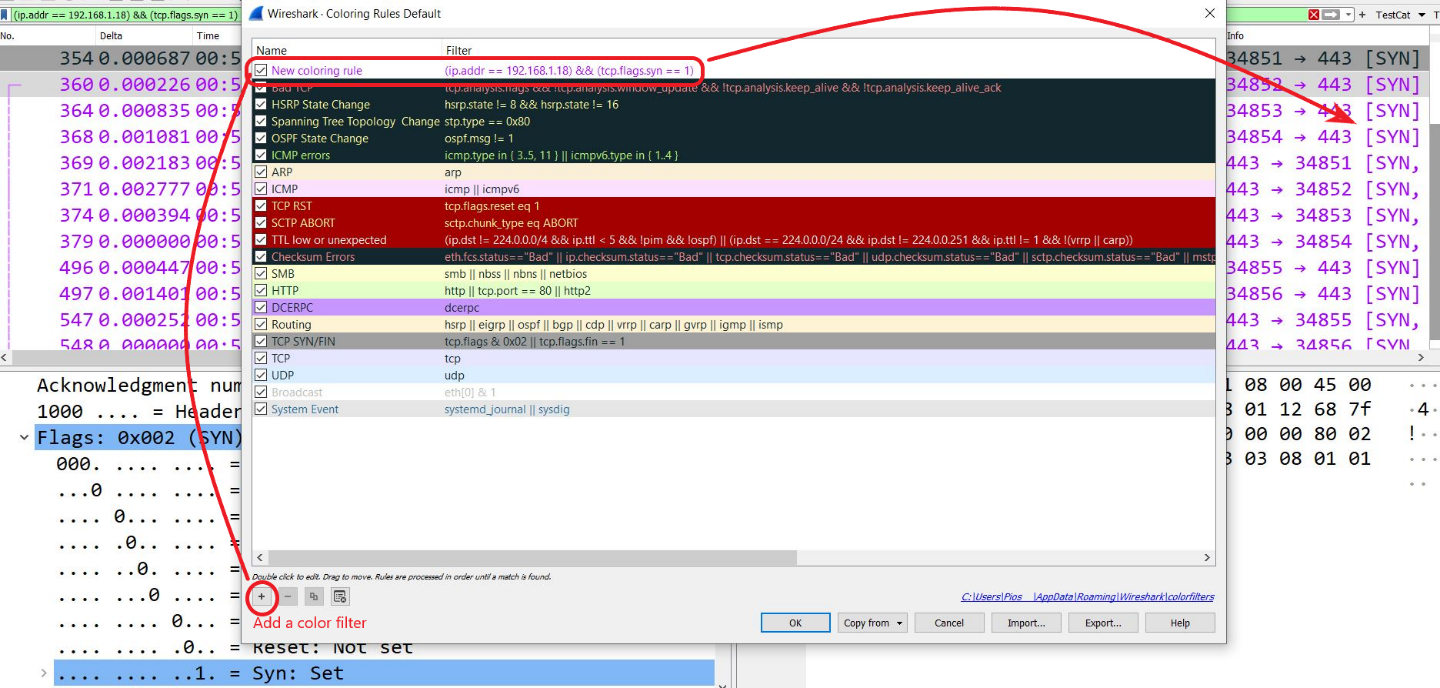
1. The final result will display all packets that are initiating a connection with our computer. Since this filter seems useful for the future, we can save it for later use by pressing the plus ➕ on the right of the text box. Save it under a category of filters TCP by typing TCP//SYNs into the label box as shown.



## Task 2: Personalize

As with every software, there are many preference options to be chosen but we will outline the navigating ones.

1. Go to the tab View -> Coloring Rules and press the plus ➕ to add a new one. Under Filter add the one from Task 1 name it as connection start and choose a distinctive colour as shown.



1. First we have our own rule second we can investigate the other colour rules. Now make sure that under: View -> Colorize Packet List is checked. Observe how on the right side of all columns there is a colour packet timeline for a better understanding of the full capture.
2. Mentioning timeline, the default column about time shows the packet time interval from stream transmission which is not intuitive but could be changed to universal clock time by going into: View -> Time Display Format -> UTC Time of Day The same option allows you to change to millisecond which is more typical in terms of packet transmission.
3. If some of the time options or packet fields are something that you are going to reference often they could be brought as a separate column. Either by right-clicking the field of interest and choosing -> Apply as Column or another way is to go to: Edit -> Preferences -> Appearance -> Columns There you can add or hide any column type you prefer.
4. If the final state of the visualization is appealing to you go to the very bottom right corner, right-click Profile and choose New... . The great outcome is that although our imagination might be limited as a start we can extract other people’s profiles or filter settings, expanding our imagination of availability.
5. Similarly, we can take a look at some of the suggested filters under: the tab Analyze -> Display Filters…

## Task 3: Filters in depth

Apparently to utilize a Wireshark at its maximum a profound knowledge of data communication and networking is required to make the data feasible. Although this is a whole discipline on its own the basic interconnection between theory and filter possibilities will be demonstrated.

1. We are aware that an IP address is broken down into a network part and a host part. By looking into an IP address, we can’t determine its bits distribution to either part, an extra information must be provided in terms of a subnet mask. Wireshark allows us to search for a particular subnet group by providing the number of subnet bits in the following manner:

ip.src = 192.168.5.1/29

/29 are the number of bits of the networking part therefore the number of hosts that would be filtered out are 2(32- maskNum) or 232-29 = 23 = 8 IPs ranging from 192.168.5.0 - 192.168.5.7

For higher convenience an online calculator can be used:

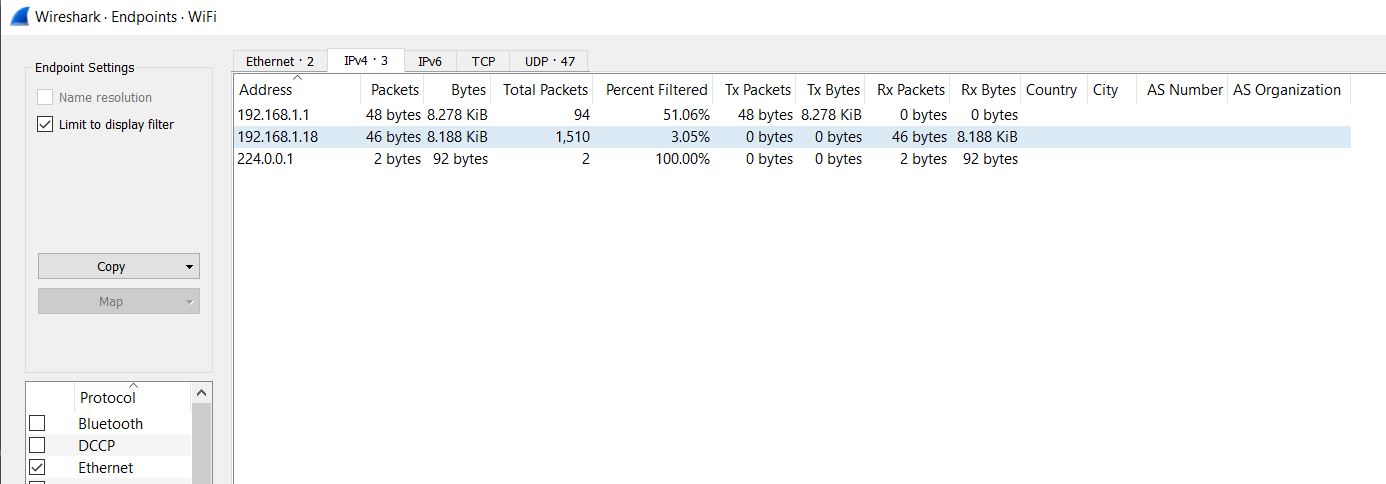
<https://www.subnet-calculator.com/>

If the desired addresses are not sequential in numbering a list can be fed to the formula.

ip.src in { 10.100.129.31 , 10.100.129.129 }

1. If the range of IPs running is high, Wireshark offers unique conversation separation that would save a lot of time enlisting them manually.

Go to tab Statistics -> Endpoints -> IPv4 tab and make sure “Limit to display filter is checked”



Similar results can be found under:

Statistics -> Conversations -> IPv4 tab

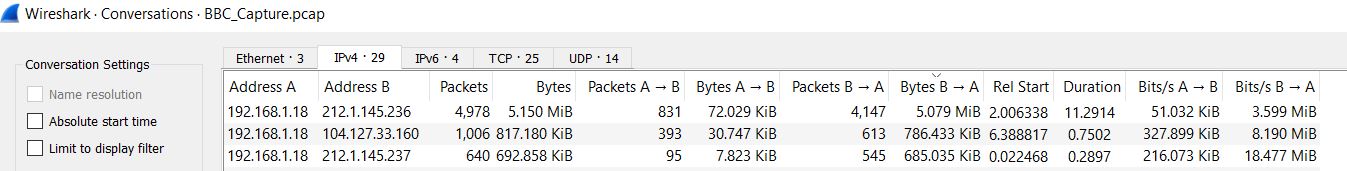
From there further specific byte size transfer sorting can be performed and particular duplet IP-port pair can be extracted as a filter conversation.

1. If we are on the main packet display menu Wireshark can establish a particular stream or conversation for a given packet. Each packet comes in a stream between two MAC or two IP addresses or TCP ports, in essence, the level of communication can be separated by layer. For example, if the host computer communicates to the bbc.co.uk website on one IP address conversation a few TCP channels might be opened.

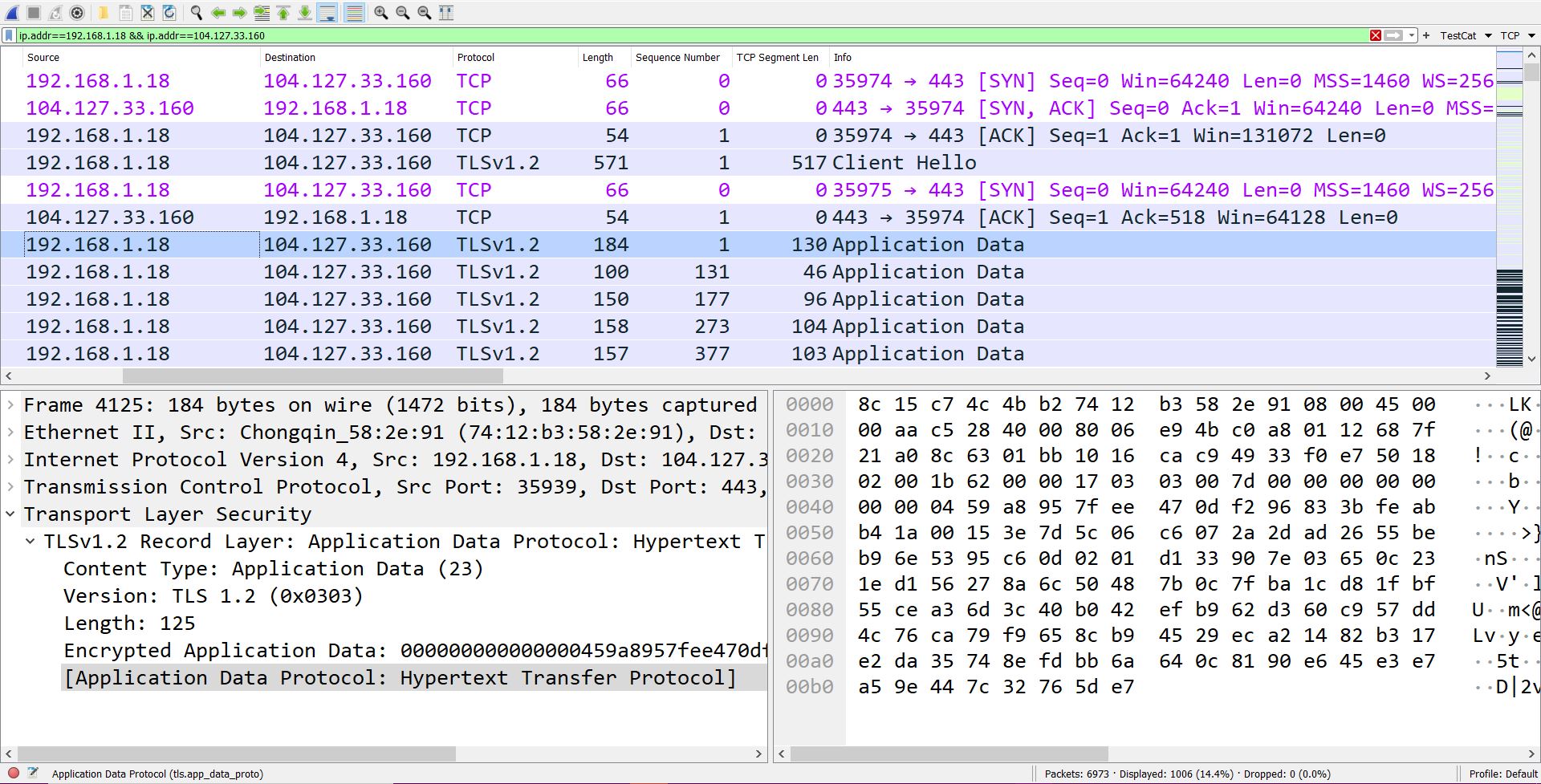
To start practicing on well-known turf open a terminal and a browser and start a new capture in Wireshark. Enter the bbc.co.uk in your browser and ping to bbc.co.uk at the same time, and immediately stop the capturing afterward. The narrow time window will guarantee us success when we try to sort and find inside the packet capture PCAP.

A homemade PCAP with the same preconditions could be found in the Blackboard area. Open it and go to: Statistics -> Conversations -> IPv4 tab

Sort by size as shown in the next picture.



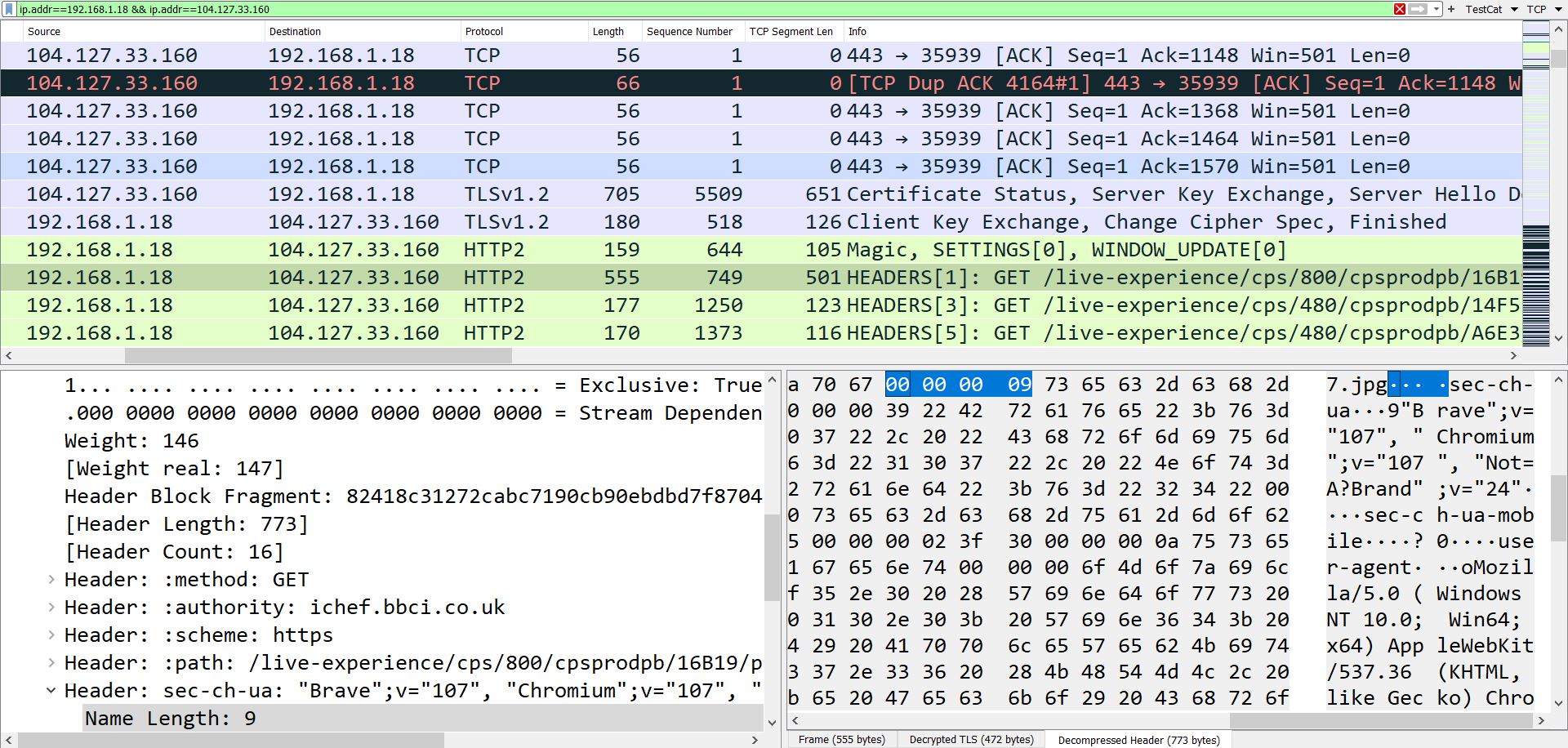
The second most loaded transmission was on 104.127.33.160 which makes sense to be our stream with the BBC. Extract it as a filter and a similar one to the next picture should appear.



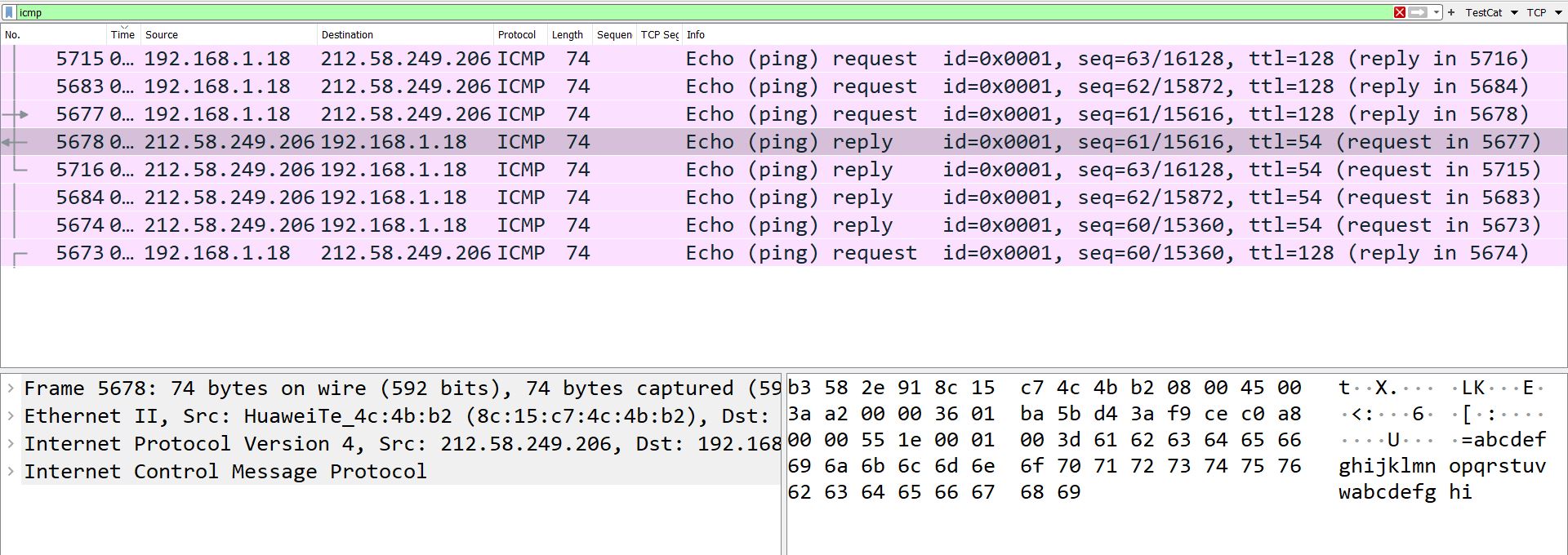
Please, take a look at the three-stage TCP handshake established by the first three packets exchange. Next, a protocol with TSL abbreviations appears which stands for Transport Secure Layer. Almost all connections in today’s web space are through HTTPS and will require sessions key exchange (more on that in the second semester). The exchange is also secured and please observe that the data is actually Encrypted.

A key is created for each session and is stored in our computer and once agreed an actual HTTP2 data starts to appear with an actual GET request from the host. Nowadays, these requests and data won’t be visible but for this PCAP the key of this transmission is also shared with the Wireshark so it can decrypt and visualize the full data exchange above the TCP layer.

Please observe in the next picture, after the two TLSs key packet exchange things are transparent because in the background the software has intercepted, decrypt, and even decompressed the GET request which appears to be browser settings.



1. Finally let’s find out what happened with our ping requests. Make a filter for ICMP messages and the next portion of packets should appear.



Strangely the destination address is different from our previous communication. Hosting a website would include separate IPs for different services which means it would be challenging to establish one clear port or IP to find the full picture. Luckily, we will work on a small-footprint device and much easier to be detected and bring into a display. But the main takeout is that browsing the web and trying to identify all entities and elements might be a time waste with subsidiary knowledge.

As homework investigate the largest stream in the data transmission that we sorted out in: Task 3 section 3: IP-212.1.145.236

What type of connection it is?

Why it is that type of connection?