Department of Electrical and Computer Systems Engineering
Monash University

Information and Networks, ECE3141

Lab 5: Network protocols

Authors: Dr. Mike Biggar
Dr. Gayathri Kongara
(This update: 1 May 2022. Some material from earlier labs by
Ahmet Şekercioğlu & David M^{c}Kechnie)

1 Introduction

In this exercise, we analyse the layered structure of network protocols using a web browsing example. We examine the header structure of the packets at the data link, IP, transport, and application layers. In particular we observe how addresses and port numbers work together to enable end-to-end applications. We learn about addressing at the link layer (MAC addresses) and network layer (IP addresses), study DNS Query and Response, TCP connection setup and HTTP GET and Response messages.

You will recall from lectures that several steps, involving several protocols, are required when we do something like clicking on a URL on a web page; even though it looks very simple to the user! The main steps at the Application and Transport layers are:

1. The web browser needs to establish a connection to the server that holds the selected URL. It knows the URL, but not the actual IP address.
2. It tells DNS to get the IP address for it, and DNS will issue a DNS query to complete this task (unless it happens to already know it – see below). The DNS query is sent in a UDP segment, which is connectionless, so there is no connection setup.
3. The local DNS server might have to contact multiple DNS servers to get the information it needs but, if everything goes well, it will be able to learn the IP address of the server and send it back to the client.
4. The web browser now wants to talk to the web server. It wants to issue HTTP requests, so HTTP now needs to establish a TCP connection with that server. So, TCP starts its 3-way handshake to set up and synchronise the connection.
5. Once the transport layer (TCP) connection is in place, HTTP can issue its "GET" request to obtain and display the requested information[1].
6. When the TCP connection is no longer needed (e.g. the "GET" is complete), it will be closed down.

[1] Of course, a "GET" request is not the only thing that HTTP can do. If the host already has a copy of a requested file in its cache, then HTTP might issue a HEAD command, to get just the header information without the actual file itself, so it can tell if the version it has is up to date. Other HTTP commands (or "methods") might be used, for instance, to return data completed in a form. For this general overview, we just use a simple page request as an example.

7. The requested web page may include lots of other URLs (for advertising, graphical components, information from different sources, etc.), so there may be many more occurrences of DNS queries, TCP connections and HTTP exchanges before the web page is completely built!

Below the Transport Layer, we have the Network and Data Link layers that look after actually sending the packets, but these layers are unaware of the protocols used, whether it is a connection-oriented or connectionless service for the higher layers, or whether transmitted segments are acknowledged. The IP (Network) layer is given some data and an IP address, and told to send the packet on the first "hop" of its journey through the network. This journey would normally involve multiple routers before it gets to its destination. For each of those hops, the Data Link layer (e.g. Ethernet) is responsible for getting that packet across the particular physical connection (e.g. the LAN or the point-to-point link between routers).

In this laboratory, we will use Wireshark to capture data frames arriving from or sent to our computer, explore how the protocols work together to enable communication such as that described above, and confirm that they exchange the header information we expect. You will carry out the following as you investigate these network protocols:

- Use Wireshark to capture frames (and all their contents) exchanged by your PC as you access a web page
- Try to make sense of the fields, queries and responses involved in the exchanges necessary to bring the web data to your PC
- Understand the caching processes involved in IP address lookup and display of web page components
- Improve your understanding of the protocol layering process

There are a lot of questions in this lab, and we ask you to write down a lot of addresses and field contents, but it is the way we get you to focus on and interpret how the data exchanges work. If you persist and do all the steps, you should end up understanding how web pages are obtained! When you've finished, you should have a good understanding of the following:

- What are MAC addresses?
- What happens to the src address field in an ethernet frame once it leaves your computer? What happens to the dst address field in an ethernet frame once it reaches the next router?
- What happens to the src IP address field in an IP packet once it leaves your computer? What happens to the dst IP address field in an IP packet once it reaches the next router?
- What values do 'well-known' port numbers lie between? What values can ephemeral port numbers take?

1.1 DNS caching

To help speed up Web browsing, computers maintains a local table (cache) containing any DNS names and corresponding IP addresses that have been looked up recently (both IPv4 and IPv6). Once the numerical IP address of a URL has been found by querying a DNS server, the information is stored locally in this table.

Whenever your browser wants access to a URL, it will first look in the local cache to see if the host name (in the URL) and its corresponding IP address are already there before querying the DNS server. If it finds the IP address, it is used immediately without losing time by generating a DNS query and waiting for a response. This arrangement should save time and reduce unnecessary network traffic and DNS server load.

Windows DNS cache

You can see the contents of the DNS cache of a Windows PC by opening a Command shell[2] and issuing the following command:

```
ipconfig /displaydns
```

You can also flush (i.e. empty) the contents of this local cache by issuing

```
ipconfig /flushdns
```

MacOS DNS cache

Things are a little more complicated on a Mac. You have to issue a few commands in a Terminal window, and also unlock “private” log data. The following process is suggested, but it may vary a little depending on the version of MacOS X you are running[3]. If you have trouble, Google it.

You must be logged into an Administrator account (unless you’ve done some unusual configuration changes with regard to privileges). Then:

1. Open a “console” window (search for it in Launchpad or Spotlight)
2. Select the Mac you’re using on the left hand side (it could also show your iPhone or other devices), type “any:mdnsresponder” in the search window and hit return.
3. Now open a Terminal window and issue the following commands:

```
sudo log config --mode "private_data:on"
sudo killall -INFO mDNSResponder
```

You will see DNS entries in the Console. There will be a lot of internal housekeeping “stuff”, but hopefully as you scroll through you will see some websites you recognise. The “private data” line is necessary to open up access to data treated as private. Otherwise, by default, all of this information would just appear as lots of lines saying <private>. To return the privacy setting, run the following before you exit:

```
sudo log config --mode "private_data:off"
```

Note that, when you run a task as superuser using “sudo”, you will be prompted for the (Administrator’s account) password. There is no echo as you type the password. The authorisation will be remembered for 5 minutes, during which further sudo commands will not be challenged.

To flush the cache, you need to run the following command. (They can all be put on the one line, separated by the “;”s).

```
sudo killall -HUP mDNSResponder;
sudo killall mDNSResponderHelper;
sudo dscacheutil -flushcache
```

[2] Type Win-R to get a “Run” window, type “cmd” and Enter. Or in Windows 10, type “cmd” in the search field in the bottom left.
[3] This should work for OSX v12 High Sierra onwards.

1.2 Protocol layering

You should now be well familiar with the concept of layered protocols and the different functions carried out at each layer. To help keep it all in mind, though, figure 1 shows the layered Internet structure, and emphasises the fact that each layer/protocol adds its own headers to the set of bits that the Application originally wanted sent. This diagram could be useful as a reminder as we investigate these layers, but it also provides a convenient place to record addresses or fields as you work through this laboratory, so it is clearer where it all fits. For that reason, if it's convenient for you to do so you may like to print Figure 1 before the lab so you can easily write on it.

1.3 MAC ("Physical") addresses

When you do this lab, we will be discussing Network and Data Link layer functions in lectures. Link layer addressing will be a fresh topic for you (it's covered in the week 8 lectures), so this section is intended to provide the background you need for the lab.

When an IP packet is to be delivered over a network (particularly a shared network such as Wi-Fi within a lecture theatre or the Ethernet Switch to which all the PCs in the lab are connected), we obviously need to send the packet to the right destination. Suppose we're accessing a remote web server from our lab PC (connected to an Ethernet switch[4]), so we're sending an HTTP "GET" request inside a TCP segment, inside an IP packet, and that's about to be carried, in turn, inside an Ethernet frame. The PC knows the IP address of its default router, but how does it deliver the Ethernet frame to that router? Remember that the IP address is assigned in software; it might be different next time the router reboots or after some network reconfiguration by the administrator. But we want to send a frame of bits to a hardware interface card. How do we know which interface card is associated with which IP address?

The solution adopted in the Internet is to use a two-layer addressing system. The IP address is software-based and used for routing throughout the inter-network. Your computer network interface (in fact, each network interface it has) is assigned an IP address which will be very similar to those of other hosts on the same network, because they are assigned as a group to that network and share many bits in common. The network interface has a hardware, or "physical" address[5] (also known as a "MAC" – Medium Access Control address) which is globally unique, "burned" into the hardware and ordinarily never changed. It is very important to understand what "globally unique" means. This is NOT the same as "global addressing". When we send a packet from our computer to a web server in London (for example), we do not know, we don't use and are not aware of the MAC address of a router in the UK through which it will pass. We don't even know the MAC address of the ultimate recipient of our request (the web server). We use the hierarchically-assigned (remember subnetting?) IP addresses for this networking. It is only on each individual link that we are concerned with these MAC addresses, but we need to be sure that there can only ever be one device on our local network with a given MAC address. THAT's why network interface hardware is manufactured and sold already set up with their MAC address, and there will only ever be one interface with that MAC address anywhere on the planet[6]. We are ONLY

[4] What is an Ethernet Switch? It's the box into which a collection of computers is plugged via their respective (familiar, commonly blue) Ethernet cables. The switch behaves like a shared medium network, in that all the connected devices can see one another, and they can send packets "on-link" to any of the others. But the switch looks only at the Data Link layer; it does not even look inside the IP packet, and does not know about or use the IP addresses.

[5] Careful to avoid any confusion here. Though it has the name "physical address", this is not concerned with the "physical layer" of the protocol stack. The Physical Address is definitely part of the Data Link layer but gets its name because it is associated with the actual hardware that does the transmission and reception of signals.

[6] In fact, we don't need to be so restrictive. The computers on the International Space Station would also have unique MAC addresses. ☺

concerned with the MAC addresses of the devices on our immediate network that we can see directly. If we want to send a message outside that network, then we send our IP packet (inside an Ethernet frame) to the MAC address of the router. The router would then strip off all of the Ethernet frame and delete the source and destination MAC addresses as it then creates a new data link layer frame to forward our IP packet to the next router. The source and destination IP addresses would be unchanged in the IP packet contained in that frame, but the source and destination MAC addresses would be new and used only for that next hop, after which the process would be repeated.

So, if we are only aware of MAC addresses within our local network, why is it important that they be "globally unique"? It means that, if we know that our network interface is the only one in the world (or beyond) with its MAC address, then we can connect any collection of devices into the one local network, confident that there can never be any duplication of addresses. This is true even behind a NAT box, where the private IP address given to your computer in your home might be the same as that used by the air-conditioning controller in my home, but the MAC addresses will be unique.

In principle, MAC addresses can have a different format with different network technologies (e.g. a token ring network could use a different addressing mechanism to a Wi-Fi network), but in practice the 48-bit address format defined by the IEEE is used almost universally. We will learn more about the structure and use of these addresses in lectures, but for this lab we need to at least be familiar with their form. IEEE MAC addresses are normally represented as 6 pairs of hexadecimal digits[7], such as 1F-C3-36-14-EB-02, and you can find the MAC address of your computer interface card among the information you get when you type *ipconfig/all* in a command window on Windows, or via the System Report on a Mac[8].

The one bit of the "puzzle" that you have probably realised is missing is how we map IP addresses to MAC addresses. That is, if I know that I want to send my IP packet to a particular destination IP address, how do I find the MAC address of the client or router so the data link layer can deliver it? The answer to this is also covered in week 8 lectures but, if you haven't been through them all yet, just accept that such a mechanism does exist[9].

Important note on the installation and use of Wireshark. Just as with Lab 4, this lab requires the use of Wireshark to reveal network addressing and to analyse the contents of the various protocol layers. However, again, we do not want to force you to install and use this software if you are reluctant to do so or are uncomfortable about sharing network addresses. If this is the case for you, we recommend that you again use the provided recording (see Moodle) of Wireshark data capture sessions, and you can use the results obtained there to complete the requested tasks.

The rest of this laboratory guide is written assuming that you are running Wireshark live.

2 Pre-lab

Prior to the laboratory class, you must read through this entire laboratory description and complete the pre-lab online quiz. Failure to do so will not only mean loss of marks for the quiz, but you will also struggle to understand the principles under investigation and will almost certainly fail to complete the laboratory on time.

[7] Each hexadecimal character represents 4 bits, so this means that the 6-pair addresses contain 6 x 2 x 4 = 48 bits, as stated earlier. 2^{48} = 281,474,976,710,656, which is enough addresses (for now!) to ensure that each is globally unique.
[8] If you've forgotten, see Lab 4 for details of where to find it.
[9] For those whose curiosity gets the better of them, you could read about ARP (Address Resolution Protocol), which is the solution to this problem for IPv4 addresses.

You must be confident about the various protocol layers to complete this lab, and be familiar with the MAC addressing introduced in Section 1.3 above. If you leave it until the start of the lab to start sorting these things out, you will certainly stand no chance of finishing.

You will need to do Wireshark packet captures in this lab, as you learned to do in lab 4. If you have forgotten how to start and stop packet captures and filter the resultant captures to find what you're interested in, go back and have another look at the start of Laboratory 4 ("Probing the Network").

3 Preparing your computer for the lab exercises

3.1 Emptying the DNS cache

As discussed in the introduction, before a browser can access a web server, it needs to know what numerical internet address corresponds to the machine which hosts the URL that it is trying to access. DNS looks after this, and it is the first thing we will start looking for.

We want to generate and capture DNS packets, but you'll remember from lectures that computers cache DNS data to reduce unnecessary network queries, so we need to first force our computer to ask for the IP address of the host names from the DNS server by emptying the DNS cache.

Before continuing, flush the DNS cache by following the steps described in Section 1.1.

3.2 Filtering in Wireshark

As a reminder from your introduction to Wireshark in lab 4, you can filter just about anything that a packet can tell you by creative use of the search field, ANDing and ORing any number of search conditions you like. For example:

```
ip.addr == 127.0.0.1

ip.dst == 127.0.0.1

tcp.port == 80 || udp.port == 80        (OR operation)

ip.src == 127.0.0.1 && ip.dst == 233.54.54.1 (AND operation)

tcp.flags.fin==1
```

You can also right-click on any packet and use it as the basis for a filter ("Apply as Filter").

3.3 Relative sequence numbers

When a connection is established in TCP using the three-way handshake, an Initial Sequence Number (ISN) is sent along with a "SYN" request. The ISN is basically a random 32-bit number (it is randomised for security reasons – so it is hard to pretend to be someone else's host machine). By default, Wireshark tries to make it easy for us by subtracting the ISN from all future sequence (and acknowledgement) numbers, so it looks like the ISN was zero.

If you want, you can see the actual sequence numbers by changing your preferences:

Edit > Preferences > Protocols > TCP > uncheck "Relative Sequence Numbers"

but if you do this you could be dealing with some very large numbers!

3.4 HTTP websites

As mentioned during lab 4, most websites are nowadays “secured”, in that they use the secure “HTTPS” protocol instead of “HTTP”. Even when we select or type URLs that start “http://....” we are often redirected to secure versions of the same thing (https://....). (We will discuss these secure protocols when we cover “Network Security” at the end of the semester.)

In this lab we want to understand as much as possible what is going on inside the various layers of packet encapsulation, and that means we cannot choose to analyse visits to (for example)

https://www.youtube.com/
https://my.monash/campusm/home#menu

Wireshark cannot decrypt or ‘sniff’ these! (It can, of course, see the encapsulation in IP datagrams and data link layer frames, but it cannot interpret what is going on at the Transport Layer or above.)

Fortunately, there are still some “unsecured” websites that use http which we can use, including

http://www.bom.gov.au/
http://www.baidu.com/

http websites can be quite difficult to find, but perhaps you will come across some others (or found some in lab 4) to add to this list.

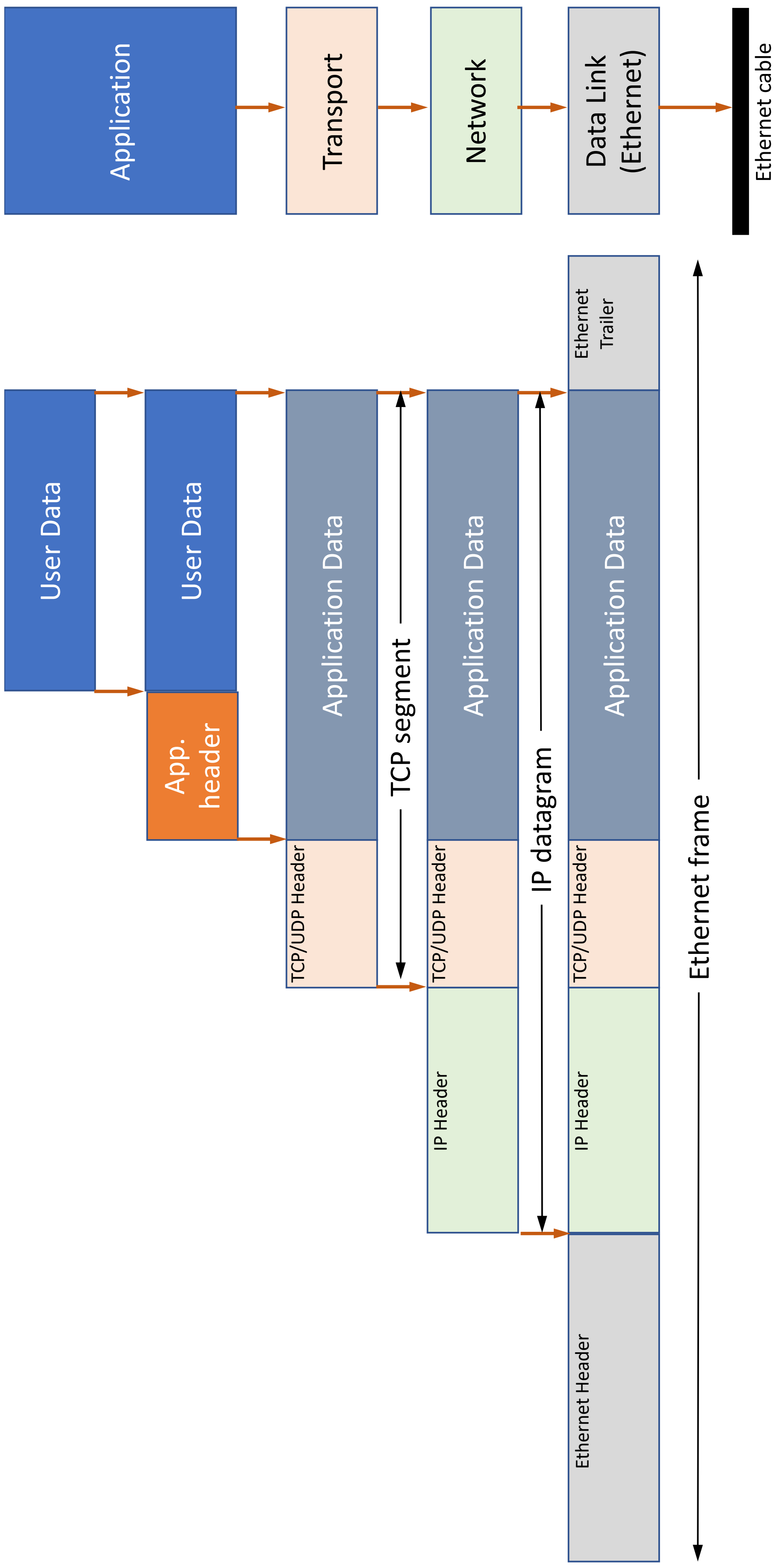


Figure 1. Layered Internet protocol structure when sending frames over Ethernet.

4 Wireshark capture

a) Get ready to start a Wireshark packet capture, but don't start the capture yet. (Review Laboratory 4 if you don't remember how to use Wireshark.)

b) Capture the exchange involved in loading a web page.
   - Start a Web browser, **clear the browser history**, and enter the URL[10] of an HTTP website of your choice (taking note of Section 3.4 above), but **do not press** the Enter key yet.
   - Start Wireshark packet capture.
   - Access the website by pressing the Enter key in the browser web page.
   - Once the page is loaded, stop the capture operation and save the capture file.

To answer the following questions, you can either keep Wireshark open and look through the packets it has just captured or open the stored version you just saved[11].

c) First, examine the protocol column in the top pane of the window. Confirm that you have captured DNS, TCP and HTTP packets.

As we are in a transition period (that will take a long time to complete) between IPv4 and IPv6, you may (depending on your computer and the network to which it is connected) find that two DNS queries are made for each URL, first for IPv4 and then for IPv6 address information[12]. You may also see a request for an IPv6 address that is made by a DNS query carried inside an IPv4 packet. There is nothing wrong with this; we're just asking the DNS name server for some information from its database.

5 Encapsulation

a) Examine the frame for the first DNS packet sent by the client (i.e. your computer)[13]. You could sort alphabetically by the protocol field, but it's generally less confusing if we don't reorder the captured frames. Instead, type dns (lower case) into the filter to find it. Then answer the following questions:
   i) Have a look at the Ethernet (MAC)[14] and IP addresses of the source of this packet. Do these addresses align with what you find when you inspect the addresses for your computer (see Section 1.3)? Add these addresses to the diagram in figure 1, indicating the header in which they are found.

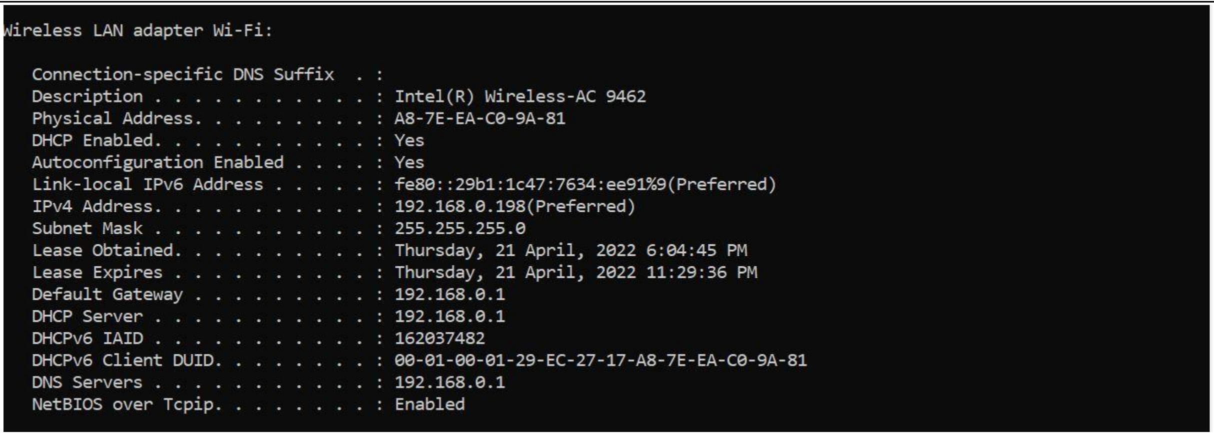
[10] You should actually type it in; don't use a search engine to find it or we could end up with confusion between search engine connections and actual web server connections

[11] The stored file is just useful to have in case you mess something up, accidentally close Wireshark, or forget to do something and have to go back. Of course, the data will be identical.

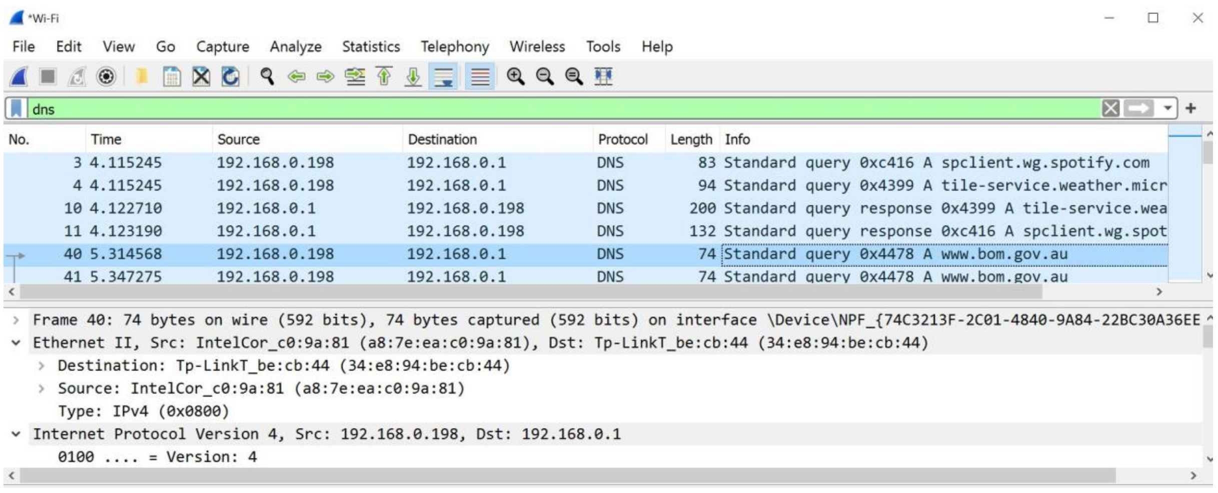
[12] You'll see in the Wireshark "Info" field the abbreviation "A", which means a request for an IPv4 address, and "AAAA", which means IPv6.

[13] Note that the remainder of this lab document is written assuming you are connecting to the network using an Ethernet cable. If you are using WiFi, some of the fields may appear a bit different, but you should still be able to interpret the information we are seeking.

[14] As we will learn in lectures, groups of MAC addresses with the first three bytes fixed are assigned to manufacturers of Network Interface hardware. As a result, knowing the first three bytes will tell you who manufactured the NIC. You will see that Wireshark looks up the database and tells you who the manufacturer is when it presents MAC addresses.



```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) Wireless-AC 9462
   Physical Address. . . . . . . . . : A8-7E-EA-C0-9A-81
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::29b1:1c47:7634:ee91%9(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.0.198(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Thursday, 21 April, 2022 6:04:45 PM
   Lease Expires . . . . . . . . . . : Thursday, 21 April, 2022 11:29:36 PM
   Default Gateway . . . . . . . . . : 192.168.0.1
   DHCP Server . . . . . . . . . . . : 192.168.0.1
   DHCPv6 IAID . . . . . . . . . . . : 162037482
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-29-EC-27-17-A8-7E-EA-C0-9A-81
   DNS Servers . . . . . . . . . . . : 192.168.0.1
   NetBIOS over Tcpip. . . . . . . . : Enabled
```


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns

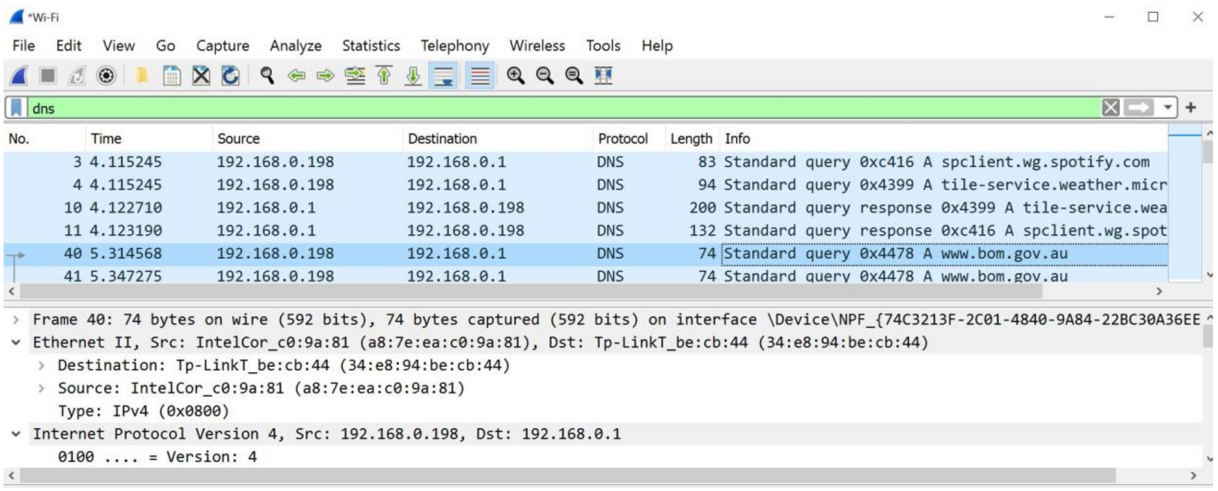
```
No.  Time      Source         Destination    Protocol Length Info
  3 4.115245   192.168.0.198  192.168.0.1    DNS       83 Standard query 0xc416 A spclient.wg.spotify.com
  4 4.115245   192.168.0.198  192.168.0.1    DNS       94 Standard query 0x4399 A tile-service.weather.micr
 10 4.122710   192.168.0.1    192.168.0.198  DNS      200 Standard query response 0x4399 A tile-service.wea
 11 4.123190   192.168.0.1    192.168.0.198  DNS      132 Standard query response 0xc416 A spclient.wg.spot
 40 5.314568   192.168.0.198  192.168.0.1    DNS       74 Standard query 0x4478 A www.bom.gov.au
 41 5.347275   192.168.0.198  192.168.0.1    DNS       74 Standard query 0x4478 A www.bom.gov.au

> Frame 40: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{74C3213F-2C01-4840-9A84-22BC30A36EE
v Ethernet II, Src: IntelCor_c0:9a:81 (a8:7e:ea:c0:9a:81), Dst: Tp-LinkT_be:cb:44 (34:e8:94:be:cb:44)
   > Destination: Tp-LinkT_be:cb:44 (34:e8:94:be:cb:44)
   > Source: IntelCor_c0:9a:81 (a8:7e:ea:c0:9a:81)
     Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 192.168.0.198, Dst: 192.168.0.1
     0100 .... = Version: 4
```

Yes, the addresses will align with the addresses from ipconfig/all. We can see that the MAC Address are “A8-7E-EA-C0-9A-81”.

ii) What is the content of the *type* field in the Ethernet frame? What action will be taken by the Ethernet protocol at the receiving end as a result of this?

The content of the type field in the Ethernet frame will indicate or show whether it encapsulates IPV4 or IPV6. For our case, it will be an IPV4 (0x0800) datagram. It can be used to determine how the payload of the frame will be processed at the receiving end.

iii) What is the destination Ethernet address? Add this also to figure 1.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns

```
No.   Time       Source          Destination     Protocol  Length Info
    3 4.115245   192.168.0.198   192.168.0.1     DNS           83 Standard query 0xc416 A spclient.wg.spotify.com
    4 4.115245   192.168.0.198   192.168.0.1     DNS           94 Standard query 0x4399 A tile-service.weather.micr
   10 4.122710   192.168.0.1     192.168.0.198   DNS          200 Standard query response 0x4399 A tile-service.wea
   11 4.123190   192.168.0.1     192.168.0.198   DNS          132 Standard query response 0xc416 A spclient.wg.spot
   40 5.314568   192.168.0.198   192.168.0.1     DNS           74 Standard query 0x4478 A www.bom.gov.au
   41 5.347275   192.168.0.198   192.168.0.1     DNS           74 Standard query 0x4478 A www.bom.gov.au

> Frame 40: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{74C3213F-2C01-4840-9A84-22BC30A36EE
v Ethernet II, Src: IntelCor_c0:9a:81 (a8:7e:ea:c0:9a:81), Dst: Tp-LinkT_be:cb:44 (34:e8:94:be:cb:44)
  > Destination: Tp-LinkT_be:cb:44 (34:e8:94:be:cb:44)
  > Source: IntelCor_c0:9a:81 (a8:7e:ea:c0:9a:81)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 192.168.0.198, Dst: 192.168.0.1
    0100 .... = Version: 4
```

As we can see in the figure above, the destination Ethernet address will be 34:e8:94:be:cb:44

iv) What is the destination IP address? Is it an IPv4 or IPv6 address? Add this also to figure 1.

The destination IP address will be 192.168.0.1. It is an IPv4 address.

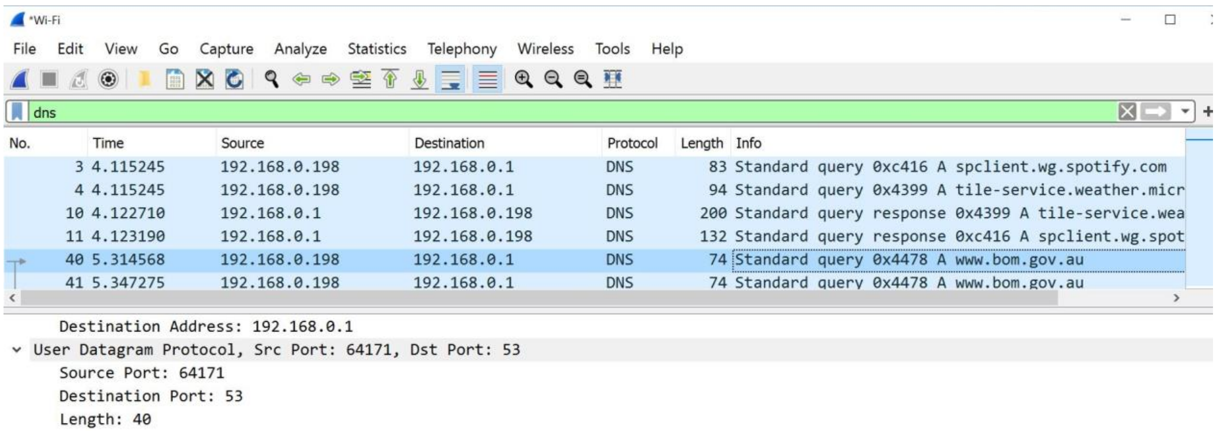
b) Looking at the same DNS packet, we'll go "up a layer"; examine the IP header for this packet to answer the following questions. Note that the entire IP datagram is contained inside the Ethernet frame. How does the IP layer at the receiver (i.e. the DNS server) know what to do with the contents of the datagram? If the packet is IPv4, what is the value of the "Protocol" field and how is it used? If IPv6, what is the Next Header (IPv6) field and how is it used?

In the first header field, the first 4 bits will indicate the version of the IP (Internet Protocol) used. For example, IPv4 will have a value of 4 (0b0100) and IPv6 will have a value of 6 (0b0110).

The IP layer will check the "Protocol" field to determine which Transport Layer Protocol it will allow to pass the received IP datagram onto after stripping off the network layer headers.

c) For the same packet, let's go up another layer; examine the UDP header of this first DNS packet sent by the client to answer the following questions:

i) Identify the client port number. What is it? Is the number significant?



*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns

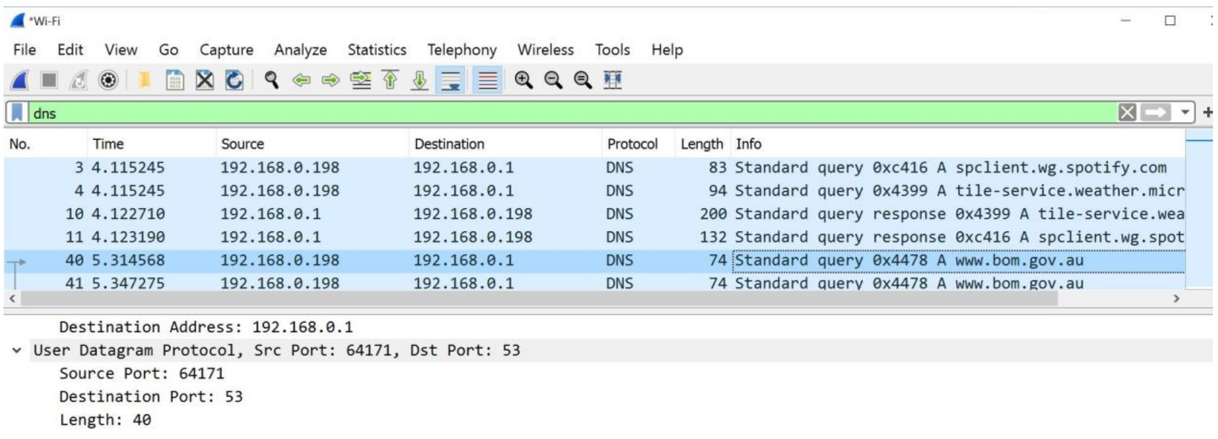
```
No.  Time      Source         Destination    Protocol Length Info
  3 4.115245   192.168.0.198  192.168.0.1    DNS       83 Standard query 0xc416 A spclient.wg.spotify.com
  4 4.115245   192.168.0.198  192.168.0.1    DNS       94 Standard query 0x4399 A tile-service.weather.micr
 10 4.122710   192.168.0.1    192.168.0.198  DNS      200 Standard query response 0x4399 A tile-service.wea
 11 4.123190   192.168.0.1    192.168.0.198  DNS      132 Standard query response 0xc416 A spclient.wg.spot
 40 5.314568   192.168.0.198  192.168.0.1    DNS       74 Standard query 0x4478 A www.bom.gov.au
 41 5.347275   192.168.0.198  192.168.0.1    DNS       74 Standard query 0x4478 A www.bom.gov.au

    Destination Address: 192.168.0.1
  User Datagram Protocol, Src Port: 64171, Dst Port: 53
    Source Port: 64171
    Destination Port: 53
    Length: 40
```

The client port number will be 64171. It is just a random or ephemeral number that is assigned for a short period of time. This number is not significant (assigned at random to unknown clients to identify the process for one connection).

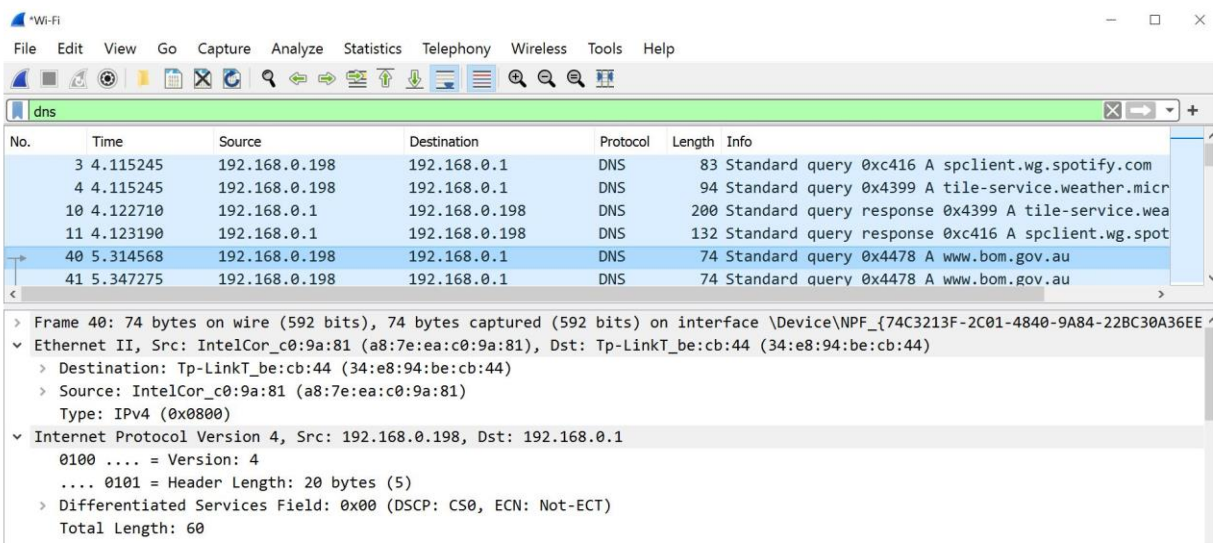
ii) What is the destination port number? What is the significance of this port number? Does it help to identify the application layer protocol of the payload?

The destination port number is 53. This port number helps to identify the application layer protocol (DNS). Yes, it helps to identify the application layer protocol of the payload.

iii) Check that the length field in the UDP header is consistent with the IP header length information.



```
dns
No.   Time      Source         Destination    Protocol Length Info
    3 4.115245  192.168.0.198  192.168.0.1    DNS        83 Standard query 0xc416 A spclient.wg.spotify.com
    4 4.115245  192.168.0.198  192.168.0.1    DNS        94 Standard query 0x4399 A tile-service.weather.micr
   10 4.122710  192.168.0.1    192.168.0.198  DNS       200 Standard query response 0x4399 A tile-service.wea
   11 4.123190  192.168.0.1    192.168.0.198  DNS       132 Standard query response 0xc416 A spclient.wg.spot
   40 5.314568  192.168.0.198  192.168.0.1    DNS        74 Standard query 0x4478 A www.bom.gov.au
   41 5.347275  192.168.0.198  192.168.0.1    DNS        74 Standard query 0x4478 A www.bom.gov.au

    Destination Address: 192.168.0.1
  User Datagram Protocol, Src Port: 64171, Dst Port: 53
    Source Port: 64171
    Destination Port: 53
    Length: 40
```



```
dns
No.   Time      Source         Destination    Protocol Length Info
    3 4.115245  192.168.0.198  192.168.0.1    DNS        83 Standard query 0xc416 A spclient.wg.spotify.com
    4 4.115245  192.168.0.198  192.168.0.1    DNS        94 Standard query 0x4399 A tile-service.weather.micr
   10 4.122710  192.168.0.1    192.168.0.198  DNS       200 Standard query response 0x4399 A tile-service.wea
   11 4.123190  192.168.0.1    192.168.0.198  DNS       132 Standard query response 0xc416 A spclient.wg.spot
   40 5.314568  192.168.0.198  192.168.0.1    DNS        74 Standard query 0x4478 A www.bom.gov.au
   41 5.347275  192.168.0.198  192.168.0.1    DNS        74 Standard query 0x4478 A www.bom.gov.au

> Frame 40: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{74C3213F-2C01-4840-9A84-22BC30A36EE
v Ethernet II, Src: IntelCor_c0:9a:81 (a8:7e:ea:c0:9a:81), Dst: Tp-LinkT_be:cb:44 (34:e8:94:be:cb:44)
  > Destination: Tp-LinkT_be:cb:44 (34:e8:94:be:cb:44)
  > Source: IntelCor_c0:9a:81 (a8:7e:ea:c0:9a:81)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 192.168.0.198, Dst: 192.168.0.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
```

As we can see from the above two graphs, the length field in the UDP header is consistent with the IP header length information. The length field is 40. Since the IP header length is 20 bytes, the IP datagram will have a total length of 60.

6 DNS

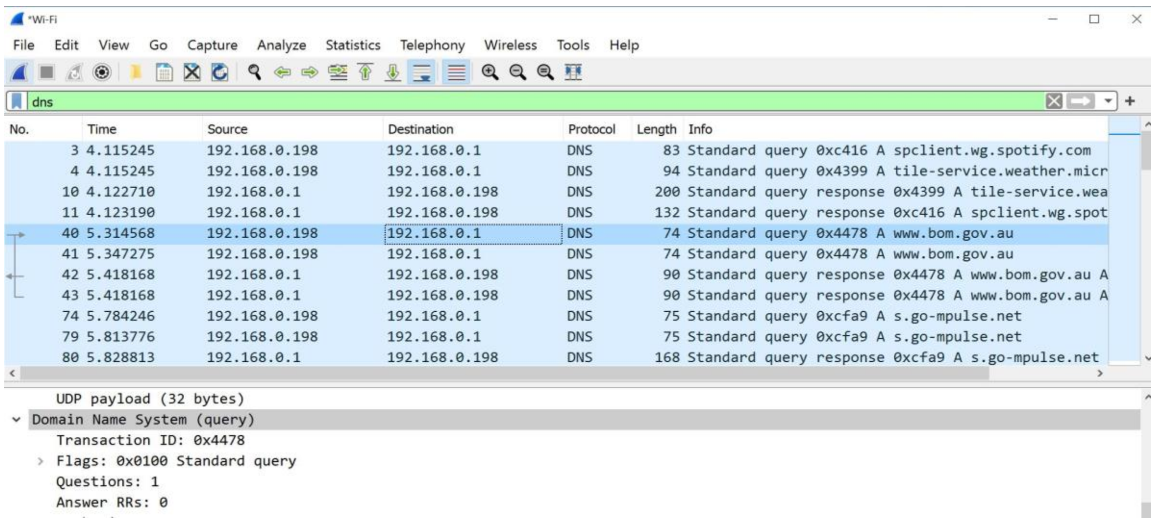
a) Examine the DNS query message in both[15] DNS packets sent by the client to answer the following questions.

[15] We are assuming here that there are two consecutive DNS queries, first for IPv4 and then IPv6 addresses, as discussed earlier. Your particular computer and network may behave a little differently.

i) (Other than by checking IP addresses and the order in which packets were sent) how do we know if this DNS message is a query or a response?

We can know if this DNS message is a query or a response by checking the extra field “answer” in the response packet. Wireshark itself will also inform the user whether the packet is a DNS query or response.

ii) What is the query transaction ID (inside the DNS query)? What value is this?


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns

```
No.  Time      Source         Destination    Protocol Length Info
   3 4.115245  192.168.0.198  192.168.0.1    DNS       83 Standard query 0xc416 A spclient.wg.spotify.com
   4 4.115245  192.168.0.198  192.168.0.1    DNS       94 Standard query 0x4399 A tile-service.weather.micr
  10 4.122710  192.168.0.1    192.168.0.198  DNS      200 Standard query response 0x4399 A tile-service.wea
  11 4.123190  192.168.0.1    192.168.0.198  DNS      132 Standard query response 0xc416 A spclient.wg.spot
  40 5.314568  192.168.0.198  192.168.0.1    DNS       74 Standard query 0x4478 A www.bom.gov.au
  41 5.347275  192.168.0.198  192.168.0.1    DNS       74 Standard query 0x4478 A www.bom.gov.au
  42 5.418168  192.168.0.1    192.168.0.198  DNS       90 Standard query response 0x4478 A www.bom.gov.au A
  43 5.418168  192.168.0.1    192.168.0.198  DNS       90 Standard query response 0x4478 A www.bom.gov.au A
  74 5.784246  192.168.0.198  192.168.0.1    DNS       75 Standard query 0xcfa9 A s.go-mpulse.net
  79 5.813776  192.168.0.198  192.168.0.1    DNS       75 Standard query 0xcfa9 A s.go-mpulse.net
  80 5.828813  192.168.0.1    192.168.0.198  DNS      168 Standard query response 0xcfa9 A s.go-mpulse.net

    UDP payload (32 bytes)
  Domain Name System (query)
    Transaction ID: 0x4478
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
```

The query transaction ID (inside the DNS query) can be said to be a 16-bits random value chosen by the client and is used by the DNS for tracking queries and responses.

The value is 0x4478

b) Now consider the packet that carries the DNS response to the above query[16] and answer the following questions:

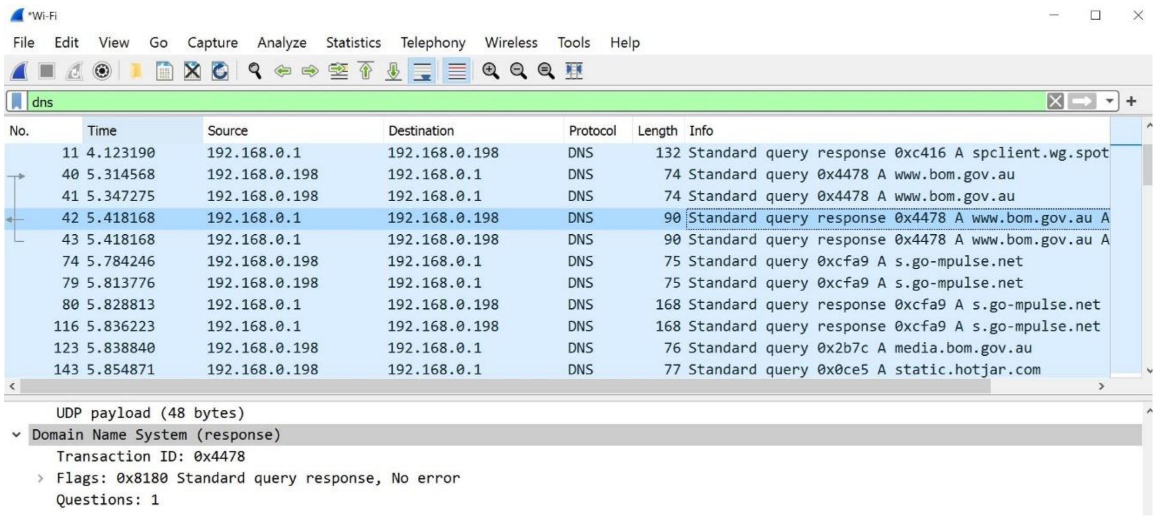
i) Are the IP addresses for this packet what you would expect? Why?

Yes, the IP addresses for this packet is what I would expect. We are receiving response form the address which was the destination address in the first packet. The queries were

[16] Note that Wireshark tells you which packet that is.

sent to the Google Public DNS. The Google DNS server would recursively query the storage location of the requested resources and send a response to the client.

ii) Confirm that the transaction ID in the response message is correct. What is "correct"?


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns

```
No.   Time       Source          Destination     Protocol Length Info
   11 4.123190   192.168.0.1     192.168.0.198   DNS      132 Standard query response 0xc416 A spclient.wg.spot
   40 5.314568   192.168.0.198   192.168.0.1     DNS       74 Standard query 0x4478 A www.bom.gov.au
   41 5.347275   192.168.0.198   192.168.0.1     DNS       74 Standard query 0x4478 A www.bom.gov.au
   42 5.418168   192.168.0.1     192.168.0.198   DNS       90 Standard query response 0x4478 A www.bom.gov.au A
   43 5.418168   192.168.0.1     192.168.0.198   DNS       90 Standard query response 0x4478 A www.bom.gov.au A
   74 5.784246   192.168.0.198   192.168.0.1     DNS       75 Standard query 0xcfa9 A s.go-mpulse.net
   79 5.813776   192.168.0.198   192.168.0.1     DNS       75 Standard query 0xcfa9 A s.go-mpulse.net
   80 5.828813   192.168.0.1     192.168.0.198   DNS      168 Standard query response 0xcfa9 A s.go-mpulse.net
  116 5.836223   192.168.0.1     192.168.0.198   DNS      168 Standard query response 0xcfa9 A s.go-mpulse.net
  123 5.838840   192.168.0.198   192.168.0.1     DNS       76 Standard query 0x2b7c A media.bom.gov.au
  143 5.854871   192.168.0.198   192.168.0.1     DNS       77 Standard query 0x0ce5 A static.hotjar.com
```

```
    UDP payload (48 bytes)
v Domain Name System (response)
    Transaction ID: 0x4478
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
```

Yes, the transaction ID in the response message is correct as it is the same as the transaction ID in the query message. The definition of "correct" is that the DNS use the transaction ID along with the value of the source port to align the responses to the previous query messages.

7 Web server connection

All that we've looked at so far has been needed for our computer to obtain the IP address of the web server that we need. We can now look at what happens when we make contact with the web server. However, before the HTTP protocol elements (that is, those running in your computer and in the remote web server) can talk to one another, they need a reliable connection from one end to the other. This is TCP's job.

7.1 TCP Connection Establishment: Three-Way handshake

A blank version of Figure 8.29 in [1] is copied to figure 2. As you work through the following questions, add to the figure the client and server's IP addresses, port numbers and the actual values[17] of sequence and acknowledgement numbers used in the handshake messages. (Again, you might prefer to print this page so you can write with pen on this diagram.)

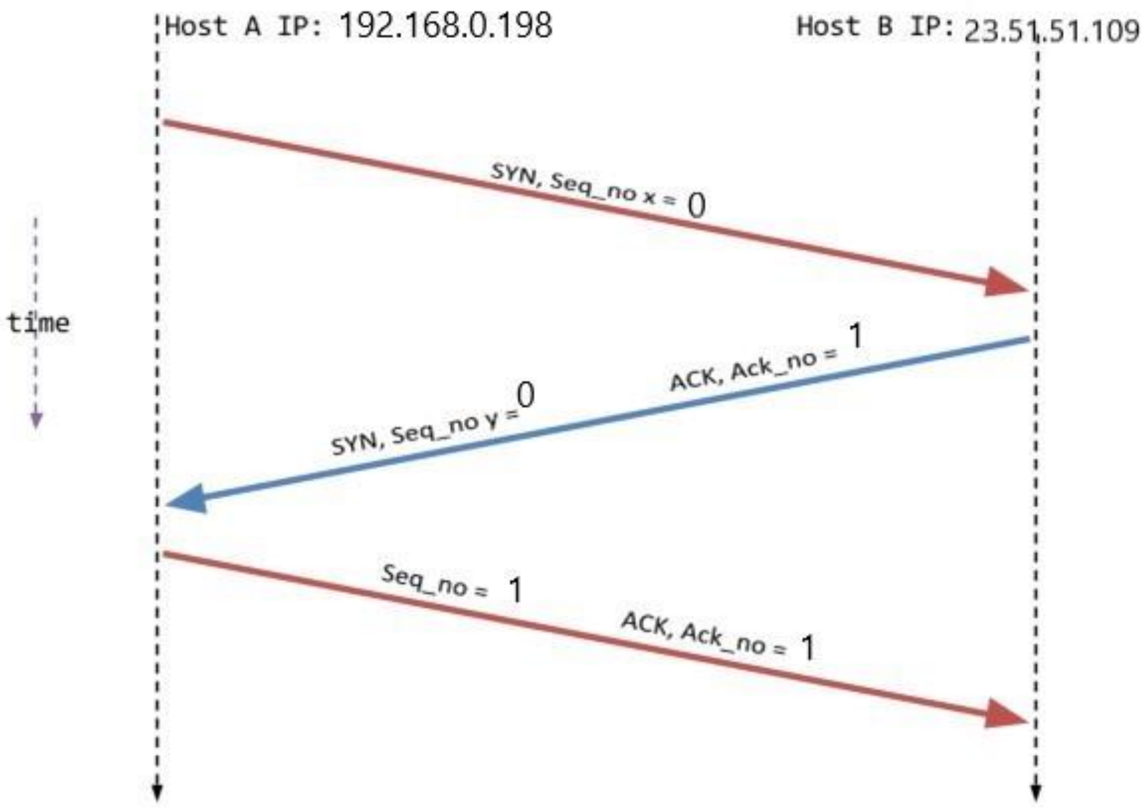


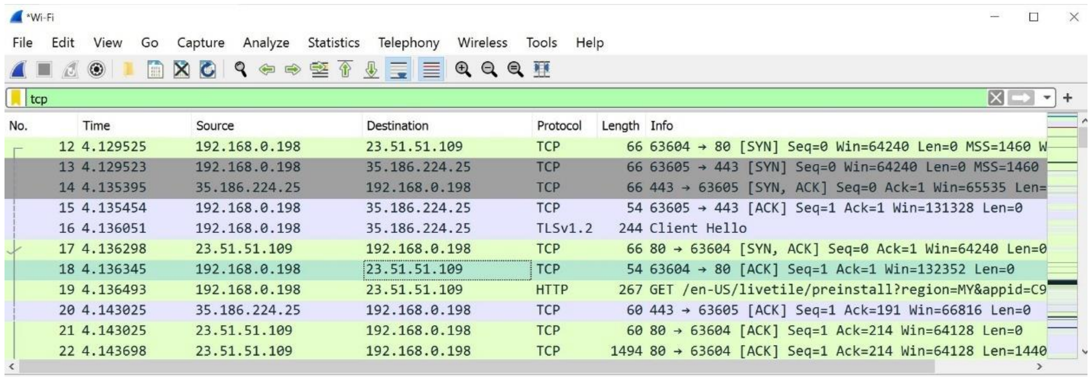
Figure 2. Timing diagram for TCP three-way handshake.

a) Identify the frame that carries the first TCP segment in the three-way handshake that sets up the connection between the HTTP (not HTTPS!) client and server.
(Hint: If it is not obvious which segment is the relevant one by looking at the order and time stamps, you could search according to the address. In the *Answers* field of the DNS *Standard query response* packet, there is the IP address for the URL you requested.)
Using absolute (i.e. not relative) sequence numbers (see Section 3.3), answer the following:

i) Explain both the source and destination Ethernet and IP address used to carry this segment. To what network equipment or computers do these addresses correspond?

Address	Which device has this address (e.g., your computer, local router, web server,…)?
Src Ethernet = a8:7e:ea:c0:9a:81	Network Interface Card MAC address
Dst Ethernet = 34:e8:94:be:cb:44	Local router MAC's Address
Src IP = 192.168.0.198	Personal Computer
Dst IP = 23.51.51.109	Web Server

[17] Note Section 3.3 about relative Sequence Numbers.

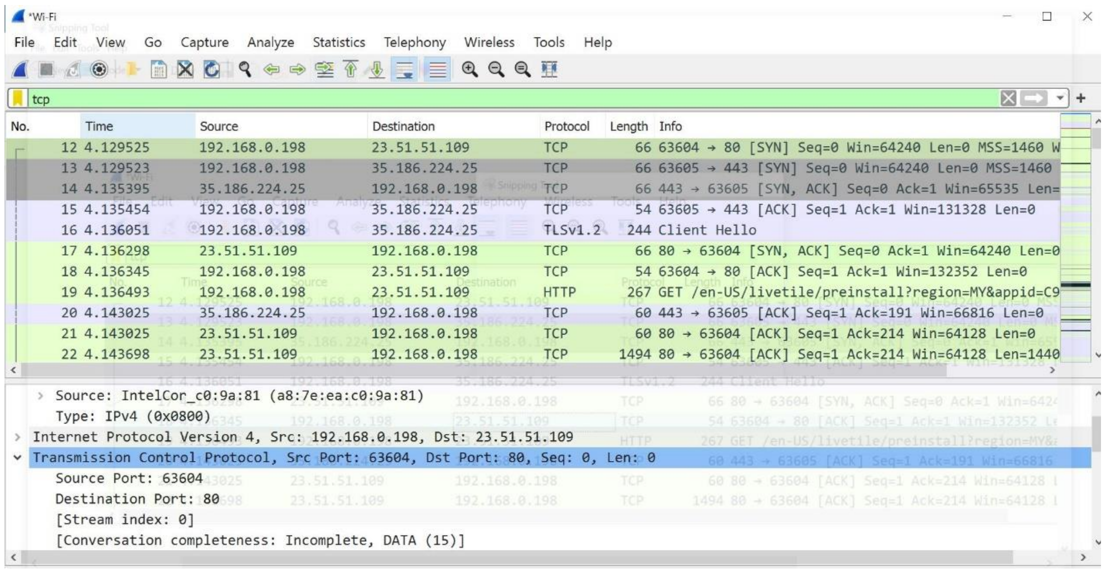


```
tcp
No.  Time         Source          Destination     Protocol  Length Info
  12 4.129525     192.168.0.198   23.51.51.109    TCP       66 63604 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 W
  13 4.129523     192.168.0.198   35.186.224.25   TCP       66 63605 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
  14 4.135395     35.186.224.25   192.168.0.198   TCP       66 443 → 63605 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=
  15 4.135454     192.168.0.198   35.186.224.25   TCP       54 63605 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
  16 4.136051     192.168.0.198   35.186.224.25   TLSv1.2  244 Client Hello
  17 4.136298     23.51.51.109    192.168.0.198   TCP       66 80 → 63604 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
  18 4.136345     192.168.0.198   23.51.51.109    TCP       54 63604 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
  19 4.136493     192.168.0.198   23.51.51.109    HTTP     267 GET /en-US/livetile/preinstall?region=MY&appid=C9
  20 4.143025     35.186.224.25   192.168.0.198   TCP       60 443 → 63605 [ACK] Seq=1 Ack=191 Win=66816 Len=0
  21 4.143025     23.51.51.109    192.168.0.198   TCP       60 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=0
  22 4.143698     23.51.51.109    192.168.0.198   TCP     1494 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=1440
```

The source Ethernet address corresponds to the unique address of the client’s device which is assigned to the NIC card. The source IP address corresponds to the connection of the client’s device in the network.

The destination Ethernet and IP address correspond to the server’s physical and network address.

ii) Looking in the same packet, identify the destination port number inserted by the client in the TCP segment and confirm that the well-known port number is the correct value for HTTP. Will the source port number be well-known or ephemeral? How will you know by inspecting it?



```
tcp
No.  Time         Source          Destination     Protocol  Length Info
  12 4.129525     192.168.0.198   23.51.51.109    TCP       66 63604 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 W
  13 4.129523     192.168.0.198   35.186.224.25   TCP       66 63605 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
  14 4.135395     35.186.224.25   192.168.0.198   TCP       66 443 → 63605 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=
  15 4.135454     192.168.0.198   35.186.224.25   TCP       54 63605 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
  16 4.136051     192.168.0.198   35.186.224.25   TLSv1.2  244 Client Hello
  17 4.136298     23.51.51.109    192.168.0.198   TCP       66 80 → 63604 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
  18 4.136345     192.168.0.198   23.51.51.109    TCP       54 63604 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
  19 4.136493     192.168.0.198   23.51.51.109    HTTP     267 GET /en-US/livetile/preinstall?region=MY&appid=C9
  20 4.143025     35.186.224.25   192.168.0.198   TCP       60 443 → 63605 [ACK] Seq=1 Ack=191 Win=66816 Len=0
  21 4.143025     23.51.51.109    192.168.0.198   TCP       60 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=0
  22 4.143698     23.51.51.109    192.168.0.198   TCP     1494 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=1440

  > Source: IntelCor_c0:9a:81 (a8:7e:ea:c0:9a:81)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 192.168.0.198, Dst: 23.51.51.109
v Transmission Control Protocol, Src Port: 63604, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 63604
    Destination Port: 80
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
```

The destination port will be well known which has a value of 80 (HTTP). The source port will be ephemeral.

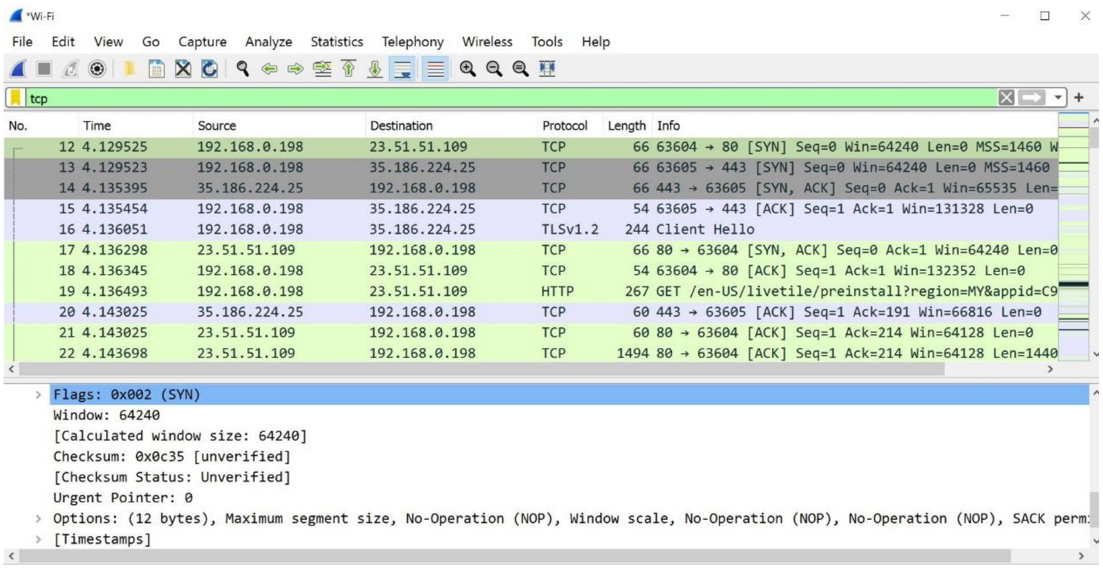
iii) Comparing the destination Ethernet and IP addresses with those you found in the DNS query packet, you will see that one is different, and one is unchanged. Why? (It is REALLY important that you understand this. If you don't, discuss with a demonstrator.)

The destination Ethernet address is unchanged while the IP address is different. The Ethernet address will be the same because client communicates with the subnet router which will then forward the packet to the next hop. The IP address is different because client is communicating with different web server.

iv) What is the initial sequence number (ISN) for the segments from the client to the server[18]?

The ISN for the segments from the client to the server is 2975867284

v) What flag bits are set? Why?


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp

```
No.   Time       Source          Destination     Protocol  Length Info
   12 4.129525   192.168.0.198   23.51.51.109    TCP          66 63604 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 W
   13 4.129523   192.168.0.198   35.186.224.25   TCP          66 63605 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
   14 4.135395   35.186.224.25   192.168.0.198   TCP          66 443 → 63605 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=
   15 4.135454   192.168.0.198   35.186.224.25   TCP          54 63605 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
   16 4.136051   192.168.0.198   35.186.224.25   TLSv1.2     244 Client Hello
   17 4.136298   23.51.51.109    192.168.0.198   TCP          66 80 → 63604 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
   18 4.136345   192.168.0.198   23.51.51.109    TCP          54 63604 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
   19 4.136493   192.168.0.198   23.51.51.109    HTTP        267 GET /en-US/livetile/preinstall?region=MY&appid=C9
   20 4.143025   35.186.224.25   192.168.0.198   TCP          60 443 → 63605 [ACK] Seq=1 Ack=191 Win=66816 Len=0
   21 4.143025   23.51.51.109    192.168.0.198   TCP          60 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=0
   22 4.143698   23.51.51.109    192.168.0.198   TCP        1494 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=1440
```

```
  > Flags: 0x002 (SYN)
    Window: 64240
    [Calculated window size: 64240]
    Checksum: 0x0c35 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK perm:
  > [Timestamps]
```

The flag bits are set to 0x002 (SYN). This is because this describes the connection between hosts (personal computer and the server).

[18] Note that the actual Initial Sequence Number is (basically) a random number, but Wireshark gives a "relative" sequence number (i.e. it subtracts the ISN from all subsequent sequence numbers). Click on the "Sequence number" row to see in the bottom pane the actual byte values that are carried in the sequence number field, or change the preferences in Wireshark (see Section 3.3).

b) Identify the frame that carries the second segment in the three-way handshake and answer the following questions: (Hint: it is a [SYN, ACK] frame).

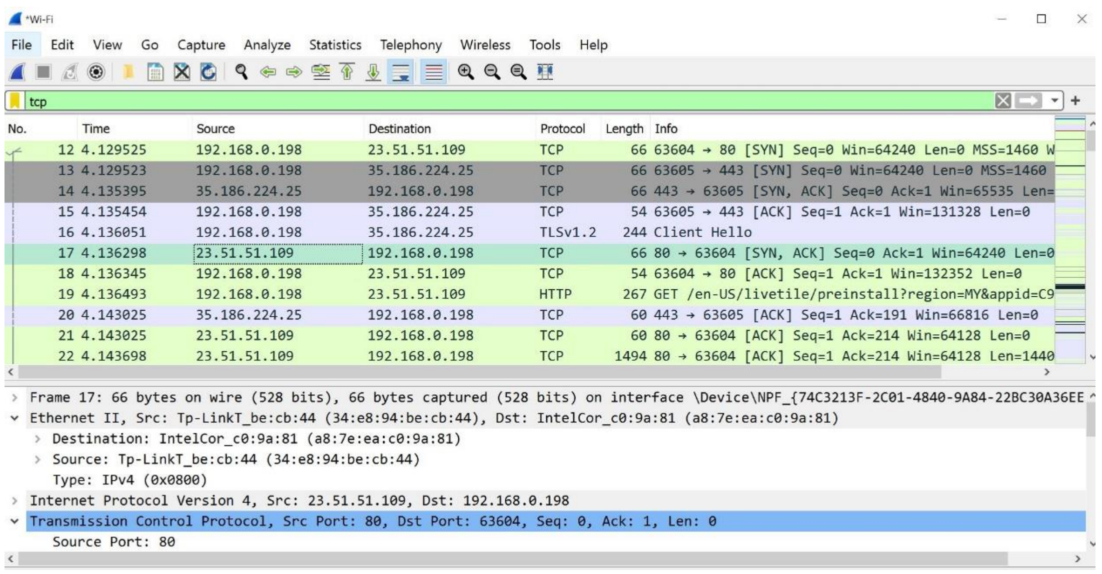
i) <u>Before</u> examining the captured packet, try to predict in advance what values the following fields will have in this frame. Then compare with the capture and make sure you understand why these fields have the values they do.

- Source and destination addresses and type field in the Ethernet frame. How do these Ethernet addresses compare with those of the frame carrying the DNS query (section 5(a) above)? Are they the same or different? Explain.

Address	Which device has this address (e.g., your computer, local router, web server,…)?
Src Ethernet = a8:7e:ea:c0:9a:81	Local Router
Dst Ethernet = 34:e8:94:be:cb:44	Personal Computer

The addresses are different from those of the frame carrying the DNS query. The destination address of the second segment is the address of the client's device while that of the DNS query packet is the address of the subnet router.

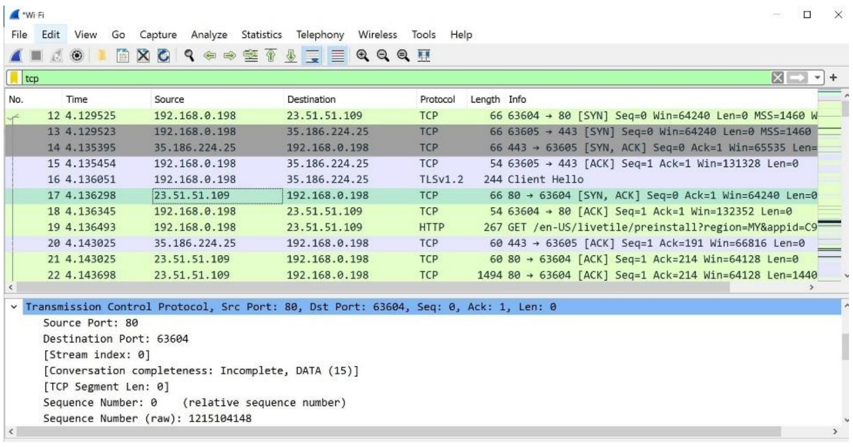
- Source and destination IP addresses.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp

```
No.  Time      Source          Destination     Protocol Length Info
  12 4.129525  192.168.0.198   23.51.51.109    TCP       66 63604 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 W
  13 4.129523  192.168.0.198   35.186.224.25   TCP       66 63605 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
  14 4.135395  35.186.224.25   192.168.0.198   TCP       66 443 → 63605 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=
  15 4.135454  192.168.0.198   35.186.224.25   TCP       54 63605 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
  16 4.136051  192.168.0.198   35.186.224.25   TLSv1.2  244 Client Hello
  17 4.136298  23.51.51.109    192.168.0.198   TCP       66 80 → 63604 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
  18 4.136345  192.168.0.198   23.51.51.109    TCP       54 63604 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
  19 4.136493  192.168.0.198   23.51.51.109    HTTP     267 GET /en-US/livetile/preinstall?region=MY&appid=C9
  20 4.143025  35.186.224.25   192.168.0.198   TCP       60 443 → 63605 [ACK] Seq=1 Ack=191 Win=66816 Len=0
  21 4.143025  23.51.51.109    192.168.0.198   TCP       60 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=0
  22 4.143698  23.51.51.109    192.168.0.198   TCP     1494 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=1440

> Frame 17: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{74C3213F-2C01-4840-9A84-22BC30A36EE
v Ethernet II, Src: Tp-LinkT_be:cb:44 (34:e8:94:be:cb:44), Dst: IntelCor_c0:9a:81 (a8:7e:ea:c0:9a:81)
  > Destination: IntelCor_c0:9a:81 (a8:7e:ea:c0:9a:81)
  > Source: Tp-LinkT_be:cb:44 (34:e8:94:be:cb:44)
    Type: IPv4 (0x0800)
> Internet Protocol Version 4, Src: 23.51.51.109, Dst: 192.168.0.198
v Transmission Control Protocol, Src Port: 80, Dst Port: 63604, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
```

Source IP Address: 23.51.51.109
Destination IP Address: 192.168.0.198

- TCP port numbers.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp

```
No.   Time       Source          Destination     Protocol  Length Info
  12 4.129525   192.168.0.198   23.51.51.109    TCP         66 63604 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 W
  13 4.129523   192.168.0.198   35.186.224.25   TCP         66 63605 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
  14 4.135395   35.186.224.25   192.168.0.198   TCP         66 443 → 63605 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=
  15 4.135454   192.168.0.198   35.186.224.25   TCP         54 63605 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
  16 4.136051   192.168.0.198   35.186.224.25   TLSv1.2    244 Client Hello
  17 4.136298   23.51.51.109    192.168.0.198   TCP         66 80 → 63604 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
  18 4.136345   192.168.0.198   23.51.51.109    TCP         54 63604 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
  19 4.136493   192.168.0.198   23.51.51.109    HTTP       267 GET /en-US/livetile/preinstall?region=MY&appid=C9
  20 4.143025   35.186.224.25   192.168.0.198   TCP         60 443 → 63605 [ACK] Seq=1 Ack=191 Win=66816 Len=0
  21 4.143025   23.51.51.109    192.168.0.198   TCP         60 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=0
  22 4.143698   23.51.51.109    192.168.0.198   TCP       1494 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=1440

v Transmission Control Protocol, Src Port: 80, Dst Port: 63604, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 63604
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1215104148
```

Src: 80	Dst: 63604

- Acknowledgement number in the TCP segment.

1 (Relative ACK Number)

- Which TCP header flag bits are set?

0x012 (SYN, ACK)

ii) What is the initial sequence number for the connection from the server to the client? Is it different to that used for client to server? Why? (Again, refer to Section 3.3.)

The initial sequence number for the connection from the server to the client will be 0 (relative sequence number). No, it will be the same.

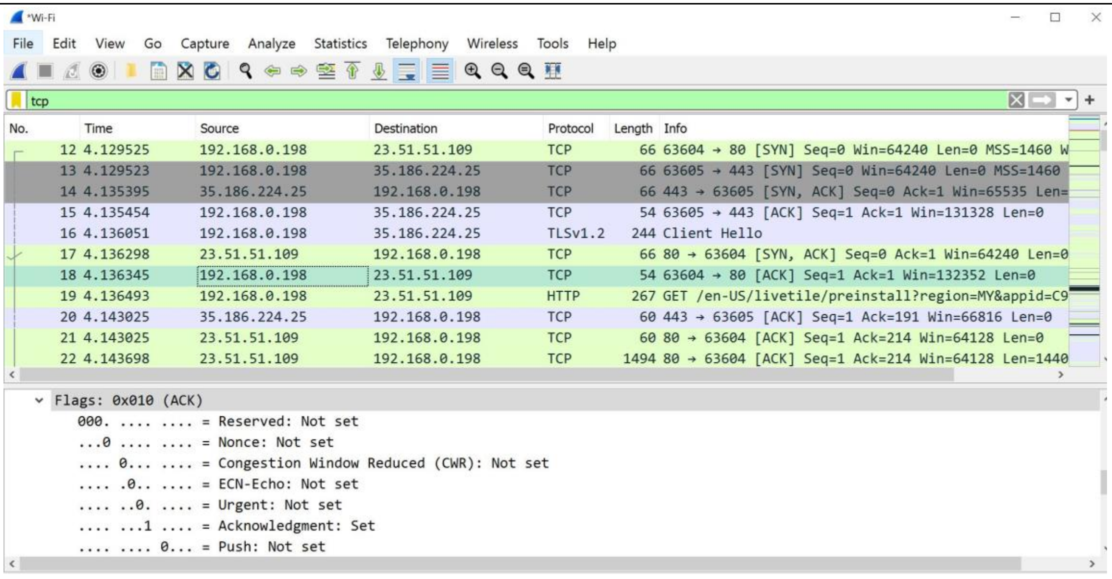
c) Identify the frame that carries the third segment in the three-way handshake, and answer the following questions:

i) As before, write down the values you expect for the following fields, and only then look at the packet capture to confirm that they are what you expect. Make sure you understand why these fields have the values they do.

- Expected TCP Seq and Ack numbers

Seq = 0 (relative sequence number)	The SYN's ISN + 1
Ack = 0 (relative sequence number)	The SYN, ACK ISN + 1

- Flag bits set


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp

```
No.   Time      Source          Destination     Protocol  Length Info
   12 4.129525  192.168.0.198   23.51.51.109    TCP         66 63604 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 W
   13 4.129523  192.168.0.198   35.186.224.25   TCP         66 63605 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
   14 4.135395  35.186.224.25   192.168.0.198   TCP         66 443 → 63605 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=
   15 4.135454  192.168.0.198   35.186.224.25   TCP         54 63605 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
   16 4.136051  192.168.0.198   35.186.224.25   TLSv1.2    244 Client Hello
   17 4.136298  23.51.51.109    192.168.0.198   TCP         66 80 → 63604 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
   18 4.136345  192.168.0.198   23.51.51.109    TCP         54 63604 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
   19 4.136493  192.168.0.198   23.51.51.109    HTTP       267 GET /en-US/livetile/preinstall?region=MY&appid=C9
   20 4.143025  35.186.224.25   192.168.0.198   TCP         60 443 → 63605 [ACK] Seq=1 Ack=191 Win=66816 Len=0
   21 4.143025  23.51.51.109    192.168.0.198   TCP         60 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=0
   22 4.143698  23.51.51.109    192.168.0.198   TCP       1494 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=1440

  Flags: 0x010 (ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
```

Flag: 0x010 (ACK)

7.2 TCP Connection Termination

Having established an on-going connection between client and server over TCP, we can use this to exchange whatever data we want (in this case, it will be used by HTTP to request and deliver a web page). Before we look at the exchange of data over the established TCP connection, let's briefly jump ahead and consider what happens at the end when we've finished with the TCP connection and want to close it down[19].

What TCP segment flag would you expect to see set if one of the ends of the link wanted to shut down the connection? Can you determine whether (any of) your TCP connections are terminated "gracefully" or not?

FIN and ACK segment flag.

TCP connections are terminated gracefully when:
1. [FIN, ACK] frame is sent to the server by client
2. [FIN, ACK] frame is replied from the server to the client
3. [ACK] frame is sent to the server by the client.

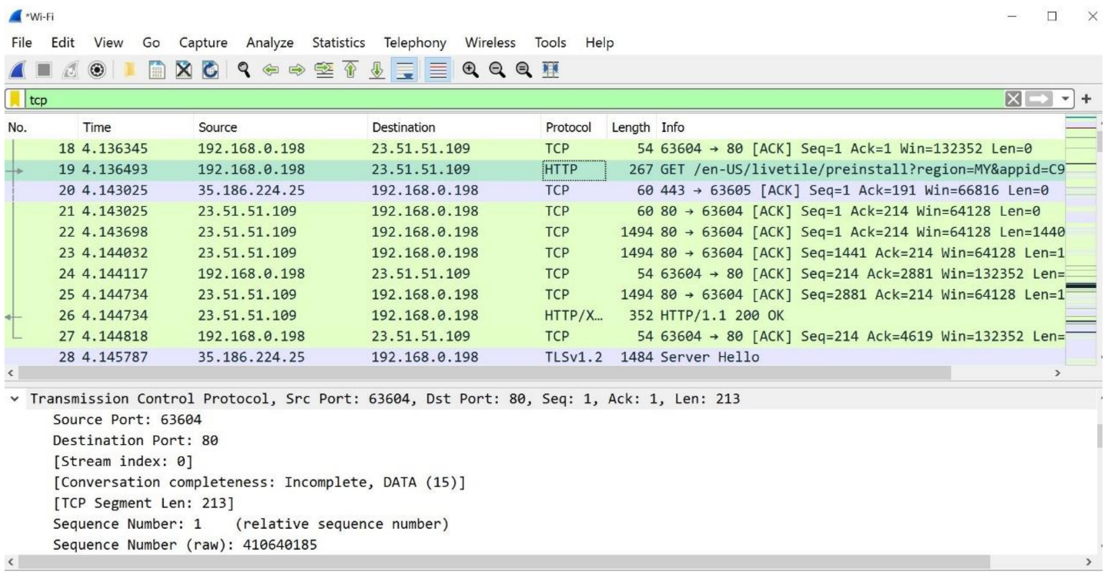
[19] When we've finished with a connection, we could just leave it hanging and simply stop sending data. Normally, there would be a time-out process and the server would eventually decide to close down the connection because it hasn't heard from the client for a while. This is not a terrible thing to do, but it does use some small amount of server resources unnecessarily until the time-out occurs. The alternative is to formally advise the server that we've finished with the connection and the two ends can agree on and confirm its termination. This latter approach is described as "graceful" closure, because it's a neat and tidy way to do it.

8 HTTP

8.1 HTTP GET

a) Identify the frame that carries the HTTP GET message. Use the absolute sequence numbers and answer the following questions:

i) What are the sequence and acknowledgement values in the TCP header? Can you relate these to the values from the three-way handshake?

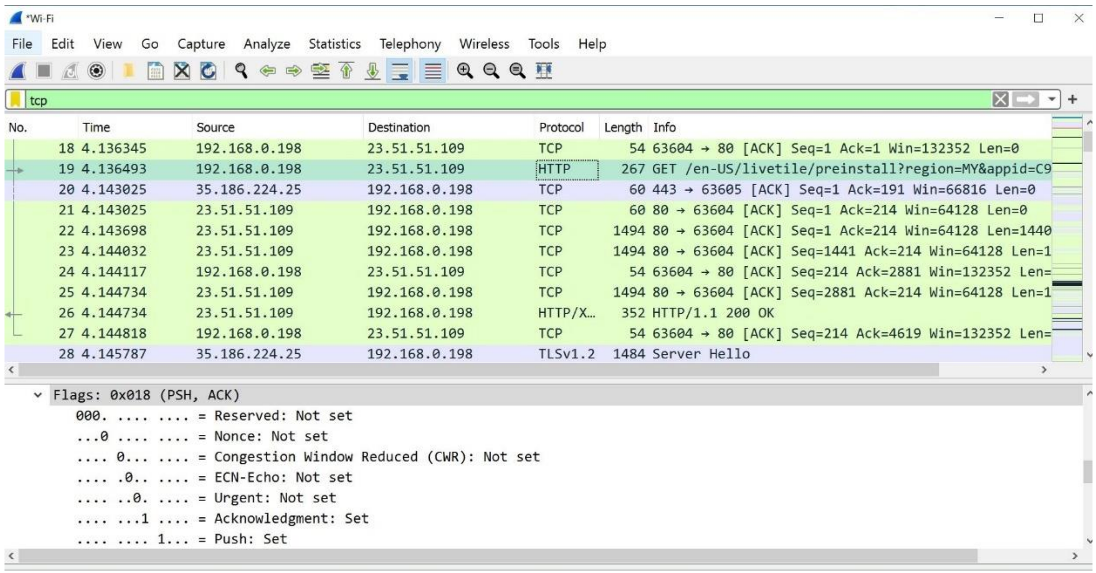


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp

```
No.  Time      Source         Destination    Protocol Length Info
  18 4.136345  192.168.0.198  23.51.51.109   TCP        54 63604 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
  19 4.136493  192.168.0.198  23.51.51.109   HTTP      267 GET /en-US/livetile/preinstall?region=MY&appid=C9
  20 4.143025  35.186.224.25  192.168.0.198  TCP        60 443 → 63605 [ACK] Seq=1 Ack=191 Win=66816 Len=0
  21 4.143025  23.51.51.109   192.168.0.198  TCP        60 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=0
  22 4.143698  23.51.51.109   192.168.0.198  TCP      1494 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=1440
  23 4.144032  23.51.51.109   192.168.0.198  TCP      1494 80 → 63604 [ACK] Seq=1441 Ack=214 Win=64128 Len=1
  24 4.144117  192.168.0.198  23.51.51.109   TCP        54 63604 → 80 [ACK] Seq=214 Ack=2881 Win=132352 Len=
  25 4.144734  23.51.51.109   192.168.0.198  TCP      1494 80 → 63604 [ACK] Seq=2881 Ack=214 Win=64128 Len=1
  26 4.144734  23.51.51.109   192.168.0.198  HTTP/X…   352 HTTP/1.1 200 OK
  27 4.144818  192.168.0.198  23.51.51.109   TCP        54 63604 → 80 [ACK] Seq=214 Ack=4619 Win=132352 Len=
  28 4.145787  35.186.224.25  192.168.0.198  TLSv1.2  1484 Server Hello

Transmission Control Protocol, Src Port: 63604, Dst Port: 80, Seq: 1, Ack: 1, Len: 213
    Source Port: 63604
    Destination Port: 80
    [Stream index: 0]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 213]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 410640185
```

The sequence value will be 1 and the acknowledgement value in the TCP header will be 1. These values will correspond to the values from the TCP three-way handshake which was initially 0. Therefore, we can conclude that they match.

ii) What flag bits in the TCP header are set, and why?


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp

```
No.    Time      Source          Destination     Protocol Length Info
    18 4.136345  192.168.0.198   23.51.51.109    TCP        54 63604 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
    19 4.136493  192.168.0.198   23.51.51.109    HTTP      267 GET /en-US/livetile/preinstall?region=MY&appid=C9
    20 4.143025  35.186.224.25   192.168.0.198   TCP        60 443 → 63605 [ACK] Seq=1 Ack=191 Win=66816 Len=0
    21 4.143025  23.51.51.109    192.168.0.198   TCP        60 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=0
    22 4.143698  23.51.51.109    192.168.0.198   TCP      1494 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=1440
    23 4.144032  23.51.51.109    192.168.0.198   TCP      1494 80 → 63604 [ACK] Seq=1441 Ack=214 Win=64128 Len=1
    24 4.144117  192.168.0.198   23.51.51.109    TCP        54 63604 → 80 [ACK] Seq=214 Ack=2881 Win=132352 Len=
    25 4.144734  23.51.51.109    192.168.0.198   TCP      1494 80 → 63604 [ACK] Seq=2881 Ack=214 Win=64128 Len=1
    26 4.144734  23.51.51.109    192.168.0.198   HTTP/X…   352 HTTP/1.1 200 OK
    27 4.144818  192.168.0.198   23.51.51.109    TCP        54 63604 → 80 [ACK] Seq=214 Ack=4619 Win=132352 Len=
    28 4.145787  35.186.224.25   192.168.0.198   TLSv1.2  1484 Server Hello

  ⌄ Flags: 0x018 (PSH, ACK)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Nonce: Not set
        .... 0... .... = Congestion Window Reduced (CWR): Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgment: Set
        .... .... 1... = Push: Set
```

The flag bits in the TCP header are set to 0x018 (PSH, ACK).
The PSH Flag will show that the client is transmitting data while the ACK is the acknowledgement for the previous SYN flag.

b) Now consider the contents of the GET message and answer the following questions:

i) Scroll down to the third pane in the Wireshark window (you might have to drag it up) and compare the decoded text with the contents of the HTTP message in the second pane. Do they match?

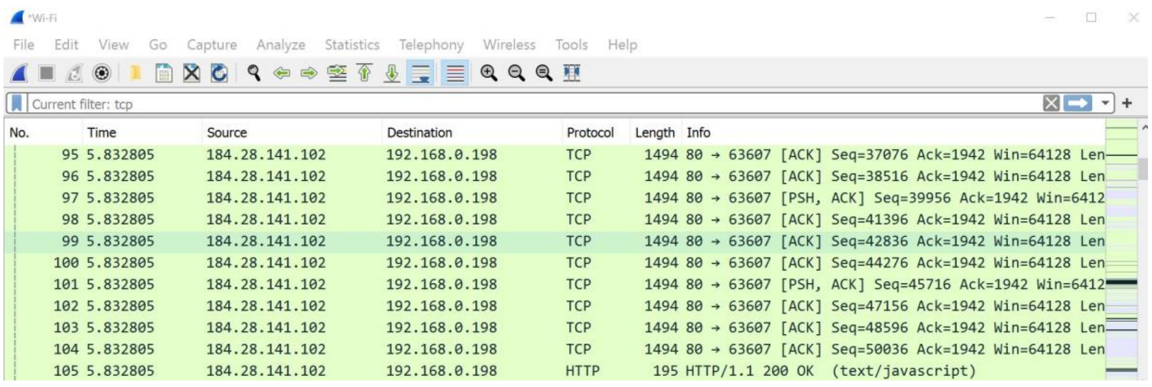
Yes, they match.

ii) What is the next sequence number that is expected in the next segment from the server? Why? (using Relative Sequence Numbers might help you)

The sequence number expected in the next segment from the client will be the previous acknowledgement number + 1. TCP is a byte stream protocol. It needs to split a big segment into multiple small segments containing pieces of message of the application and reassemble it at the host.

8.2 HTTP Response

a) Determine whether the server responds with an HTTP response message or simply with a TCP ACK segment.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Current filter: tcp

```
No.  Time      Source          Destination    Protocol  Length Info
  95 5.832805  184.28.141.102  192.168.0.198  TCP       1494 80 → 63607 [ACK] Seq=37076 Ack=1942 Win=64128 Len
  96 5.832805  184.28.141.102  192.168.0.198  TCP       1494 80 → 63607 [ACK] Seq=38516 Ack=1942 Win=64128 Len
  97 5.832805  184.28.141.102  192.168.0.198  TCP       1494 80 → 63607 [PSH, ACK] Seq=39956 Ack=1942 Win=6412
  98 5.832805  184.28.141.102  192.168.0.198  TCP       1494 80 → 63607 [ACK] Seq=41396 Ack=1942 Win=64128 Len
  99 5.832805  184.28.141.102  192.168.0.198  TCP       1494 80 → 63607 [ACK] Seq=42836 Ack=1942 Win=64128 Len
 100 5.832805  184.28.141.102  192.168.0.198  TCP       1494 80 → 63607 [ACK] Seq=44276 Ack=1942 Win=64128 Len
 101 5.832805  184.28.141.102  192.168.0.198  TCP       1494 80 → 63607 [PSH, ACK] Seq=45716 Ack=1942 Win=6412
 102 5.832805  184.28.141.102  192.168.0.198  TCP       1494 80 → 63607 [ACK] Seq=47156 Ack=1942 Win=64128 Len
 103 5.832805  184.28.141.102  192.168.0.198  TCP       1494 80 → 63607 [ACK] Seq=48596 Ack=1942 Win=64128 Len
 104 5.832805  184.28.141.102  192.168.0.198  TCP       1494 80 → 63607 [ACK] Seq=50036 Ack=1942 Win=64128 Len
 105 5.832805  184.28.141.102  192.168.0.198  HTTP       195 HTTP/1.1 200 OK  (text/javascript)
```

The server responds with multiple TCP ACK segment and followed by a HTTP response message. TCP is a byte stream protocol; it needs to split a big segment into multiple small segments containing pieces of message of the application and reassemble it at the host.

b) Consider the TCP segment that contains the HTTP response message and answer the following questions:

i) Verify that the sequence number in the segment from the server is as expected.

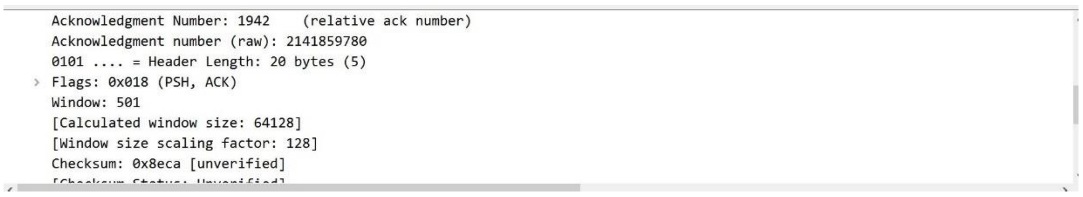
Acknowledgement number from the GET request is 1357847701
2 TCP segment from the web server 2x5000 = 10000
Sequence number of the HTTP Response Message from the web server = 1357857701

ii) What acknowledgement number is expected in the next segment from the client?

The acknowledgement number of the next segment is the sequence number of the HTTP response message + the length of the TCP segment length.
1357857701 + 1093 = 1357858794

The acknowledgement number expected in the next segment from the client will be the previous sequence number + 1.

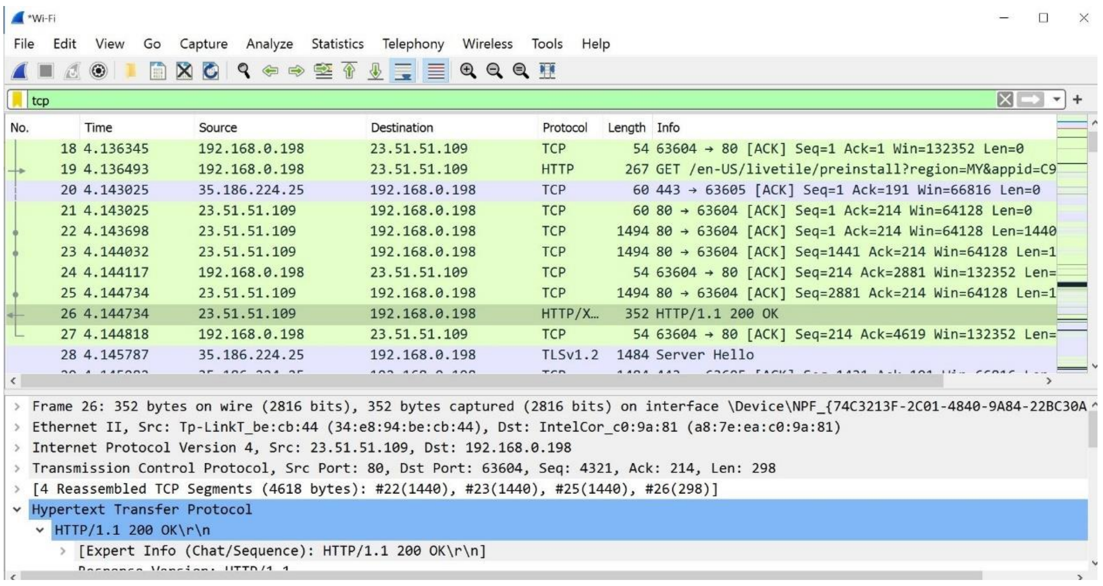
iii) Examine whether any of the flags are set and explain why they are set.



```
Acknowledgment Number: 1942    (relative ack number)
Acknowledgment number (raw): 2141859780
0101 .... = Header Length: 20 bytes (5)
> Flags: 0x018 (PSH, ACK)
Window: 501
[Calculated window size: 64128]
[Window size scaling factor: 128]
Checksum: 0x8eca [unverified]
```

The PSH and ACK Flag has been set. The PSH Flag will show that the client is transmitting data while the ACK is the acknowledgement for the previous SYN flag.

c) Now consider the HTTP response message inside the TCP segment: What is the result code in the response message?

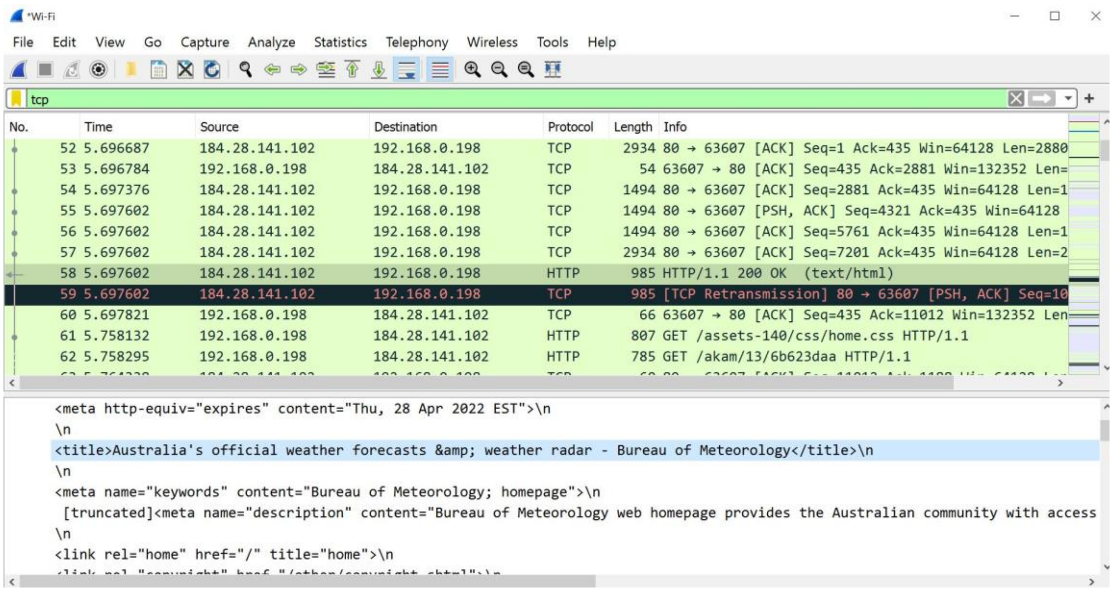

*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp

```
No.   Time      Source          Destination     Protocol  Length Info
   18 4.136345  192.168.0.198   23.51.51.109    TCP         54 63604 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
   19 4.136493  192.168.0.198   23.51.51.109    HTTP       267 GET /en-US/livetile/preinstall?region=MY&appid=C9
   20 4.143025  35.186.224.25   192.168.0.198   TCP         60 443 → 63605 [ACK] Seq=1 Ack=191 Win=66816 Len=0
   21 4.143025  23.51.51.109    192.168.0.198   TCP         60 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=0
   22 4.143698  23.51.51.109    192.168.0.198   TCP       1494 80 → 63604 [ACK] Seq=1 Ack=214 Win=64128 Len=1440
   23 4.144032  23.51.51.109    192.168.0.198   TCP       1494 80 → 63604 [ACK] Seq=1441 Ack=214 Win=64128 Len=1
   24 4.144117  192.168.0.198   23.51.51.109    TCP         54 63604 → 80 [ACK] Seq=214 Ack=2881 Win=132352 Len=
   25 4.144734  23.51.51.109    192.168.0.198   TCP       1494 80 → 63604 [ACK] Seq=2881 Ack=214 Win=64128 Len=1
   26 4.144734  23.51.51.109    192.168.0.198   HTTP/X…    352 HTTP/1.1 200 OK
   27 4.144818  192.168.0.198   23.51.51.109    TCP         54 63604 → 80 [ACK] Seq=214 Ack=4619 Win=132352 Len=
   28 4.145787  35.186.224.25   192.168.0.198   TLSv1.2   1484 Server Hello

> Frame 26: 352 bytes on wire (2816 bits), 352 bytes captured (2816 bits) on interface \Device\NPF_{74C3213F-2C01-4840-9A84-22BC30A
> Ethernet II, Src: Tp-LinkT_be:cb:44 (34:e8:94:be:cb:44), Dst: IntelCor_c0:9a:81 (a8:7e:ea:c0:9a:81)
> Internet Protocol Version 4, Src: 23.51.51.109, Dst: 192.168.0.198
> Transmission Control Protocol, Src Port: 80, Dst Port: 63604, Seq: 4321, Ack: 214, Len: 298
> [4 Reassembled TCP Segments (4618 bytes): #22(1440), #23(1440), #25(1440), #26(298)]
v Hypertext Transfer Protocol
  v HTTP/1.1 200 OK\r\n
    > [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
```

The result code in the response message is a response code 200 (OK). We will get HTML Code.

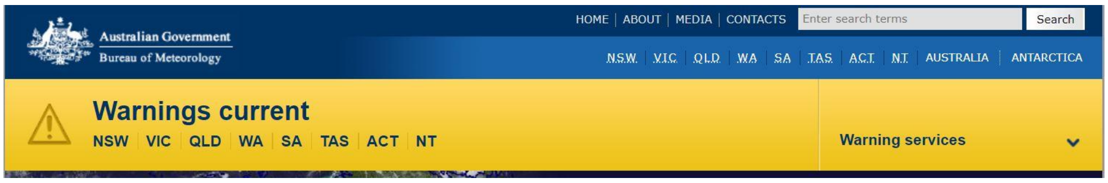
d) Highlight the "data" section of the HTTP response message. Scroll down to the third pane in the Wireshark window and compare and comment on the decoded text with the contents of the web page that was displayed on your screen[20].

[20] Don't worry if you can't recognise anything here. Because you've chosen an arbitrary web page, you can't necessarily expect to be able to recognise what comes back in the first packet. It might not be recognisable text, and it might even be zipped. If the page is not too complicated, though, you might be able to see text in the packet that also appears on-screen within the browser.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp

```
No.    Time       Source           Destination      Protocol  Length Info
  52 5.696687   184.28.141.102   192.168.0.198    TCP       2934 80 → 63607 [ACK] Seq=1 Ack=435 Win=64128 Len=2880
  53 5.696784   192.168.0.198    184.28.141.102   TCP         54 63607 → 80 [ACK] Seq=435 Ack=2881 Win=132352 Len=
  54 5.697376   184.28.141.102   192.168.0.198    TCP       1494 80 → 63607 [ACK] Seq=2881 Ack=435 Win=64128 Len=1
  55 5.697602   184.28.141.102   192.168.0.198    TCP       1494 80 → 63607 [PSH, ACK] Seq=4321 Ack=435 Win=64128
  56 5.697602   184.28.141.102   192.168.0.198    TCP       1494 80 → 63607 [ACK] Seq=5761 Ack=435 Win=64128 Len=1
  57 5.697602   184.28.141.102   192.168.0.198    TCP       2934 80 → 63607 [ACK] Seq=7201 Ack=435 Win=64128 Len=2
  58 5.697602   184.28.141.102   192.168.0.198    HTTP       985 HTTP/1.1 200 OK  (text/html)
  59 5.697602   184.28.141.102   192.168.0.198    TCP        985 [TCP Retransmission] 80 → 63607 [PSH, ACK] Seq=10
  60 5.697821   192.168.0.198    184.28.141.102   TCP         66 63607 → 80 [ACK] Seq=435 Ack=11012 Win=132352 Len
  61 5.758132   192.168.0.198    184.28.141.102   HTTP       807 GET /assets-140/css/home.css HTTP/1.1
  62 5.758295   192.168.0.198    184.28.141.102   HTTP       785 GET /akam/13/6b623daa HTTP/1.1
```

```
<meta http-equiv="expires" content="Thu, 28 Apr 2022 EST">\n
\n
<title>Australia's official weather forecasts &amp; weather radar - Bureau of Meteorology</title>\n
\n
<meta name="keywords" content="Bureau of Meteorology; homepage">\n
 [truncated]<meta name="description" content="Bureau of Meteorology web homepage provides the Australian community with access
\n
<link rel="home" href="/" title="home">\n
```


HOME | ABOUT | MEDIA | CONTACTS
Enter search terms
Search
Australian Government
Bureau of Meteorology
N.S.W. V.I.C. Q.L.D. W.A. S.A. T.A.S. A.C.T. N.T. AUSTRALIA ANTARCTICA
Warnings current
NSW VIC QLD WA SA TAS ACT NT
Warning services

The decoded text in Wireshark will show HTML code which consists of header commands and new line commands.

e) How many "GET" requests to different URLs resulted from your request for a web page? Can you relate this number (not the exact number, but in terms of whether it is small or large) to the on-screen appearance of the web page?[21]

There is a total of 5 "GET" requests to different URL's resulted from my request for a web page.

The number of requests made is based on the data that needs to be displayed in a web page. For example, a webpage which consists of many images and URL will need more "GET" requests.

[21] Many web pages are composites, in which each element may have its own URL and require a separate GET. The "simple3141" html document would be retrieved with a single GET.

9 Conclusion

In this laboratory class, you have investigated the encapsulation of information requests and responses in the various layers used in the TCP/IP protocol suite. You have hopefully gained a greater understanding of the multiple layers of addressing in the Internet, as well as the series of processes that result from simply clicking on a URL.

Before finishing, could each student please click on the "Feedback" icon for this laboratory on Moodle and provide some brief feedback on this laboratory exercise (all inputs are anonymous).

Finally, please submit this completed report via Moodle by the stated deadline. In so doing, please be aware of the following:

- Even if you have had a mark assigned to you during the lab session, this mark will not be registered unless you have also submitted the report.
- Your mark may also not be accepted or may be modified if your report is incomplete or is identical to that of another student.
- By uploading the report, you are agreeing with the following student statement on collusion and plagiarism, and must be aware of the possible consequences.

References

[1] A. Leon-Garcia and I Widjaja. *Communications Networks: Fundamental Concepts and Key Architectures,* 2nd ed., New York: McGraw-Hill, 2004

– END –