Wireshark

Wireshark is a network packet analyzer, which can be downloaded for use at home from https://www.wireshark.org/#download. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

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

In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, all that has changed. Wireshark is perhaps one of the best open source packet analyzers available today. Here are some examples people use Wireshark for:

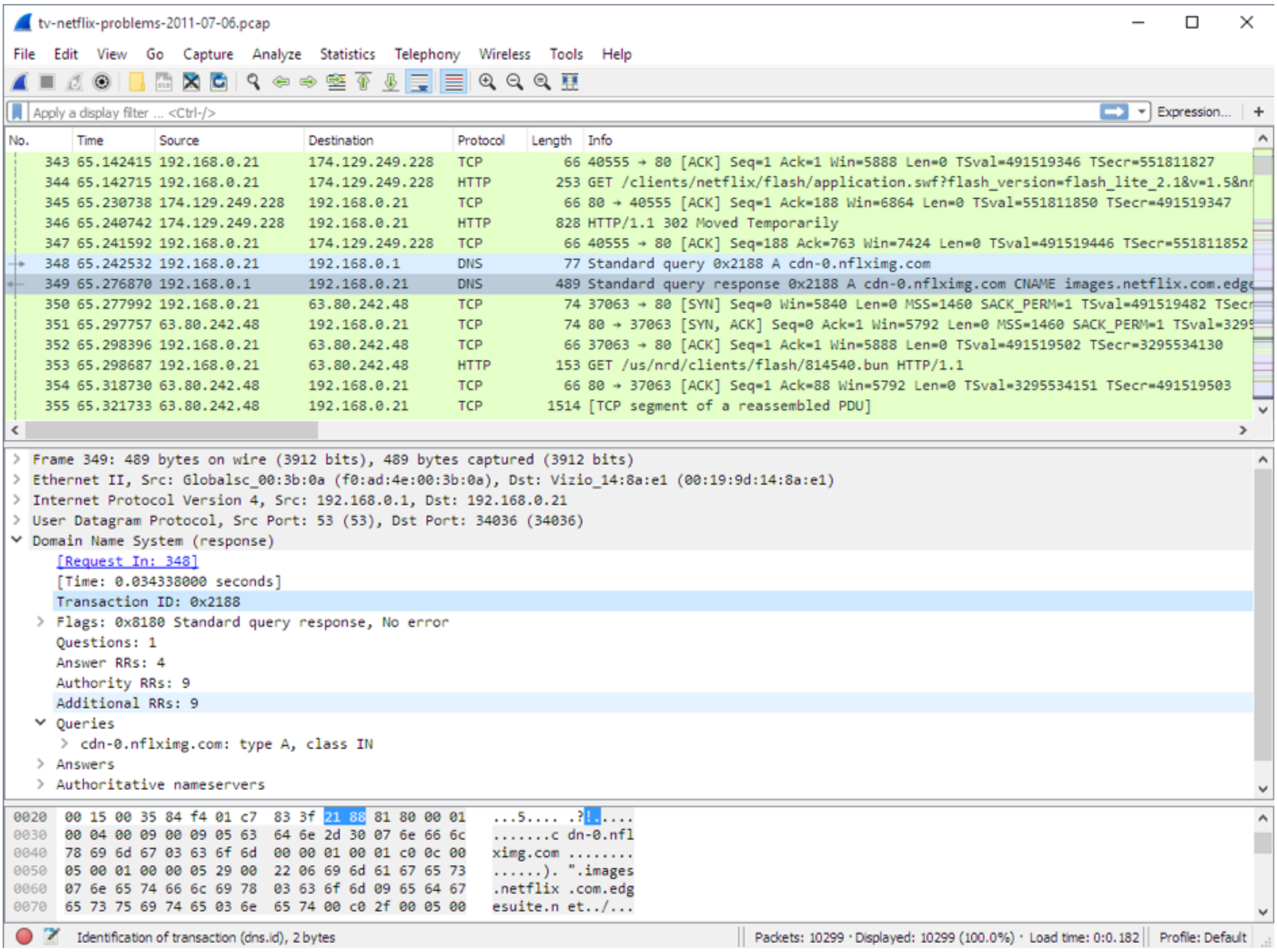
- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- QA engineers use it to verify network applications
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

Beside these examples Wireshark can be helpful in many other situations too. The following are some of the many features Wireshark provides:

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

However, to really appreciate its power you have to start using it.

Figure 1 shows Wireshark having captured some packets and waiting for you to examine them.



```
No.    Time      Source          Destination     Protocol Length Info
   343 65.142415 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
   344 65.142715 192.168.0.21    174.129.249.228 HTTP      253 GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
   345 65.230738 174.129.249.228 192.168.0.21    TCP        66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
   346 65.240742 174.129.249.228 192.168.0.21    HTTP      828 HTTP/1.1 302 Moved Temporarily
   347 65.241592 192.168.0.21    174.129.249.228 TCP        66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
   348 65.242532 192.168.0.21    192.168.0.1     DNS        77 Standard query 0x2188 A cdn-0.nflximg.com
   349 65.276870 192.168.0.1     192.168.0.21    DNS       489 Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
   350 65.277992 192.168.0.21    63.80.242.48    TCP        74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
   351 65.297757 63.80.242.48    192.168.0.21    TCP        74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
   352 65.298396 192.168.0.21    63.80.242.48    TCP        66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
   353 65.298687 192.168.0.21    63.80.242.48    HTTP      153 GET /us/nrd/clients/flash/814540.bun HTTP/1.1
   354 65.318730 63.80.242.48    192.168.0.21    TCP        66 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
   355 65.321733 63.80.242.48    192.168.0.21    TCP      1514 [TCP segment of a reassembled PDU]
```

```
> Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
> Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
v Domain Name System (response)
    [Request In: 348]
    [Time: 0.034338000 seconds]
    Transaction ID: 0x2188
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 4
    Authority RRs: 9
    Additional RRs: 9
  v Queries
    > cdn-0.nflximg.com: type A, class IN
  > Answers
  > Authoritative nameservers
```

```
0020  00 15 00 35 84 f4 01 c7  83 3f 21 88 81 80 00 01   ...5.... .?!.....
0030  00 04 00 09 00 09 05 63  64 6e 2d 30 07 6e 66 6c   .......c dn-0.nfl
0040  78 69 6d 67 03 63 6f 6d  00 00 01 00 01 c0 0c 00   ximg.com ........
0050  05 00 01 00 00 05 29 00  22 06 69 6d 61 67 65 73   ......). ".images
0060  07 6e 65 74 66 6c 69 78  03 63 6f 6d 09 65 64 67   .netflix .com.edg
0070  65 73 75 69 74 65 03 6e  65 74 00 c0 2f 00 05 00   esuite.n et../...
```

```
Identification of transaction (dns.id), 2 bytes    Packets: 10299 · Displayed: 10299 (100.0%) · Load time: 0:0.182    Profile: Default
```

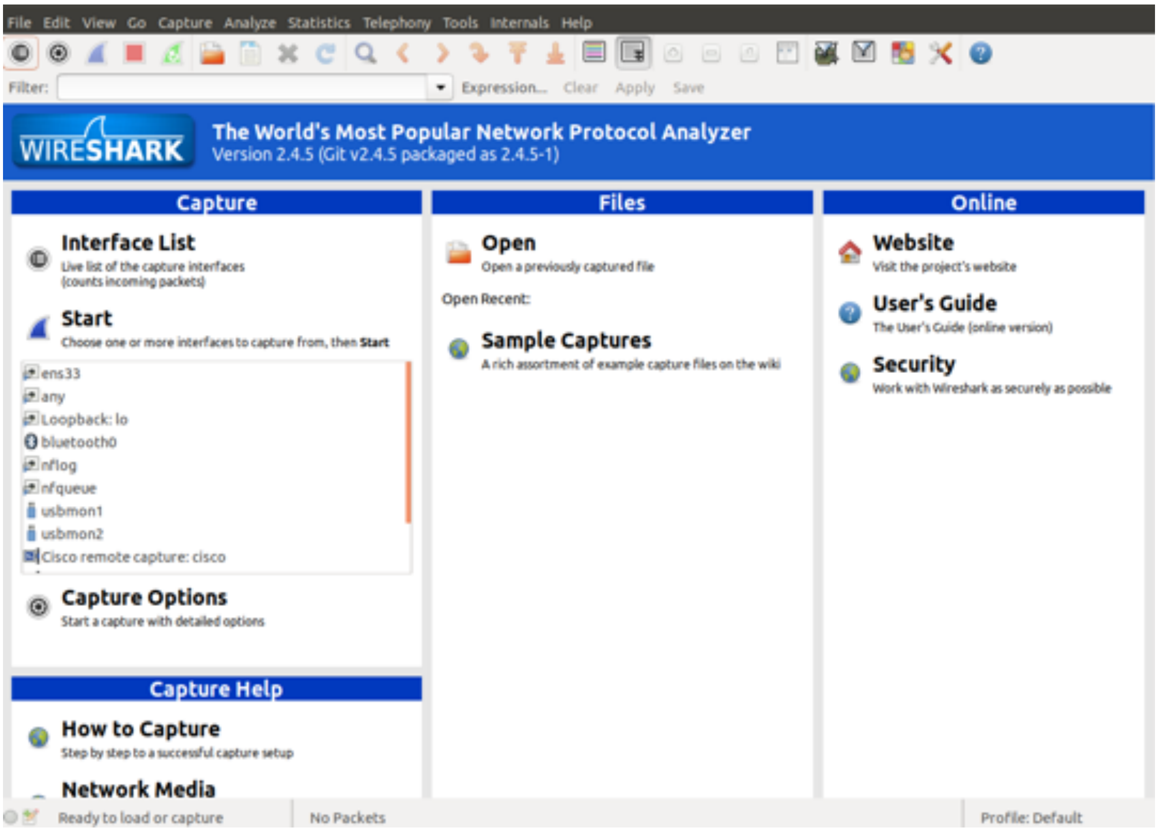
Figure 1 - Wireshark captures packets and lets you examine their contents

Here are some things Wireshark does not provide:

- Wireshark is not an intrusion detection system. It will not warn you when someone does strange things on your network that he/she is not allowed to do. However, if strange things happen, Wireshark might help you figure out what is really going on.
- Wireshark will not manipulate things on the network, it will only "measure" things from it. Wireshark does not send packets on the network or do other active things (except for name resolutions, but even that can be disabled).

The interface

When you start wireshark you should see an interface like the one shown in Figure 2.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
WIRESHARK
Capture
Files
Online
Capture Help
Ready to load or capture
No Packets
Profile: Default

The World's Most Popular Network Protocol Analyzer
Version 2.4.5 (Git v2.4.5 packaged as 2.4.5-1)

Interface List
Live list of the capture interfaces (counts incoming packets)

Start
Choose one or more interfaces to capture from, then **Start**

ens33
any
Loopback: lo
bluetooth0
nflog
nfqueue
usbmon1
usbmon2
Cisco remote capture: cisco

Capture Options
Start a capture with detailed options

How to Capture
Step by step to a successful capture setup

Network Media

Open
Open a previously captured file

Open Recent:

Sample Captures
A rich assortment of example capture files on the wiki

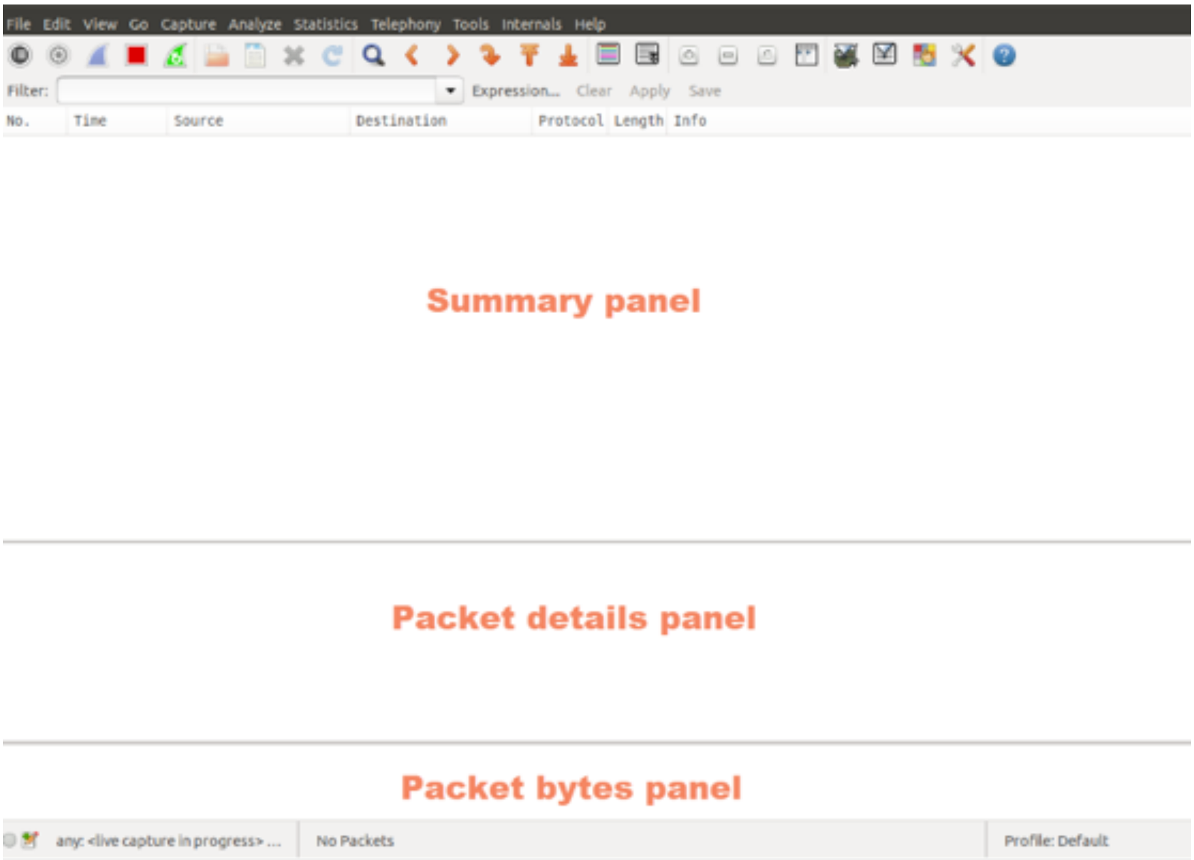
Website
Visit the project's website

User's Guide
The User's Guide (online version)

Security
Work with Wireshark as securely as possible

*Figure 2 - Default start up screen for Wireshark *

A blank screen is shown below.

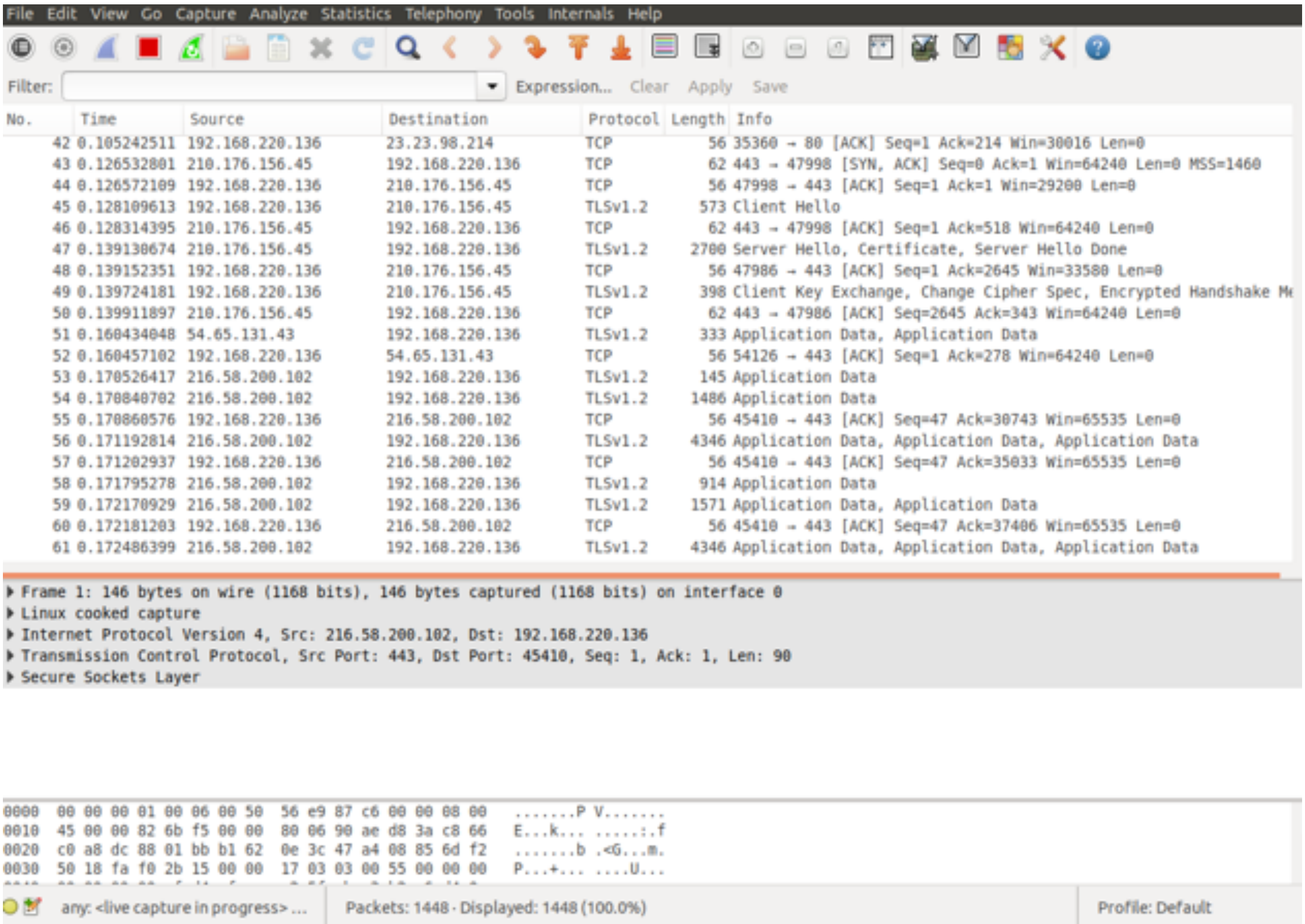

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
Summary panel
Packet details panel
Packet bytes panel
any: <live capture in progress> ...
No Packets
Profile: Default

This screen has three main parts: 1. The summary panel shows the list of packets seen by wireshark; 2.

The packet details displays the packet selected in the summary panel in more details; and 3. The packet bytes pane displays the data from the packet selected in the packet list pane and highlights the field selected in the packet details pane.

You can load precaptured network traces into Wireshark for analysis, we can't actually capture live traffic on our systems because of security restrictions.

Below you see traffic captured earlier by visiting this site: http://bbc.co.uk

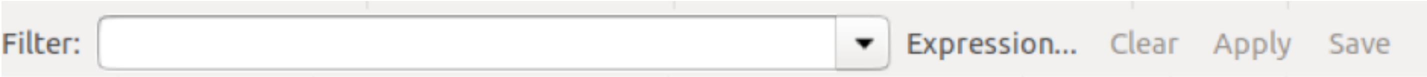

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
Clear
Apply
Save

```
No.     Time         Source            Destination       Protocol Length Info
     42 0.105242511  192.168.220.136   23.23.98.214      TCP        56 35360 → 80 [ACK] Seq=1 Ack=214 Win=30016 Len=0
     43 0.126532801  210.176.156.45    192.168.220.136   TCP        62 443 → 47998 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
     44 0.126572109  192.168.220.136   210.176.156.45    TCP        56 47998 → 443 [ACK] Seq=1 Ack=1 Win=29200 Len=0
     45 0.128109613  192.168.220.136   210.176.156.45    TLSv1.2   573 Client Hello
     46 0.128314395  210.176.156.45    192.168.220.136   TCP        62 443 → 47998 [ACK] Seq=1 Ack=518 Win=64240 Len=0
     47 0.139130674  210.176.156.45    192.168.220.136   TLSv1.2  2700 Server Hello, Certificate, Server Hello Done
     48 0.139152351  192.168.220.136   210.176.156.45    TCP        56 47986 → 443 [ACK] Seq=1 Ack=2645 Win=33580 Len=0
     49 0.139724181  192.168.220.136   210.176.156.45    TLSv1.2   398 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Me
     50 0.139911897  210.176.156.45    192.168.220.136   TCP        62 443 → 47986 [ACK] Seq=2645 Ack=343 Win=64240 Len=0
     51 0.160434048  54.65.131.43      192.168.220.136   TLSv1.2   333 Application Data, Application Data
     52 0.160457102  192.168.220.136   54.65.131.43      TCP        56 54126 → 443 [ACK] Seq=1 Ack=278 Win=64240 Len=0
     53 0.170526417  216.58.200.102    192.168.220.136   TLSv1.2   145 Application Data
     54 0.170840702  216.58.200.102    192.168.220.136   TLSv1.2  1486 Application Data
     55 0.170860576  192.168.220.136   216.58.200.102    TCP        56 45410 → 443 [ACK] Seq=47 Ack=30743 Win=65535 Len=0
     56 0.171192814  216.58.200.102    192.168.220.136   TLSv1.2  4346 Application Data, Application Data, Application Data
     57 0.171202937  192.168.220.136   216.58.200.102    TCP        56 45410 → 443 [ACK] Seq=47 Ack=35033 Win=65535 Len=0
     58 0.171795278  216.58.200.102    192.168.220.136   TLSv1.2   914 Application Data
     59 0.172170929  216.58.200.102    192.168.220.136   TLSv1.2  1571 Application Data, Application Data
     60 0.172181203  192.168.220.136   216.58.200.102    TCP        56 45410 → 443 [ACK] Seq=47 Ack=37406 Win=65535 Len=0
     61 0.172486399  216.58.200.102    192.168.220.136   TLSv1.2  4346 Application Data, Application Data, Application Data

▸ Frame 1: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 216.58.200.102, Dst: 192.168.220.136
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 45410, Seq: 1, Ack: 1, Len: 90
▸ Secure Sockets Layer

0000  00 00 00 01 00 06 00 50  56 e9 87 c6 00 00 08 00   .......P V.......
0010  45 00 00 82 6b f5 00 00  80 06 90 ae d8 3a c8 66   E...k... ......:.f
0020  c0 a8 dc 88 01 bb b1 62  0e 3c 47 a4 08 85 6d f2   .......b .<G...m.
0030  50 18 fa f0 2b 15 00 00  17 03 03 00 55 00 00 00   P...+... ....U...

any: <live capture in progress> ...   Packets: 1448 · Displayed: 1448 (100.0%)   Profile: Default
```

This is showing a stream of packet captured that relate to the web browsing you just did, you should see some of the protocol names discussed in lectures such as TCP, DNS, HTTP etc.

Filtering traffic

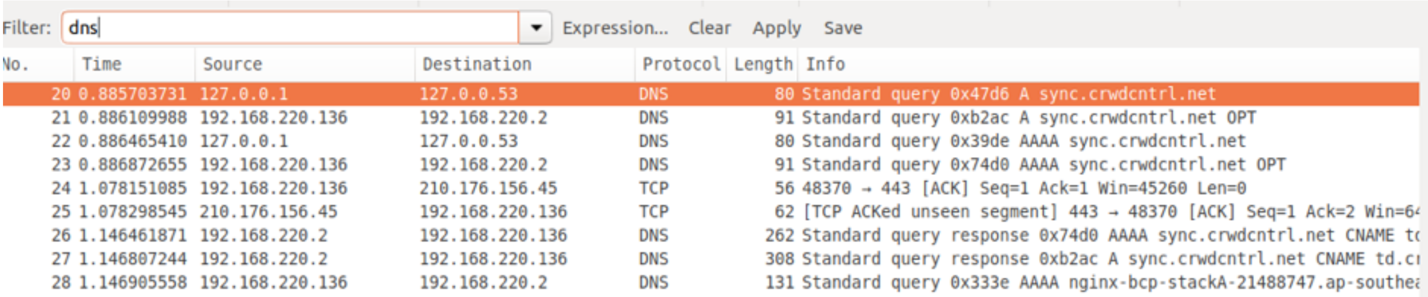
Wireshark will capture a lot of network traffic, on a busy network it will capture more traffic than you can analyse. What if you are only interested in specific traffic, this is where the filter box comes in handy.



This allow you to filter using an expression that can use terms such as the protocol, destination IP address, source IP address and more.

Say that we are interested in looking for DNS lookup traffic. An example would be an infected machine might be trying to contact a command and control centre and we are interested in associated DNS lookups.

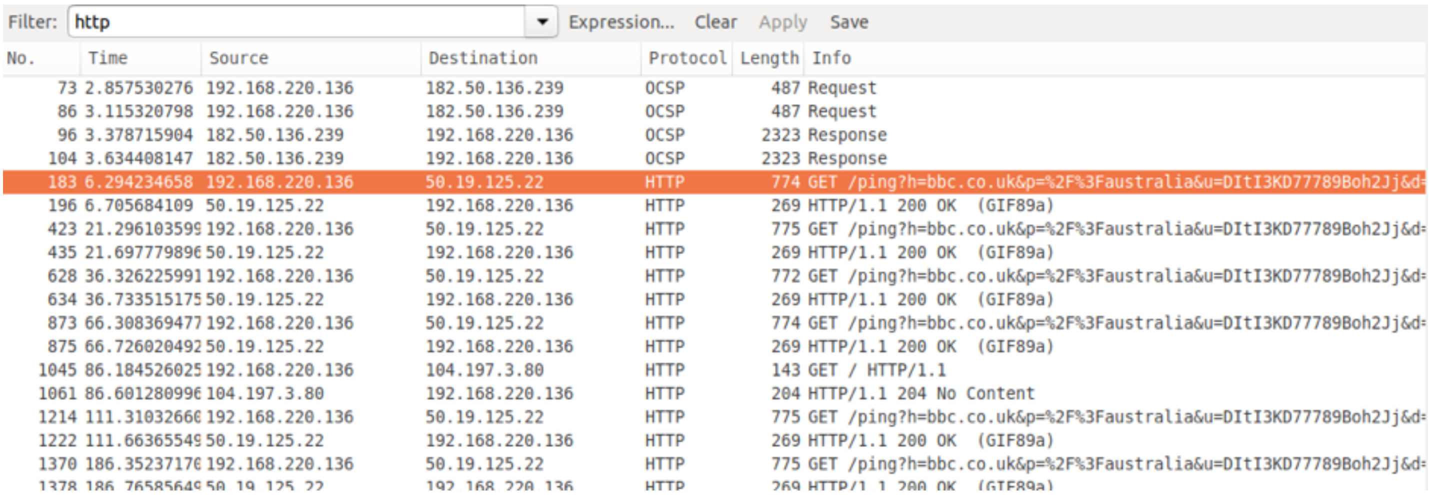
To select DNS traffic you would enter DNS into the filter bar and select Apply. You would see something like this.



```
No.  Time         Source           Destination      Protocol Length Info
  20 0.885703731  127.0.0.1        127.0.0.53       DNS        80 Standard query 0x47d6 A sync.crwdcntrl.net
  21 0.886109988  192.168.220.136  192.168.220.2    DNS        91 Standard query 0xb2ac A sync.crwdcntrl.net OPT
  22 0.886465410  127.0.0.1        127.0.0.53       DNS        80 Standard query 0x39de AAAA sync.crwdcntrl.net
  23 0.886872655  192.168.220.136  192.168.220.2    DNS        91 Standard query 0x74d0 AAAA sync.crwdcntrl.net OPT
  24 1.078151085  192.168.220.136  210.176.156.45   TCP        56 48370 → 443 [ACK] Seq=1 Ack=1 Win=45260 Len=0
  25 1.078298545  210.176.156.45   192.168.220.136  TCP        62 [TCP ACKed unseen segment] 443 → 48370 [ACK] Seq=1 Ack=2 Win=6
  26 1.146461871  192.168.220.2    192.168.220.136  DNS       262 Standard query response 0x74d0 AAAA sync.crwdcntrl.net CNAME t
  27 1.146807244  192.168.220.2    192.168.220.136  DNS       308 Standard query response 0xb2ac A sync.crwdcntrl.net CNAME td.c
  28 1.146905558  192.168.220.136  192.168.220.2    DNS       131 Standard query 0x333e AAAA nginx-bcp-stackA-21488747.ap-southe
```

This shows DNS queries for `sync.crwdcntrolnet` sent from my computer `192.168.220.136` to a DNS server (in this case I am running in a virtual machine so its sending it to my host computer). You will probably see something different.

You could now filter out the HTTP traffic by following the same approach used for DNS traffic. You would see something like this.



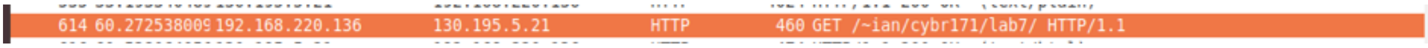
```
No.  Time         Source           Destination      Protocol Length Info
  73 2.857530276  192.168.220.136  182.50.136.239   OCSP      487 Request
  86 3.115320798  192.168.220.136  182.50.136.239   OCSP      487 Request
  96 3.378715904  182.50.136.239   192.168.220.136  OCSP     2323 Response
 104 3.634408147  182.50.136.239   192.168.220.136  OCSP     2323 Response
 183 6.294234658  192.168.220.136  50.19.125.22     HTTP      774 GET /ping?h=bbc.co.uk&p=%2F%3Faustralia&u=DItI3KD77789Boh2Jj&d=
 196 6.705684109  50.19.125.22     192.168.220.136  HTTP      269 HTTP/1.1 200 OK  (GIF89a)
 423 21.29610359  192.168.220.136  50.19.125.22     HTTP      775 GET /ping?h=bbc.co.uk&p=%2F%3Faustralia&u=DItI3KD77789Boh2Jj&d=
 435 21.69777989  50.19.125.22     192.168.220.136  HTTP      269 HTTP/1.1 200 OK  (GIF89a)
 628 36.32622599  192.168.220.136  50.19.125.22     HTTP      772 GET /ping?h=bbc.co.uk&p=%2F%3Faustralia&u=DItI3KD77789Boh2Jj&d=
 634 36.73351517  50.19.125.22     192.168.220.136  HTTP      269 HTTP/1.1 200 OK  (GIF89a)
 873 66.30836947  192.168.220.136  50.19.125.22     HTTP      774 GET /ping?h=bbc.co.uk&p=%2F%3Faustralia&u=DItI3KD77789Boh2Jj&d=
 875 66.72602049  50.19.125.22     192.168.220.136  HTTP      269 HTTP/1.1 200 OK  (GIF89a)
1045 86.18452602  192.168.220.136  104.197.3.80     HTTP      143 GET / HTTP/1.1
1061 86.60128099  104.197.3.80     192.168.220.136  HTTP      204 HTTP/1.1 204 No Content
1214 111.3103266  192.168.220.136  50.19.125.22     HTTP      775 GET /ping?h=bbc.co.uk&p=%2F%3Faustralia&u=DItI3KD77789Boh2Jj&d=
1222 111.6636554  50.19.125.22     192.168.220.136  HTTP      269 HTTP/1.1 200 OK  (GIF89a)
1370 186.3523717  192.168.220.136  50.19.125.22     HTTP      775 GET /ping?h=bbc.co.uk&p=%2F%3Faustralia&u=DItI3KD77789Boh2Jj&d=
1378 186.7658564  50.19.125.22     192.168.220.136  HTTP      269 HTTP/1.1 200 OK  (GIF89a)
```

Following a stream

Many of our protocols have requests and responses that are spread across many packets. What we want to do when analysing traffic is to put these back together to make it easier to see what is being sent and retrieved.

For example, to look at a request for a HTTP page and the reply.

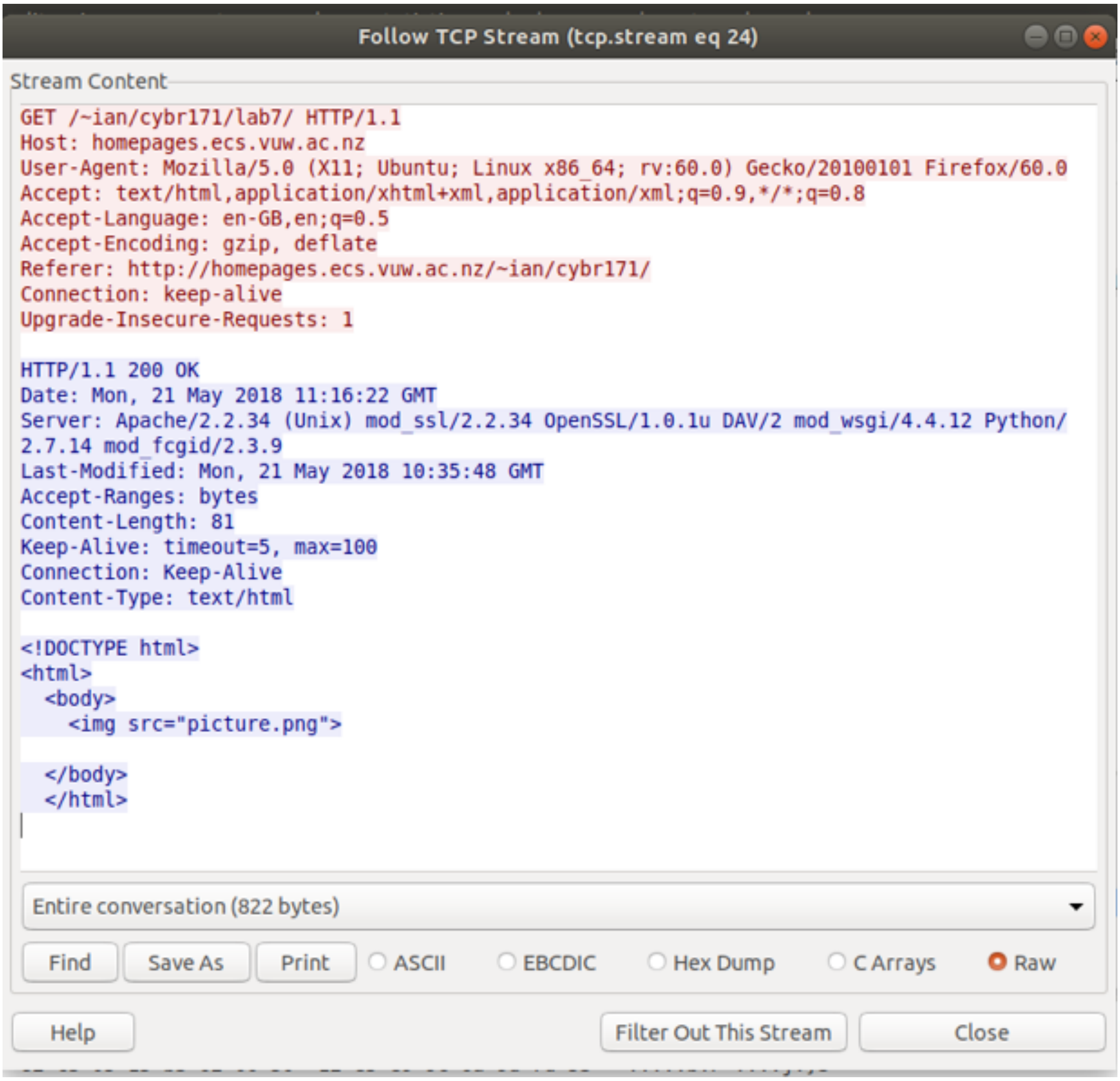
First you would use filtering to find all of the HTTP packets, and then locate a packet for interest. For example:



```
 614 60.27253800  192.168.220.136  130.195.5.21     HTTP      460 GET /~ian/cybr171/lab7/ HTTP/1.1
```

Assuming that we want to follow HTTP packets sent to a web server at `130.195.5.21` we would find a packet with this source address.

We then Right click on it and choose **Follow TCP stream**. You should see the full details of the request and response, such as:



Follow TCP Stream (tcp.stream eq 24)

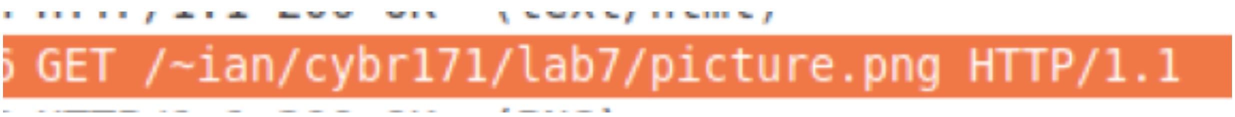
Stream Content

```
GET /~ian/cybr171/lab7/ HTTP/1.1
Host: homepages.ecs.vuw.ac.nz
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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
Referer: http://homepages.ecs.vuw.ac.nz/~ian/cybr171/
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Mon, 21 May 2018 11:16:22 GMT
Server: Apache/2.2.34 (Unix) mod_ssl/2.2.34 OpenSSL/1.0.1u DAV/2 mod_wsgi/4.4.12 Python/
2.7.14 mod_fcgid/2.3.9
Last-Modified: Mon, 21 May 2018 10:35:48 GMT
Accept-Ranges: bytes
Content-Length: 81
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html>
<html>
  <body>
    <img src="picture.png">

  </body>
  </html>
```

Entire conversation (822 bytes)

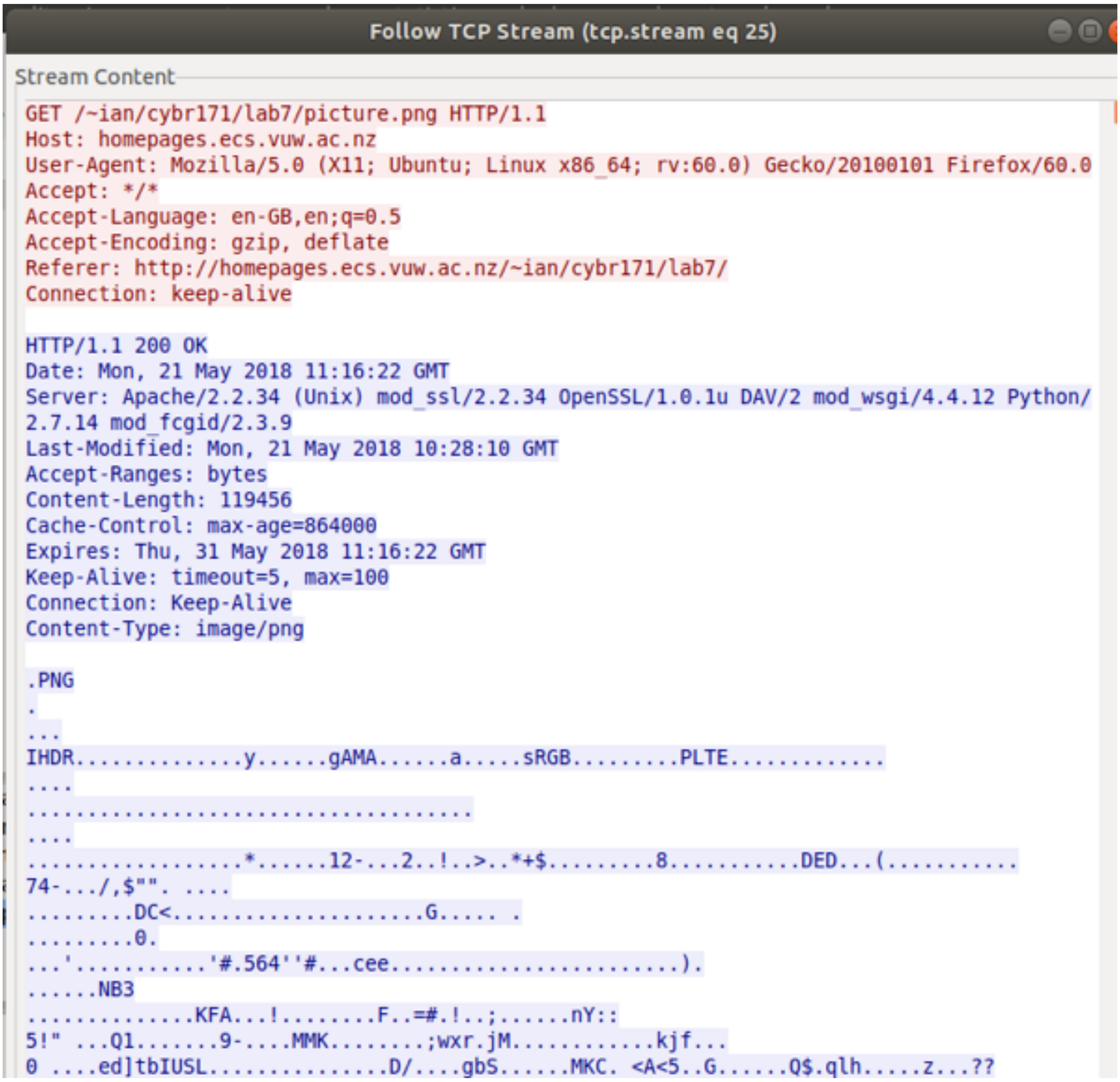
○ ASCII ○ EBCDIC ○ Hex Dump ○ C Arrays ● Raw

This shows that we retrieve the page using `GET /~ian/cybr171/lab7/` and the text in blue is the reply including the HTML for the webpage. You can also use the same approach to retrieve the image file `picture.png`. Find the request for the image file:



```
5 GET /~ian/cybr171/lab7/picture.png HTTP/1.1
```

You would then be able to use *Follow TCP stream* to show what is returned, for example:



```
GET /~ian/cybr171/lab7/picture.png HTTP/1.1
Host: homepages.ecs.vuw.ac.nz
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://homepages.ecs.vuw.ac.nz/~ian/cybr171/lab7/
Connection: keep-alive

HTTP/1.1 200 OK
Date: Mon, 21 May 2018 11:16:22 GMT
Server: Apache/2.2.34 (Unix) mod_ssl/2.2.34 OpenSSL/1.0.1u DAV/2 mod_wsgi/4.4.12 Python/
2.7.14 mod_fcgid/2.3.9
Last-Modified: Mon, 21 May 2018 10:28:10 GMT
Accept-Ranges: bytes
Content-Length: 119456
Cache-Control: max-age=864000
Expires: Thu, 31 May 2018 11:16:22 GMT
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: image/png

.PNG
.
...
IHDR.............y......gAMA......a.....sRGB.........PLTE.............
....
.....................................
....
..................*......12-...2..!..>..*+$.........8..........DED...(..........
74-.../,$"". ....
.........DC<.....................G..... .
.........0.
...'...........'#.564''#...cee........................).
......NB3
..............KFA...!........F..=#.!..;......nY::
5!" ...Q1.......9-....MMK........;wxr.jM............kjf...
0 ....ed]tbIUSL...............D/....gbS......MKC. <A<5..G......Q$.qlh.....z...??
```

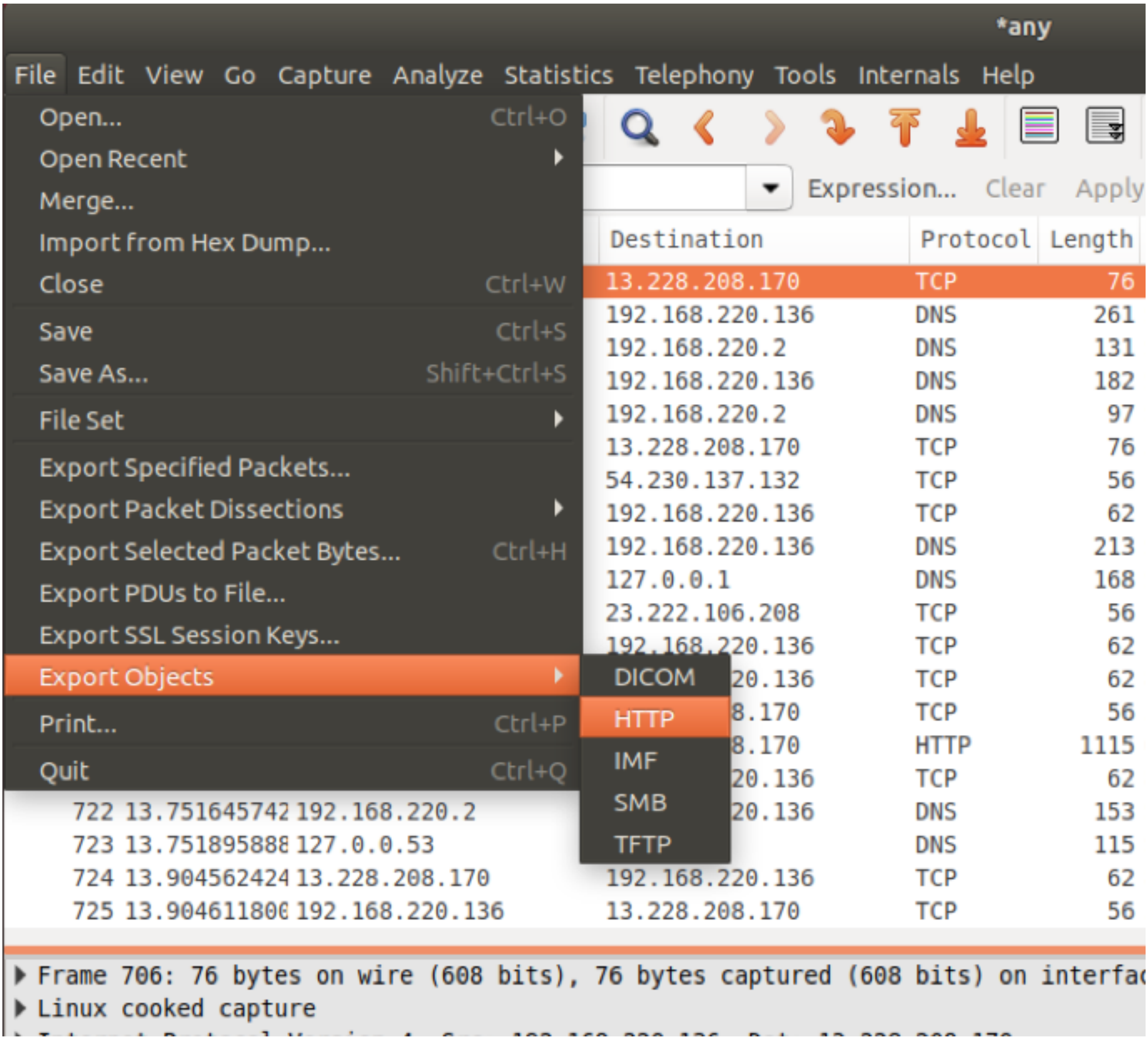
The binary representation of `picture.png` is shown in blue.

Extracting HTTP objects from a stream

What is we want to analyse what a person has been browsing, in particular extract all of the pictures that they have been viewing (perhaps to look for content violating the organisational policies).

Wireshark comes with a useful features for exporting packet data in different formats. This makes some forms of analysis very easy because Wireshark can extract the data and convert it into useful formats, in this case extract files sent and received via HTTP.

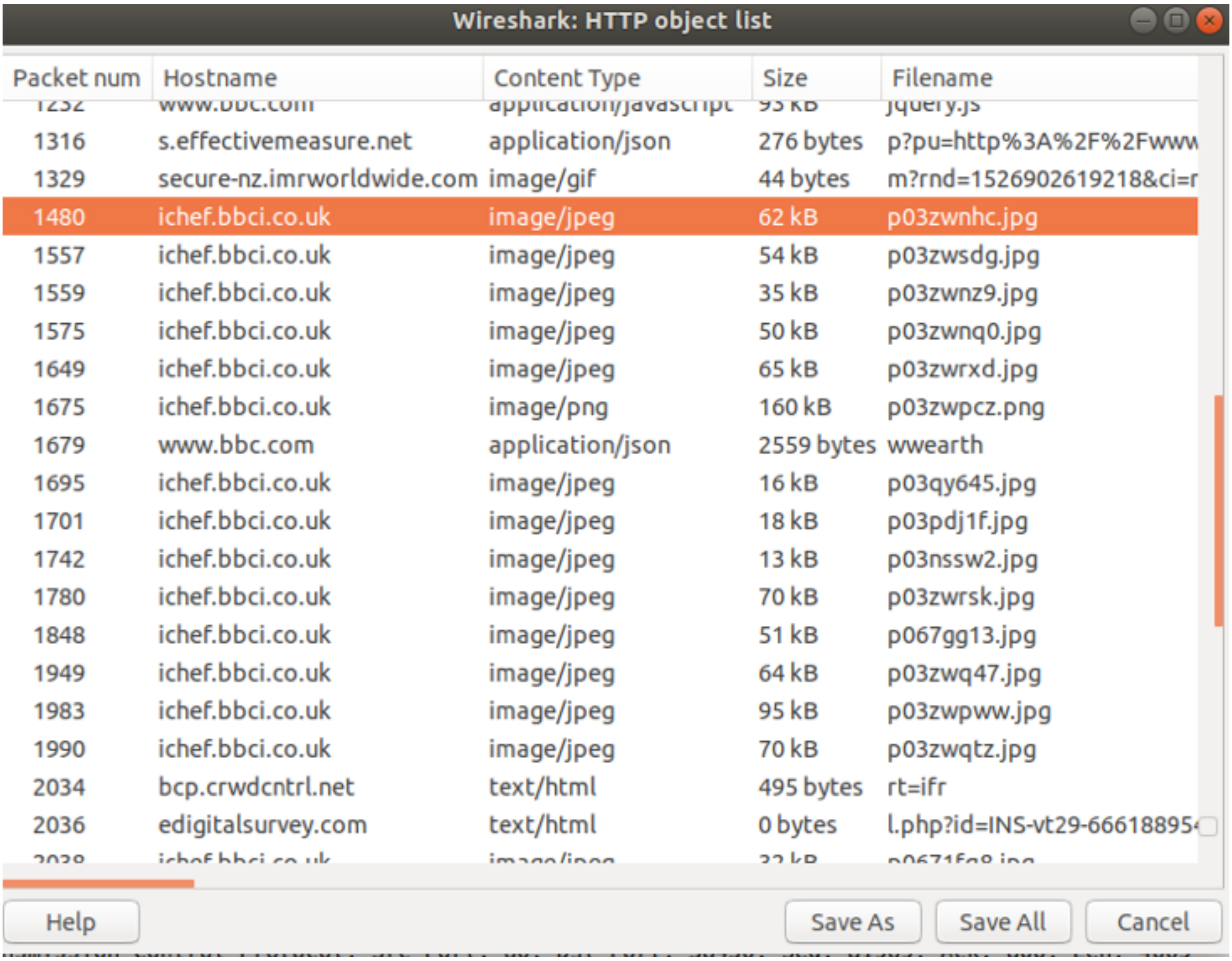
To do this choose File, go to Export Objects and choose HTTP.



```
                                    Destination          Protocol  Length
                                    13.228.208.170       TCP           76
                                    192.168.220.136      DNS          261
                                    192.168.220.2        DNS          131
                                    192.168.220.136      DNS          182
                                    192.168.220.2        DNS           97
                                    13.228.208.170       TCP           76
                                    54.230.137.132       TCP           56
                                    192.168.220.136      TCP           62
                                    192.168.220.136      DNS          213
                                    127.0.0.1            DNS          168
                                    23.222.106.208       TCP           56
                                    192.168.220.136      TCP           62
                                    20.136               TCP           62
                                    8.170                TCP           56
                                    8.170                HTTP        1115
                                    20.136               TCP           62
    722 13.751645742 192.168.220.2  20.136               DNS          153
    723 13.751895888 127.0.0.53                          DNS          115
    724 13.904562424 13.228.208.170 192.168.220.136      TCP           62
    725 13.904611806 192.168.220.136 13.228.208.170      TCP           56
```

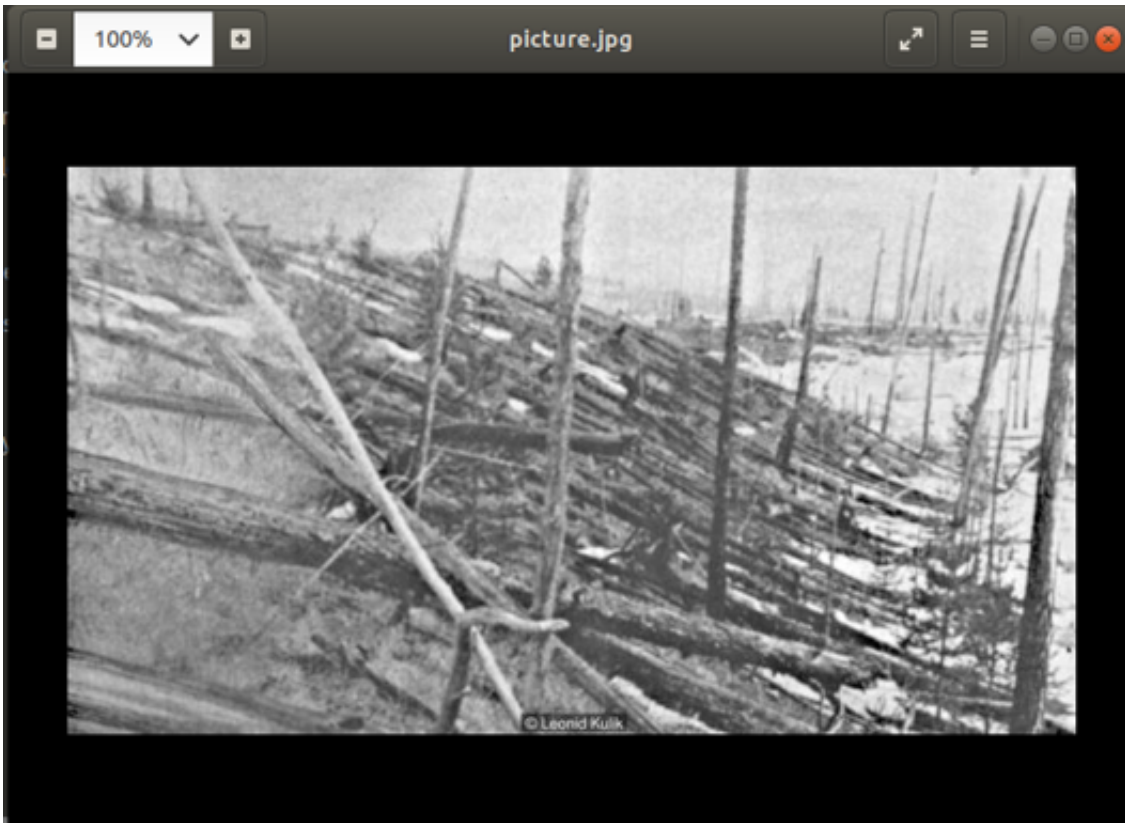
```
▶ Frame 706: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interfac
▶ Linux cooked capture
```

You should now see all of the HTTP objects (CSS, javascript, html and image files). You could save all of them off for later analysis but we will choose a single one.



Packet num	Hostname	Content Type	Size	Filename
1232	www.bbc.com	application/javascript	93 kB	jquery.js
1316	s.effectivemeasure.net	application/json	276 bytes	p?pu=http%3A%2F%2Fwww
1329	secure-nz.imrworldwide.com	image/gif	44 bytes	m?rnd=1526902619218&ci=r
1480	ichef.bbci.co.uk	image/jpeg	62 kB	p03zwnhc.jpg
1557	ichef.bbci.co.uk	image/jpeg	54 kB	p03zwsdg.jpg
1559	ichef.bbci.co.uk	image/jpeg	35 kB	p03zwnz9.jpg
1575	ichef.bbci.co.uk	image/jpeg	50 kB	p03zwnq0.jpg
1649	ichef.bbci.co.uk	image/jpeg	65 kB	p03zwrxd.jpg
1675	ichef.bbci.co.uk	image/png	160 kB	p03zwpcz.png
1679	www.bbc.com	application/json	2559 bytes	wwearth
1695	ichef.bbci.co.uk	image/jpeg	16 kB	p03qy645.jpg
1701	ichef.bbci.co.uk	image/jpeg	18 kB	p03pdj1f.jpg
1742	ichef.bbci.co.uk	image/jpeg	13 kB	p03nssw2.jpg
1780	ichef.bbci.co.uk	image/jpeg	70 kB	p03zwrsk.jpg
1848	ichef.bbci.co.uk	image/jpeg	51 kB	p067gg13.jpg
1949	ichef.bbci.co.uk	image/jpeg	64 kB	p03zwq47.jpg
1983	ichef.bbci.co.uk	image/jpeg	95 kB	p03zwpww.jpg
1990	ichef.bbci.co.uk	image/jpeg	70 kB	p03zwqtz.jpg
2034	bcp.crwdcntrl.net	text/html	495 bytes	rt=ifr
2036	edigitalsurvey.com	text/html	0 bytes	l.php?id=INS-vt29-66618895
2038	ichef.bbci.co.uk	image/jpeg	32 kB	p0671fq8.jpg

It is possbile to save this to the desktop as picture.jpg and open it using a viewing program such as a browser. For example:



Learning more

We have scratched the surface of Wireshark. You can install this at home and try this on your own computer, see https://www.wireshark.org/#download. Furthermore, there are also some excellent video tutorials at https://www.wireshark.org/docs/.

This Document's Licence (GPL)