U10814
—
LAB 5: Protocol Analysis Using Wireshark

Author	**Danny Werb**
Contributors	**Alexios Louridas & Seb Blair**
Last Update	**2023-03-16**
Number of pages	**14**

1 Introduction

This lab will introduce you to packet capturing (packet sniffing) and network traffic analysis with *Wireshark*.

1.1 What is Wireshark?

Wireshark is a network protocol analyser. It lets you capture and interactively browse the traffic running on a computer network. It has a rich and powerful feature set and is the world's most popular tool of its kind. It runs on most computing platforms including Windows, macOS, Linux, and UNIX. Network professionals, security experts, developers, and educators around the world use it regularly. It is freely available as open source, and is released under the GNU General Public License version 2.

It is developed and maintained by a global team of protocol experts, and it is an example of a disruptive technology.[1]

1.2 What is Packet Capture (Packet Sniffing)?

A *packet sniffer* is an application which can capture and analyse network traffic which is passing through a system's Network Interface Card (NIC). The sniffer sets the card to *promiscuous mode* which means all traffic is read, whether it is addressed to that machine or not. The figure below shows an attacker sniffing packets from the network, and the *Wireshark* packet sniffer/analyser.

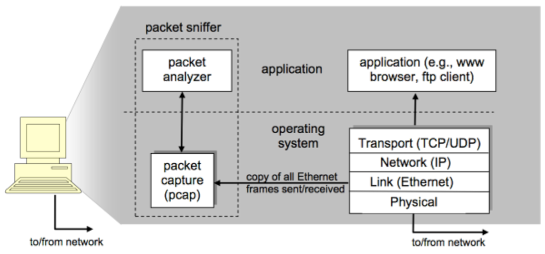


Figure 1: Packet Sniffing[2]

1.3 What is Packet Analysis?

Wireshark is an open source cross-platform packet capture and analysis tool, with versions for Windows and Linux. The GUI window gives a detailed breakdown of the network protocol stack for each packet, colourising packet details based on protocol, as well as having functionality to filter and search the traffic, and pick out TCP streams. Wireshark can also save packet data to files for offline analysis and export/import packet captures to/from other tools. Statistics can also be generated for packet capture files.

Wireshark can be used for *network troubleshooting*, to *investigate security issues* and to *analyse and understand network protocols*. The packet sniffer can exploit information passed in plaintext, i.e.,

[1]https://www.wireshark.org/faq.html#q1.1
[2]http://mathcs.pugetsound.edu/~tmullen/images/nw/packet_sniffer.png

not encrypted. Examples of *protocols* which pass information in plaintext are *Telnet*, *FTP*, *SNMP*, *POP*, and *HTTP*.

Wireshark is a GUI based network capture tool. There is a command line based version of the packet capture utility, called *tshark*. Tshark provides many of the same features as its big sibling, but is console-based. It can be a good alternative if only command line access is available, and also uses less resources as it has no GUI to generate.

2 Using Wireshark to Capture and Analyse Traffic

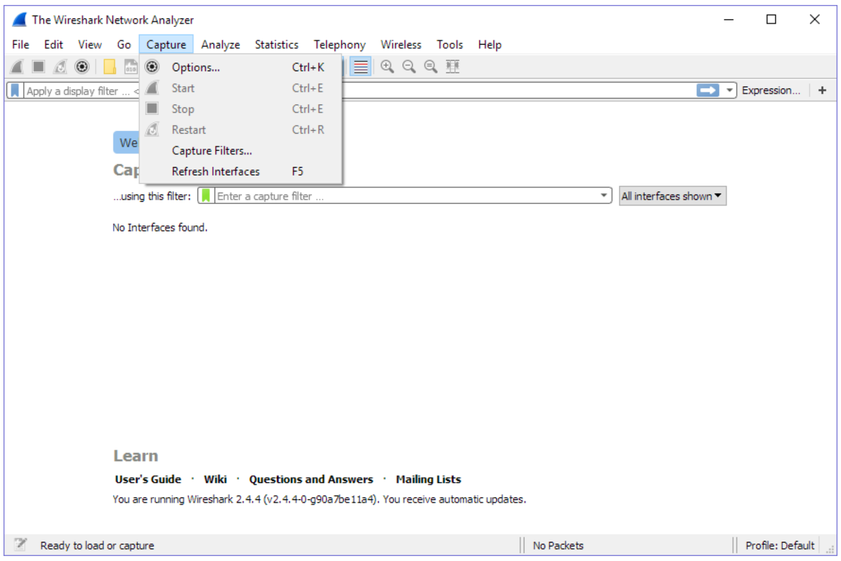
In this exercise, the fundamentals of the *Wireshark Packet Sniffer and Protocol Analyser* tool will be introduced. Then Wireshark will be used to perform basic protocol analysis on TCP/IP network traffic.

The Wireshark User Guide can be found at:

`http://www.wireshark.org/docs/wsug_html_chunged/`

2.1 Using Wireshark to Capture Traffic

Start the Wireshark application. When Wireshark is first run, a default, or blank window is shown. To list the available network interfaces, either select the Capture ⟩ Options menu option (see Figure 2 and Figure 3), or select from an interface listed directly on the main Wireshark screen.


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Options... Ctrl+K
Start Ctrl+E
Stop Ctrl+E
Restart Ctrl+R
Capture Filters...
Refresh Interfaces F5
Apply a display filter ...
Expression...
...using this filter:
Enter a capture filter ...
All interfaces shown
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
Ready to load or capture
No Packets
Profile: Default

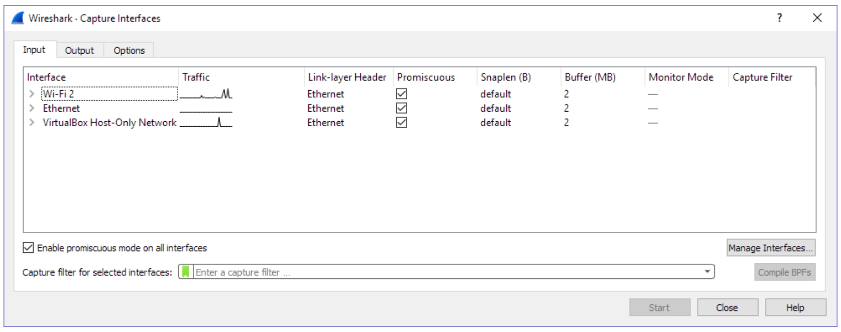
No Interfaces found.

You are running Wireshark 2.4.4 (v2.4.4-0-g90a7be11a4). You receive automatic updates.

Figure 2: Select Capture ⇒ Options

Wireshark should display a popup window such as the one shown in Figure 2. To capture network traffic click the Start button for the network interface you want to capture traffic on.

Windows can have a long list of virtual interfaces, before the Ethernet Network Interface Card (NIC).



Interface	Traffic	Link-layer Header	Promiscuous	Snaplen (B)	Buffer (MB)	Monitor Mode	Capture Filter
Wi-Fi 2		Ethernet	✓	default	2	—	
Ethernet		Ethernet	✓	default	2	—	
VirtualBox Host-Only Network		Ethernet	✓	default	2	—	

☑ Enable promiscuous mode on all interfaces

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

Figure 3: Showing the options input tab with interfaces to select from

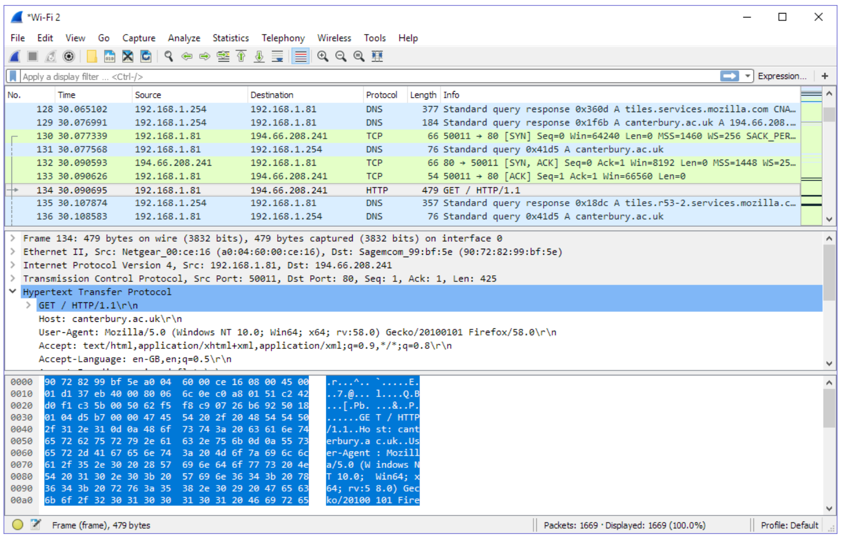
QUESTIONS:

☞ Which Interface is connected to a local network (Ethernet)?

☞ How many packets have passed through the interface?

Note: The total incoming packets, for each interface, are displayed in the column to the left of the Start button.

Generate some network traffic with a Web Browser, such as Internet Explorer, Firefox or Chrome. Your Wireshark window should show the packets, and now look something like in Figure 4:


*Wi-Fi 2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.   Time       Source          Destination     Protocol Length Info
  128 30.065102  192.168.1.254   192.168.1.81    DNS       377 Standard query response 0x360d A tiles.services.mozilla.com CNA…
  129 30.076991  192.168.1.254   192.168.1.81    DNS       184 Standard query response 0x1f6b A canterbury.ac.uk A 194.66.208.…
  130 30.077339  192.168.1.81    194.66.208.241  TCP        66 50011 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PER…
  131 30.077568  192.168.1.81    192.168.1.254   DNS        76 Standard query 0x41d5 A canterbury.ac.uk
  132 30.090593  194.66.208.241  192.168.1.81    TCP        66 80 → 50011 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1448 WS=25…
  133 30.090626  192.168.1.81    194.66.208.241  TCP        54 50011 → 80 [ACK] Seq=1 Ack=1 Win=66560 Len=0
  134 30.090695  192.168.1.81    194.66.208.241  HTTP      479 GET / HTTP/1.1
  135 30.107874  192.168.1.254   192.168.1.81    DNS       357 Standard query response 0x18dc A tiles.r53-2.services.mozilla.c…
  136 30.108583  192.168.1.81    192.168.1.254   DNS        76 Standard query 0x41d5 A canterbury.ac.uk
```

```
> Frame 134: 479 bytes on wire (3832 bits), 479 bytes captured (3832 bits) on interface 0
> Ethernet II, Src: Netgear_00:ce:16 (a0:04:60:00:ce:16), Dst: Sagemcom_99:bf:5e (90:72:82:99:bf:5e)
> Internet Protocol Version 4, Src: 192.168.1.81, Dst: 194.66.208.241
> Transmission Control Protocol, Src Port: 50011, Dst Port: 80, Seq: 1, Ack: 1, Len: 425
v Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: canterbury.ac.uk\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-GB,en;q=0.5\r\n
```

```
0000  90 72 82 99 bf 5e a0 04  60 00 ce 16 08 00 45 00   .r...^.. `.....E.
0010  01 d1 37 eb 40 00 80 06  6c 0e c0 a8 01 51 c2 42   ..7.@... l....Q.B
0020  d0 f1 c3 5b 00 50 62 f5  f8 c9 07 26 b6 92 50 18   ...[.Pb. ...&..P.
0030  01 04 d5 b7 00 00 47 45  54 20 2f 20 48 54 54 50   ......GE T / HTTP
0040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 63 61 6e 74   /1.1..Ho st: cant
0050  65 72 62 75 72 79 2e 61  63 2e 75 6b 0d 0a 55 73   erbury.a c.uk..Us
0060  65 72 2d 41 67 65 6e 74  3a 20 4d 6f 7a 69 6c 6c   er-Agent : Mozill
0070  61 2f 35 2e 30 20 28 57  69 6e 64 6f 77 73 20 4e   a/5.0 (W indows N
0080  54 20 31 30 2e 30 3b 20  57 69 6e 36 34 3b 20 78   T 10.0;  Win64; x
0090  36 34 3b 20 72 76 3a 35  38 2e 30 29 20 47 65 63   64; rv:5 8.0) Gec
00a0  6b 6f 2f 32 30 31 30 30  31 30 31 20 46 69 72 65   ko/20100 101 Fire
```

Frame (frame), 479 bytes | Packets: 1669 · Displayed: 1669 (100.0%) | Profile: Default

Figure 4: Showing the options input tab with interfaces to select from

To stop the capture select the Capture › Stop menu option, Ctrl+E, or the Stop toolbar button. What you have created is a Packet Capture or **'pcap'**, which you can now view and analyse using the Wireshark interface, or save to disk to analyse later.

The capture is split into 3 parts:

1. **Packet List Panel** – this is a list of packets in the current capture. It colours the packets based on the protocol type. When a packet is selected, the details are shown in the two panels below.
2. **Packet Details Panel** – this shows the details of the selected packet. It shows the different protocols making up the layers of data for this packet. Layers include Frame, Ethernet, IP, TCP/UDP/ICMP, and application protocols such as HTTP.
3. **Packet Bytes Panel** – shows the packet bytes in a Hex and ASCII encoding.

Search through your capture, and find an **HTTP** packet containing **GET** command. You are able to expand each layer. Click on the packet in the **Packet List Panel**. Then expand the HTTP layer in the **Packet Details Panel**, from the packet.

QUESTIONS:

☞ From the **Packet Details Panel**, within the GET command, what is the value of the **Host**?

☞ Can you see the **Hex** and **ASCII** representations of the packet in the **Packet Bytes Panel**?

☞ From the **Packet Bytes Panel**, what are the first 4 bytes of the Hex value of the **Host** parameter?

To select more detailed options when starting a capture, select the Capture » Options menu option, or Ctrl + K, or the Capture Options button on the toolbar (the cog).

Some of the more interesting options are hidden away in the other tabs:

- Input » Interface – It is important to select the correct Network Interface to capture traffic, and so it will show you on a mini-linegraph if there is any traffic on that interface.
- Output » Capture to a permanent file – Useful to save a file of the packet capture in real time, in case of a system crash. You can also specify that it should cut the capture into chunks of defined file sizes or time.
- Options » Display Options » Update list of packets in real time – A display option, which should be checked if you want to view the capture as it happens (typically switched off to capture straight to a file, for later analysis).
- Options » Name Resolution » MAC name resolution – Resolves the first 3 bytes of the MAC address, the Organisation Unique Identifier (OUI), which represents the manufacturer of the card.
- Options » Name Resolution » Network name resolution – Does a DNS lookup for the IP Addresses captured, to display the network name. Set to off by default, so covert scans do not generate this DNS traffic, and tip off who's packets you are sniffing.

Make sure the MAC name resolution is selected. Start the capture, and generate some Web traffic again, then stop the capture.

Search through your capture, and find an HTTP packet coming back from the server (TCP Source Port == 80). Expand the Ethernet layer in the **Packet Details Panel**.

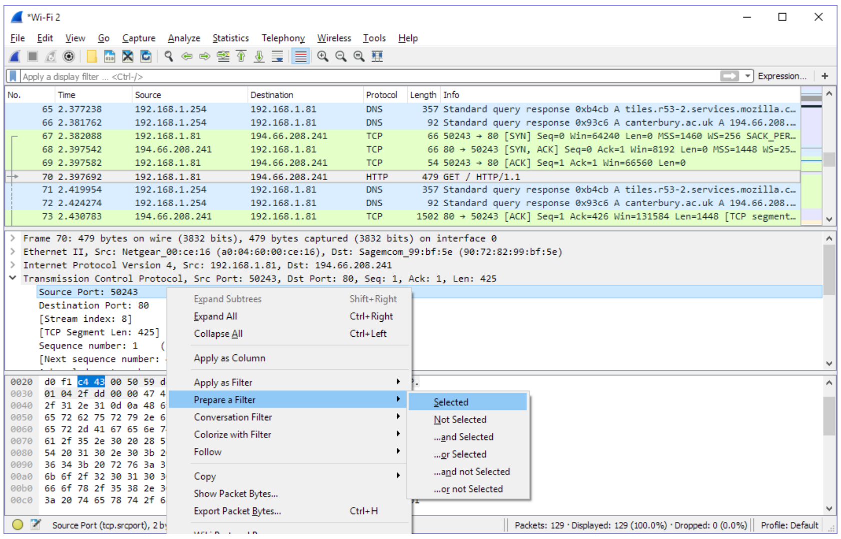
QUESTIONS:

☞ What are the manufacturers of your PC's Network Interface Card (NIC), and the server's NIC (NB: While the MAC address always exists, not all NICs report a manufacturer)?

☞ What are the hex values (shown in the raw bytes panel) of the two NIC manufacturer's OUIs?

2.2 Wireshark Display Filters

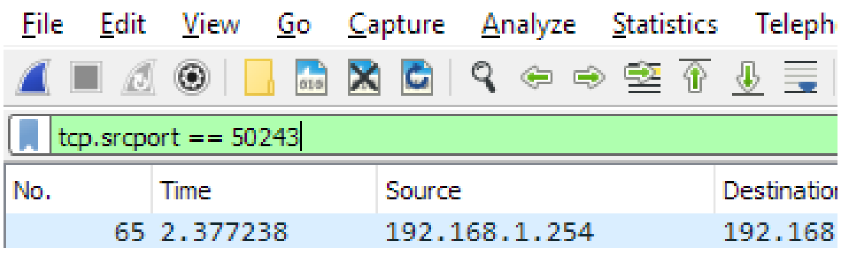
Right click on the **Source Port** field in the **Packet Details Panel**. Select Prepare a Filter » Selected.



```
No.  Time      Source          Destination     Protocol Length Info
  65 2.377238  192.168.1.254   192.168.1.81    DNS       357 Standard query response 0xb4cb A tiles.r53-2.services.mozilla.c…
  66 2.381762  192.168.1.254   192.168.1.81    DNS        92 Standard query response 0x93c6 A canterbury.ac.uk A 194.66.208.…
  67 2.382088  192.168.1.81    194.66.208.241  TCP        66 50243 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PER…
  68 2.397542  194.66.208.241  192.168.1.81    TCP        66 80 → 50243 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1448 WS=25…
  69 2.397582  192.168.1.81    194.66.208.241  TCP        54 50243 → 80 [ACK] Seq=1 Ack=1 Win=66560 Len=0
  70 2.397692  192.168.1.81    194.66.208.241  HTTP      479 GET / HTTP/1.1
  71 2.419954  192.168.1.254   192.168.1.81    DNS       357 Standard query response 0xb4cb A tiles.r53-2.services.mozilla.c…
  72 2.424274  192.168.1.254   192.168.1.81    DNS        92 Standard query response 0x93c6 A canterbury.ac.uk A 194.66.208.…
  73 2.430783  194.66.208.241  192.168.1.81    TCP      1502 80 → 50243 [ACK] Seq=1 Ack=426 Win=131584 Len=1448 [TCP segment…

> Frame 70: 479 bytes on wire (3832 bits), 479 bytes captured (3832 bits) on interface 0
> Ethernet II, Src: Netgear_00:ce:16 (a0:04:60:00:ce:16), Dst: Sagemcom_99:bf:5e (90:72:82:99:bf:5e)
> Internet Protocol Version 4, Src: 192.168.1.81, Dst: 194.66.208.241
v Transmission Control Protocol, Src Port: 50243, Dst Port: 80, Seq: 1, Ack: 1, Len: 425
    Source Port: 50243
    Destination Port: 80
    [Stream index: 8]
    [TCP Segment Len: 425]
    Sequence number: 1    (
    [Next sequence number:

0020  d0 f1 c4 43 00 50 59 d
0030  01 04 2f dd 00 00 47 4
0040  2f 31 2e 31 0d 0a 48 6
0050  65 72 62 75 72 79 2e 6
0060  65 72 2d 41 67 65 6e 7
0070  61 2f 35 2e 30 20 28 5
0080  54 20 31 30 2e 30 3b 2
0090  36 34 3b 20 72 76 3a 3
00a0  6b 6f 2f 32 30 31 30 3
00b0  66 6f 78 2f 35 38 2e 3
00c0  3a 20 74 65 78 74 2f 6

Source Port (tcp.srcport), 2 by    Packets: 129 · Displayed: 129 (100.0%) · Dropped: 0 (0.0%)    Profile: Default
```

Figure 5: Select to Prepare a Filter

Wireshark automatically generates a **Display Filter**, and applies it to the capture. The filter is shown in the **Filter Bar**, below the button toolbar. Only packets captured with a Source Port of the value selected should be displayed. The window should be similar to that shown in figure 6. This same process can be performed on most fields within Wireshark, and can be used to include or exclude traffic.



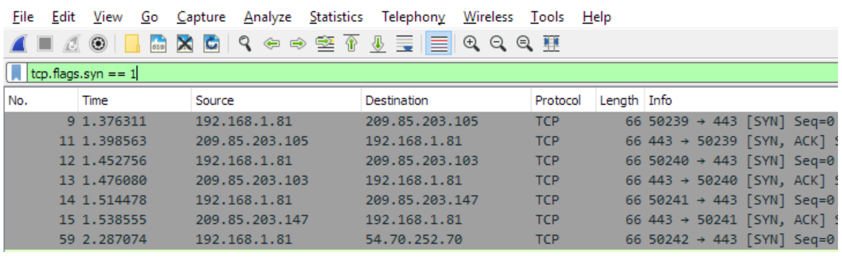
```
tcp.srcport == 50243

No.  Time      Source         Destinatio
  65 2.377238  192.168.1.254  192.168
```

Figure 6: Prepared filter applied with source port from selected package

2.3 Analysing a TCP Session using Wireshark

Start a capture, and generate some Web traffic by going to the webserver in the lab, then stop the capture. Scroll back to the top of the capture trace. Find the first SYN packet, sent from your PC to the webserver. This signifies the start of a TCP 3-way handshake.

If you're having trouble finding the first SYN packet, click into the **Filter Bar** and start typing **tcp.flags.** (at this point you should get a list of the flags to choose from). Choose the correct flag, **tcp.flags.syn** and add **== 1** at the end of it and press Enter or click on the right-pointing arrow next to the filter; now the first SYN packet in the trace should be highlighted as seen in figure 7. On a side-note, you can also find other valuable information such as Hex signatures (e.g., malware signatures) and Strings (e.g., protocol commands) by clicking on the Edit 〉 Find Packet menu option. Select the **Display Filter** drop-down list and choose from the available search options (Display Filter, Hex, String, Regular Expression).


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.flags.syn == 1
```

```
No.  Time        Source          Destination     Protocol Length Info
   9 1.376311    192.168.1.81    209.85.203.105  TCP          66 50239 → 443 [SYN] Seq=0
  11 1.398563    209.85.203.105  192.168.1.81    TCP          66 443 → 50239 [SYN, ACK] S
  12 1.452756    192.168.1.81    209.85.203.103  TCP          66 50240 → 443 [SYN] Seq=0
  13 1.476080    209.85.203.103  192.168.1.81    TCP          66 443 → 50240 [SYN, ACK] S
  14 1.514478    192.168.1.81    209.85.203.147  TCP          66 50241 → 443 [SYN] Seq=0
  15 1.538555    209.85.203.147  192.168.1.81    TCP          66 443 → 50241 [SYN, ACK] S
  59 2.287074    192.168.1.81    54.70.252.70    TCP          66 50242 → 443 [SYN] Seq=0
```

Figure 7: Filter applied to only show SYN packets

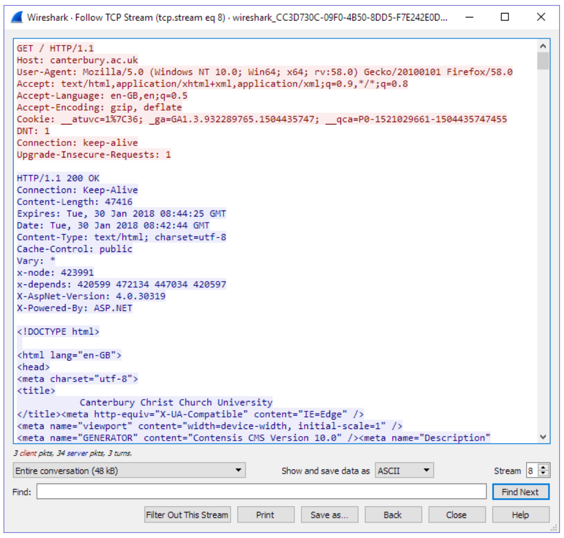
QUESTION:

☞ Can you identify the rest of the TCP 3-way handshake easily? (if not read on)

A quick way to create a **Wireshark Display Filter** to isolate a TCP stream is to right click on a packet in the **Packet List Panel** and select **Follow TCP Stream**. This creates an automatic Display Filter which displays packets from that TCP session only.

It also pops up a session display window, containing by default, an ASCII representation of the TCP session with the client packets in red and the server packets in blue.

The window should look something like Figure 8. This is very useful for viewing human readable protocol payloads, such as HTTP, SMTP, and FTP.


Wireshark · Follow TCP Stream (tcp.stream eq 8) · wireshark_CC3D730C-09F0-4B50-8DD5-F7E242E0D...
Find Next
Filter Out This Stream
Print
Save as...
Back
Close
Help

```
GET / HTTP/1.1
Host: canterbury.ac.uk
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: __atuvc=1%7C36; _ga=GA1.3.932289765.1504435747; __qca=P0-1521029661-1504435747455
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Length: 47416
Expires: Tue, 30 Jan 2018 08:44:25 GMT
Date: Tue, 30 Jan 2018 08:42:44 GMT
Content-Type: text/html; charset=utf-8
Cache-Control: public
Vary: *
x-node: 423991
x-depends: 420599 472134 447034 420597
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET

<!DOCTYPE html>

<html lang="en-GB">
<head>
<meta charset="utf-8">
<title>
            Canterbury Christ Church University
</title><meta http-equiv="X-UA-Compatible" content="IE=Edge" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<meta name="GENERATOR" content="Contensis CMS Version 10.0" /><meta name="Description"
```

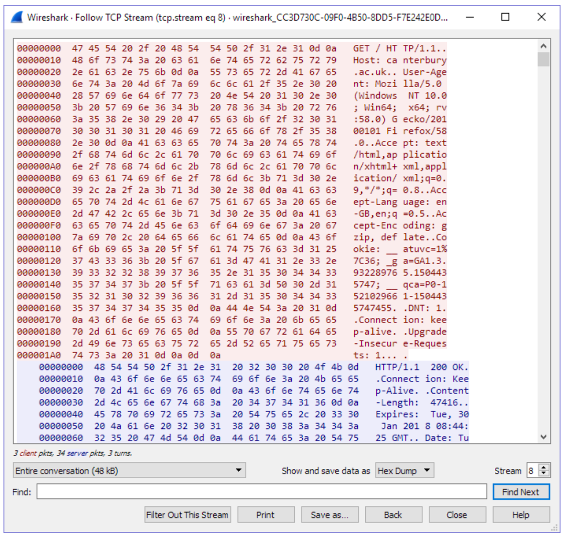
3 client pkts, 34 server pkts, 3 turns.

Entire conversation (48 kB) — Show and save data as: ASCII — Stream: 8

Find:

Figure 8: Follow TCP stream window

Change to Hex Dump Mode and view the payloads in raw Hex, as shown below.



Wireshark · Follow TCP Stream (tcp.stream eq 8) · wireshark_CC3D730C-09F0-4B50-8DD5-F7E242E0D...

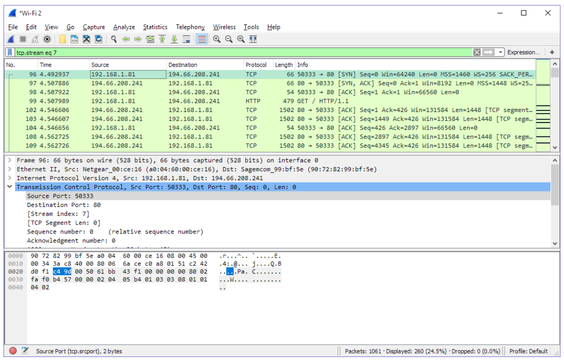
```
00000000  47 45 54 20 2f 20 48 54  54 50 2f 31 2e 31 0d 0a   GET / HT TP/1.1..
00000010  48 6f 73 74 3a 20 63 61  6e 74 65 72 62 75 72 79   Host: ca nterbury
00000020  2e 61 63 2e 75 6b 0d 0a  55 73 65 72 2d 41 67 65   .ac.uk.. User-Age
00000030  6e 74 3a 20 4d 6f 7a 69  6c 6c 61 2f 35 2e 30 20   nt: Mozi lla/5.0 
00000040  28 57 69 6e 64 6f 77 73  20 4e 54 20 31 30 2e 30   (Windows  NT 10.0
00000050  3b 20 57 69 6e 36 34 3b  20 78 36 34 3b 20 72 76   ; Win64;  x64; rv
00000060  3a 35 38 2e 30 29 20 47  65 63 6b 6f 2f 32 30 31   :58.0) G ecko/201
00000070  30 30 31 30 31 20 46 69  72 65 66 6f 78 2f 35 38   00101 Fi refox/58
00000080  2e 30 0d 0a 41 63 63 65  70 74 3a 20 74 65 78 74   .0..Acce pt: text
00000090  2f 68 74 6d 6c 2c 61 70  70 6c 69 63 61 74 69 6f   /html,ap plicatio
000000A0  6e 2f 78 68 74 6d 6c 2b  78 6d 6c 2c 61 70 70 6c   n/xhtml+ xml,appl
000000B0  69 63 61 74 69 6f 6e 2f  78 6d 6c 3b 71 3d 30 2e   ication/ xml;q=0.
000000C0  39 2c 2a 2f 2a 3b 71 3d  30 2e 38 0d 0a 41 63 63   9,*/*;q= 0.8..Acc
000000D0  65 70 74 2d 4c 61 6e 67  75 61 67 65 3a 20 65 6e   ept-Lang uage: en
000000E0  2d 47 42 2c 65 6e 3b 71  3d 30 2e 35 0d 0a 41 63   -GB,en;q =0.5..Ac
000000F0  63 65 70 74 2d 45 6e 63  6f 64 69 6e 67 3a 20 67   cept-Enc oding: g
00000100  7a 69 70 2c 20 64 65 66  6c 61 74 65 0d 0a 43 6f   zip, def late..Co
00000110  6f 6b 69 65 3a 20 5f 5f  61 74 75 76 63 3d 31 25   okie: __ atuvc=1%
00000120  37 43 33 36 3b 20 5f 67  61 3d 47 41 31 2e 33 2e   7C36; _g a=GA1.3.
00000130  39 33 32 32 38 39 37 36  35 2e 31 35 30 34 34 33   93228976 5.150443
00000140  35 37 34 37 3b 20 5f 5f  71 63 61 3d 50 30 2d 31   5747; __ qca=P0-1
00000150  35 32 31 30 32 39 36 36  31 2d 31 35 30 34 34 33   52102966 1-150443
00000160  35 37 34 37 34 35 35 0d  0a 44 4e 54 3a 20 31 0d   5747455. .DNT: 1.
00000170  0a 43 6f 6e 6e 65 63 74  69 6f 6e 3a 20 6b 65 65   .Connect ion: kee
00000180  70 2d 61 6c 69 76 65 0d  0a 55 70 67 72 61 64 65   p-alive. .Upgrade
00000190  2d 49 6e 73 65 63 75 72  65 2d 52 65 71 75 65 73   -Insecur e-Reques
000001A0  74 73 3a 20 31 0d 0a 0d  0a                        ts: 1... .
    00000000  48 54 54 50 2f 31 2e 31  20 32 30 30 20 4f 4b 0d   HTTP/1.1  200 OK.
    00000010  0a 43 6f 6e 6e 65 63 74  69 6f 6e 3a 20 4b 65 65   .Connect ion: Kee
    00000020  70 2d 41 6c 69 76 65 0d  0a 43 6f 6e 74 65 6e 74   p-Alive. .Content
    00000030  2d 4c 65 6e 67 74 68 3a  20 34 37 34 31 36 0d 0a   -Length:  47416..
    00000040  45 78 70 69 72 65 73 3a  20 54 75 65 2c 20 33 30   Expires:  Tue, 30
    00000050  20 4a 61 6e 20 32 30 31  38 20 30 38 3a 34 34 3a    Jan 201 8 08:44:
    00000060  32 35 20 47 4d 54 0d 0a  44 61 74 65 3a 20 54 75   25 GMT.. Date: Tu
```

3 client pkts, 34 server pkts, 3 turns.

Entire conversation (48 kB) — Show and save data as: Hex Dump — Stream: 8

Figure 9: Follow TCP stream window – Hex view

Wireshark is capable of finding TCP streams and will be able to present those to you with a stream ID. You can choose which stream to look at by clicking on the up and down arrow next to **Stream** (on the bottom right). Ensure that you have selected the stream which had your 3-way handshake. Close the popup window. Wireshark now only shows the packets from the selected TCP Stream. You should be able to identify the 3-way handshake easily now.



*Wi-Fi 2

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

tcp.stream eq 7

```
No.   Time      Source          Destination     Protocol Length Info
   96 4.492937  192.168.1.81    194.66.208.241  TCP        66 50333 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PER…
   97 4.507886  194.66.208.241  192.168.1.81    TCP        66 80 → 50333 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1448 WS=25…
   98 4.507922  192.168.1.81    194.66.208.241  TCP        54 50333 → 80 [ACK] Seq=1 Ack=1 Win=66560 Len=0
   99 4.507989  192.168.1.81    194.66.208.241  HTTP      479 GET / HTTP/1.1
  102 4.546606  194.66.208.241  192.168.1.81    TCP      1502 80 → 50333 [ACK] Seq=1 Ack=426 Win=131584 Len=1448 [TCP segment…
  103 4.546607  194.66.208.241  192.168.1.81    TCP      1502 80 → 50333 [ACK] Seq=1449 Ack=426 Win=131584 Len=1448 [TCP segm…
  104 4.546656  192.168.1.81    194.66.208.241  TCP        54 50333 → 80 [ACK] Seq=426 Ack=2897 Win=66560 Len=0
  108 4.562725  194.66.208.241  192.168.1.81    TCP      1502 80 → 50333 [ACK] Seq=2897 Ack=426 Win=131584 Len=1448 [TCP segm…
  109 4.562726  194.66.208.241  192.168.1.81    TCP      1502 80 → 50333 [ACK] Seq=4345 Ack=426 Win=131584 Len=1448 [TCP segm…
```

```
> Frame 96: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: Netgear_00:ce:16 (a0:04:60:00:ce:16), Dst: Sagemcom_99:bf:5e (90:72:82:99:bf:5e)
> Internet Protocol Version 4, Src: 192.168.1.81, Dst: 194.66.208.241
v Transmission Control Protocol, Src Port: 50333, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 50333
    Destination Port: 80
    [Stream index: 7]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 0
```

```
0000  90 72 82 99 bf 5e a0 04  60 00 ce 16 08 00 45 00   .r...^.. `.....E.
0010  00 34 3a c8 40 00 80 06  6a ce c0 a8 01 51 c2 42   .4:.@... j....Q.B
0020  d0 f1 c4 9d 00 50 61 bb  43 f1 00 00 00 00 80 02   .....Pa. C.......
0030  fa f0 b4 57 00 00 02 04  05 b4 01 03 03 08 01 01   ...W.... ........
0040  04 02                                              ..
```

Source Port (tcp.srcport), 2 bytes | Packets: 1061 · Displayed: 260 (24.5%) · Dropped: 0 (0.0%) | Profile: Default

Figure 10: Follow TCP stream ID

Note: Wireshark has automatically created a **Display Filter** to filter out this TCP conversation. In this case: **(tcp.stream eq 7)**

QUESTIONS:

☞ From your Wireshark Capture, fill in the diagram below with the IP Addresses and Port Numbers for the Client and the Server

☞ For each packet in the TCP 3-way handshake, fill in the Sequence and Acknowledgement numbers on the diagram below.

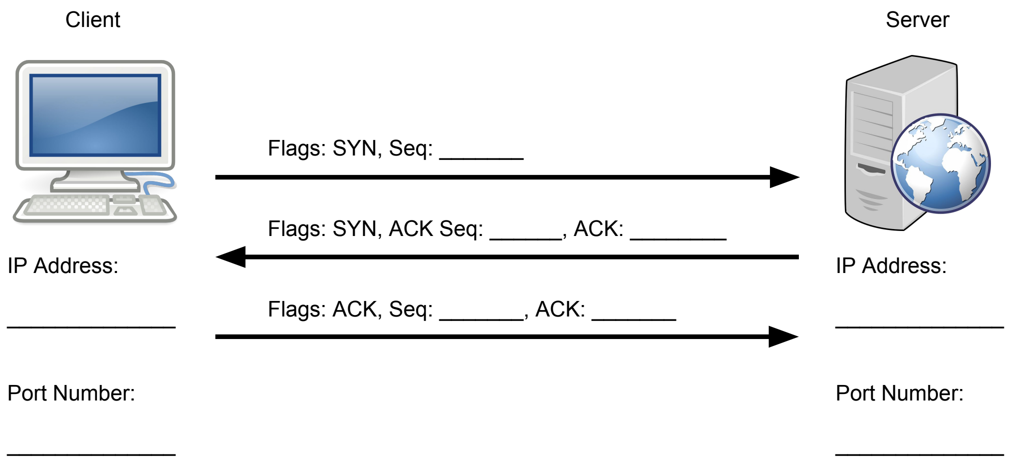


Figure 11: Sheet SYN ACK Fillout

2.4 Saving Packet Captures

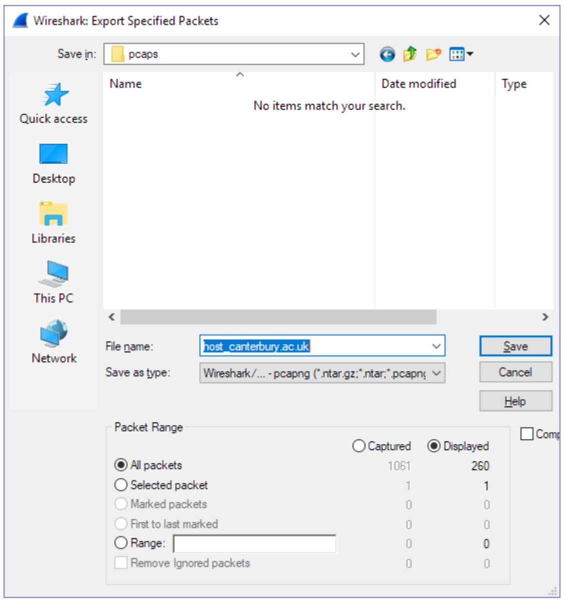
Often captures should be saved to disk, for later analysis. To save a capture, select File » Save As and save the trace. By default this creates a Wireshark **pcapng** file, or if you select **pcap**, a file many tools can understand. For example a `tcpdump` output file in this format can be read into Wireshark for analysis. This saves all the captured packets to the file.

QUESTIONS:

☞ Did you successfully save your capture to disk? ☞ Copy the **Display Filter** into the clipboard, and close and start Wireshark again, then reload the file. Was the whole capture saved or just the displayed packets?

Paste the display filter back into the Filter Bar, and Apply it.

To save *only the displayed packets*, select File 〉Export Specified Packets, and make sure the **Displayed** radio button is selected rather than the **Captured** option. This creates a **pcap** file, with only the packets filtered by the current display filter.



No items match your search.

File name: host_canterbury.ac.uk

Save as type: Wireshark/... - pcapng (*.ntar.gz;*.ntar;*.pcapng

Packet Range

	Captured	[✓ Displayed]
[✓ All packets]	1061	260
Selected packet	1	1
Marked packets	0	0
First to last marked	0	0
Range:	0	0
Remove Ignored packets	0	0

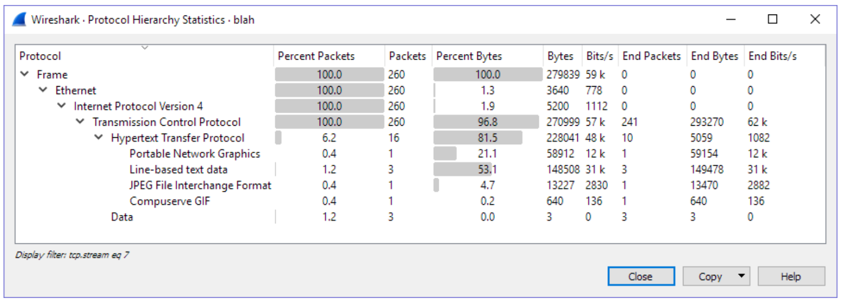
Figure 12: Save Dialogue

QUESTION:

Close and start Wireshark again, then reload the file. Was the whole capture saved or just the displayed packets?

2.5 Wireshark Statistics

Start the capture, and generate some Web traffic by going to the webserver in the lab, then stop the capture, and select the Statistics 〉Protocol Hierarchy menu option. A window similar to Figure 13 should be shown, displaying statistics about the **pcap**. Note that all the packets are Ethernet (Local Area Network) packets, but at the network layer all of the packets are TCP in this capture, but usually some are also UDP.



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	260	100.0	279839	59 k	0	0	0
Ethernet	100.0	260	1.3	3640	778	0	0	0
Internet Protocol Version 4	100.0	260	1.9	5200	1112	0	0	0
Transmission Control Protocol	100.0	260	96.8	270999	57 k	241	293270	62 k
Hypertext Transfer Protocol	6.2	16	81.5	228041	48 k	10	5059	1082
Portable Network Graphics	0.4	1	21.1	58912	12 k	1	59154	12 k
Line-based text data	1.2	3	53.1	148508	31 k	3	149478	31 k
JPEG File Interchange Format	0.4	1	4.7	13227	2830	1	13470	2882
Compuserve GIF	0.4	1	0.2	640	136	1	640	136
Data	1.2	3	0.0	3	0	3	3	0

Display filter: tcp.stream eq 7

Figure 13: Wireshark Statistics

QUESTIONS:

☞ What percentage of packets in your capture are TCP, and give an example of the higher level protocol which uses TCP?

☞ What percentage of packets in your capture are UDP, and give an example of the higher level protocol which uses UDP? (use the figure below)

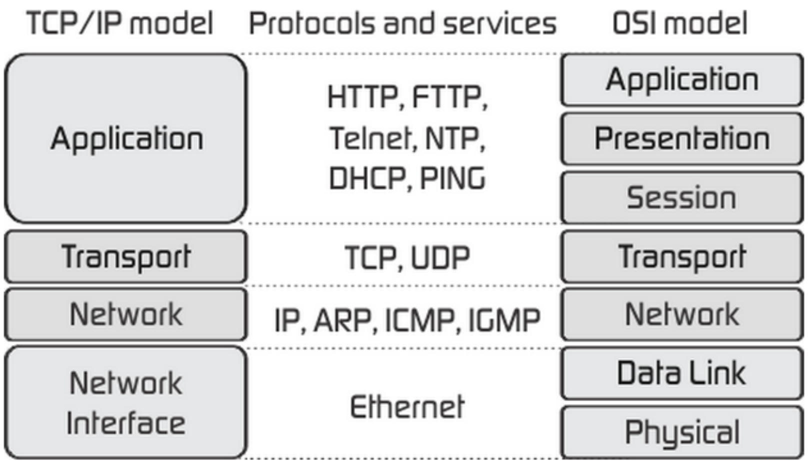
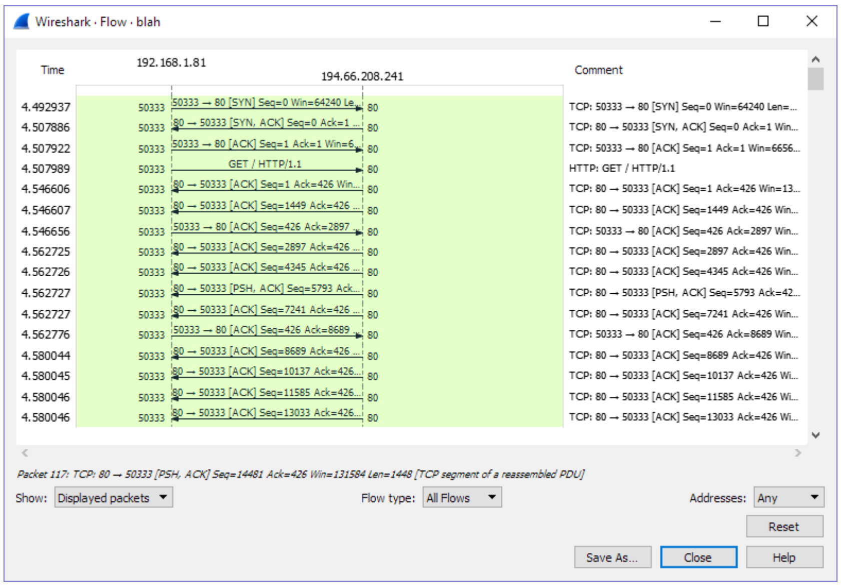


Figure 14: OSI Model and TCP/IP Model

Select the Statistics › Flow Graph menu option. Choose **General Flow** and **Network Source** options, and click on the **OK** button. A window similar to Figure 15 should be displayed, showing the flow of traffic.



Time	192.168.1.81		194.66.208.241	Comment
4.492937	50333	50333 → 80 [SYN] Seq=0 Win=64240 Le...	80	TCP: 50333 → 80 [SYN] Seq=0 Win=64240 Len=...
4.507886	50333	80 → 50333 [SYN, ACK] Seq=0 Ack=1 ...	80	TCP: 80 → 50333 [SYN, ACK] Seq=0 Ack=1 Win...
4.507922	50333	50333 → 80 [ACK] Seq=1 Ack=1 Win=6...	80	TCP: 50333 → 80 [ACK] Seq=1 Ack=1 Win=6656...
4.507989	50333	GET / HTTP/1.1	80	HTTP: GET / HTTP/1.1
4.546606	50333	80 → 50333 [ACK] Seq=1 Ack=426 Win...	80	TCP: 80 → 50333 [ACK] Seq=1 Ack=426 Win=13...
4.546607	50333	80 → 50333 [ACK] Seq=1449 Ack=426 ...	80	TCP: 80 → 50333 [ACK] Seq=1449 Ack=426 Win...
4.546656	50333	50333 → 80 [ACK] Seq=426 Ack=2897 ...	80	TCP: 50333 → 80 [ACK] Seq=426 Ack=2897 Win...
4.562725	50333	80 → 50333 [ACK] Seq=2897 Ack=426 ...	80	TCP: 80 → 50333 [ACK] Seq=2897 Ack=426 Win...
4.562726	50333	80 → 50333 [ACK] Seq=4345 Ack=426 ...	80	TCP: 80 → 50333 [ACK] Seq=4345 Ack=426 Win...
4.562727	50333	80 → 50333 [PSH, ACK] Seq=5793 Ack...	80	TCP: 80 → 50333 [PSH, ACK] Seq=5793 Ack=42...
4.562727	50333	80 → 50333 [ACK] Seq=7241 Ack=426 ...	80	TCP: 80 → 50333 [ACK] Seq=7241 Ack=426 Win...
4.562776	50333	50333 → 80 [ACK] Seq=426 Ack=8689 ...	80	TCP: 50333 → 80 [ACK] Seq=426 Ack=8689 Win...
4.580044	50333	80 → 50333 [ACK] Seq=8689 Ack=426 ...	80	TCP: 80 → 50333 [ACK] Seq=8689 Ack=426 Win...
4.580045	50333	80 → 50333 [ACK] Seq=10137 Ack=426...	80	TCP: 80 → 50333 [ACK] Seq=10137 Ack=426 Wi...
4.580046	50333	80 → 50333 [ACK] Seq=11585 Ack=426...	80	TCP: 80 → 50333 [ACK] Seq=11585 Ack=426 Wi...
4.580046	50333	80 → 50333 [ACK] Seq=13033 Ack=426...	80	TCP: 80 → 50333 [ACK] Seq=13033 Ack=426 Wi...

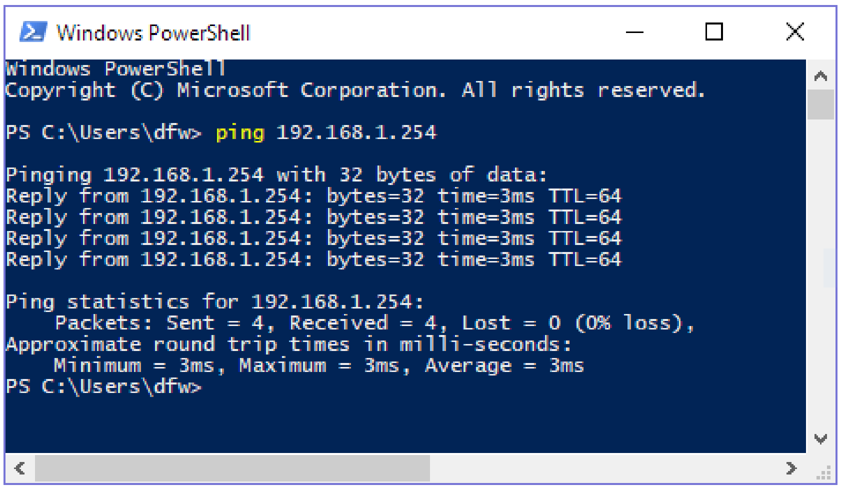
Packet 117: TCP: 80 → 50333 [PSH, ACK] Seq=14481 Ack=426 Win=131584 Len=1448 [TCP segment of a reassembled PDU]

Show: Displayed packets Flow type: All Flows Addresses: Any

Figure 15: Wireshark Flow Graph

2.6 Capture ARP & ICMP Protocol Traffic using Wireshark

Start a Wireshark capture. Open a Windows console window, and generate some ICMP traffic by using the `ping` command line tool to check the connectivity of a neighbouring machine (or your home router).



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\dfw> ping 192.168.1.254

Pinging 192.168.1.254 with 32 bytes of data:
Reply from 192.168.1.254: bytes=32 time=3ms TTL=64
Reply from 192.168.1.254: bytes=32 time=3ms TTL=64
Reply from 192.168.1.254: bytes=32 time=3ms TTL=64
Reply from 192.168.1.254: bytes=32 time=3ms TTL=64

Ping statistics for 192.168.1.254:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 3ms, Average = 3ms
PS C:\Users\dfw>
```

Figure 16: Windows `ping` command

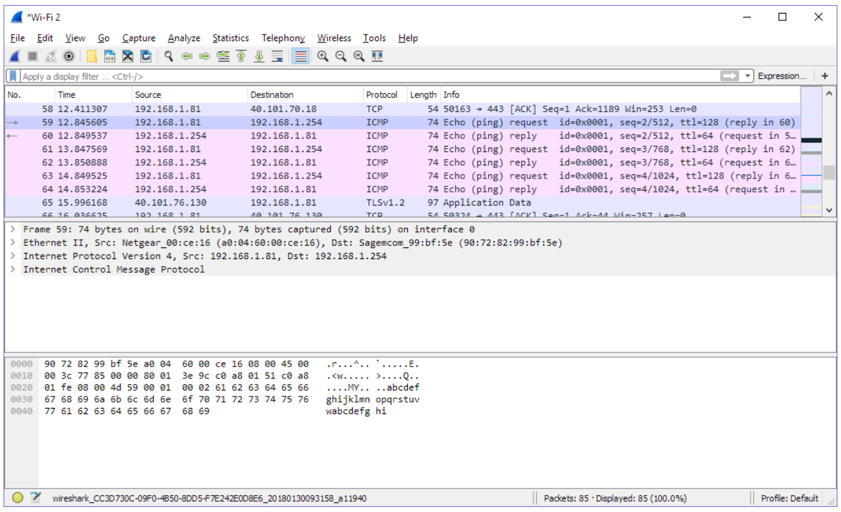
Stop the capture and Wireshark should now look something like figure 16.

The Address Resolution Protocol (ARP) and ICMP packets are difficult to pick out; create a

display filter to only show ARP and ICMP packets.

Some useful Wireshark **display filgers** can be found at:

`http://wiki.wireshark.org/DisplayFilters`


*Wi-Fi 2
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.   Time        Source          Destination     Protocol  Length Info
   58 12.411307   192.168.1.81    40.101.70.18    TCP       54 50163 → 443 [ACK] Seq=1 Ack=1189 Win=253 Len=0
→  59 12.845605   192.168.1.81    192.168.1.254   ICMP      74 Echo (ping) request  id=0x0001, seq=2/512, ttl=128 (reply in 60)
←  60 12.849537   192.168.1.254   192.168.1.81    ICMP      74 Echo (ping) reply    id=0x0001, seq=2/512, ttl=64 (request in 5…
   61 13.847569   192.168.1.81    192.168.1.254   ICMP      74 Echo (ping) request  id=0x0001, seq=3/768, ttl=128 (reply in 62)
   62 13.850888   192.168.1.254   192.168.1.81    ICMP      74 Echo (ping) reply    id=0x0001, seq=3/768, ttl=64 (request in 6…
   63 14.849525   192.168.1.81    192.168.1.254   ICMP      74 Echo (ping) request  id=0x0001, seq=4/1024, ttl=128 (reply in 6…
   64 14.853224   192.168.1.254   192.168.1.81    ICMP      74 Echo (ping) reply    id=0x0001, seq=4/1024, ttl=64 (request in …
   65 15.996168   40.101.76.130   192.168.1.81    TLSv1.2   97 Application Data
   66 16.036625   192.168.1.81    40.101.76.130   TCP       54 50324 → 443 [ACK] Seq=1 Ack=44 Win=257 Len=0

> Frame 59: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: Netgear_00:ce:16 (a0:04:60:00:ce:16), Dst: Sagemcom_99:bf:5e (90:72:82:99:bf:5e)
> Internet Protocol Version 4, Src: 192.168.1.81, Dst: 192.168.1.254
> Internet Control Message Protocol

0000  90 72 82 99 bf 5e a0 04  60 00 ce 16 08 00 45 00   .r...^.. `.....E.
0010  00 3c 77 85 00 00 80 01  3e 9c c0 a8 01 51 c0 a8   .<w..... >....Q..
0020  01 fe 08 00 4d 59 00 01  00 02 61 62 63 64 65 66   ....MY.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi

wireshark_CC3D730C-09F0-4B50-8DD5-F7E242E0D8E6_20180130093158_a11940   Packets: 85 · Displayed: 85 (100.0%)   Profile: Default
```

Figure 17: ICMP Packets

Note the results in Wireshark. The initial ARP request broadcast from your PC determines the physical MAC address of the network IP Address, and the ARP reply from the neighbouring system. After the ARP request, the pings (ICMP echo request and replies) can be seen.

QUESTIONS:

☞ After the first ping command, are the ARP and ICMP packets captured by Wireshark?

YES/NO

☞ After the second or third ping command, are the ARP and ICMP packets captured by Wireshark?

ARP: YES/NO

ICMP: YES/NO

☞ Why is this?

If pinging the same system more than once, delete the ARP cache on your system, using the `arp` command, as shown below, so a new ARP request will be generated:

On Windows systems you can type, for example:

```
C:\> ping 10.0.0.2

... ping output ...

C:\> arp -d *
```

On Linux systems you can type, for example:

```
user@hostname:~$ ping 10.0.0.2

... ping output ...

user@hostname:~$ sudo ip -s -s neigh flush all
```

Note the results in Wireshark. The initial ARP request broadcast from your PC determines the physical MAC address of the network IP Address `10.0.0.2`, and the ARP reply from the neighbouring system. After the ARP request the pings (ICMP echo request and replies) can be seen.