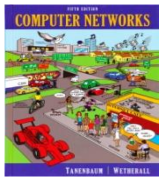
Lab Exercise – HTTP



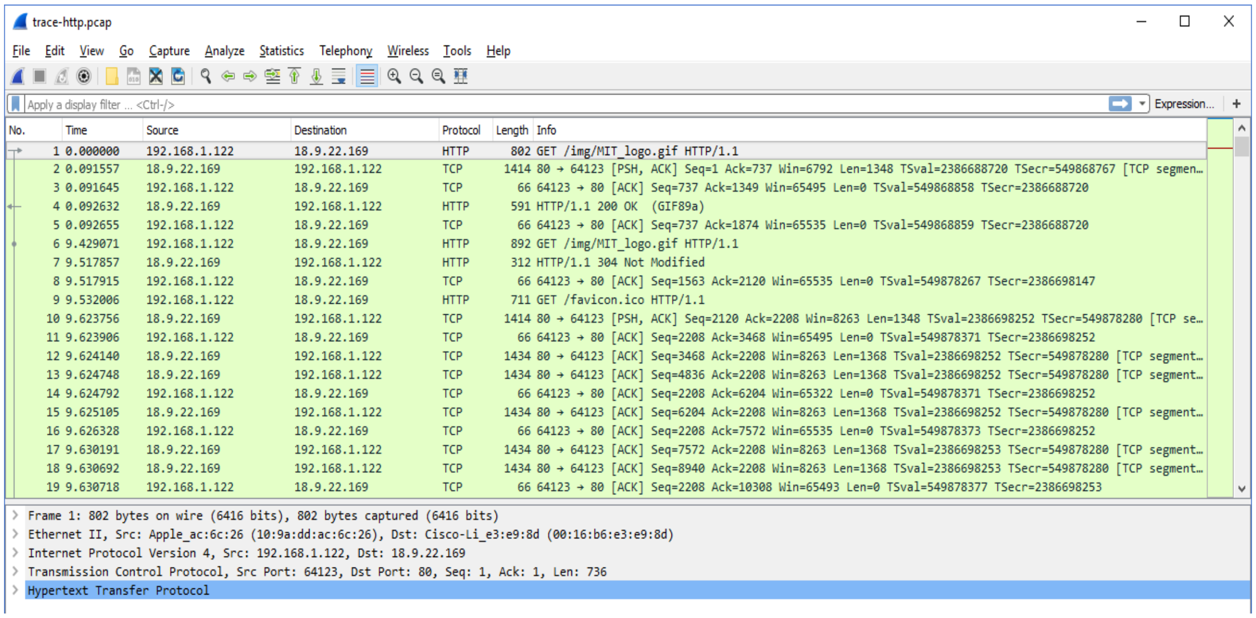
Objective

HTTP (HyperText Transfer Protocol) is the main protocol underlying the Web. HTTP functions as a request–response protocol in the client–server computing model. A web browser, for example, may be the client and an application running on a computer hosting a website may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body. A web browser is an example of a user agent (UA). Other types of user agent include the indexing software used by search providers (web crawlers), voice browsers, mobile apps, and other software that accesses, consumes, or displays web content. HTTP is designed to permit intermediate network elements to improve or enable communications between clients and servers. High-traffic websites often benefit from web cache servers that deliver content on behalf of upstream servers to improve response time. Web browsers cache previously accessed web resources and reuse them when possible to reduce network traffic. HTTP is an application layer protocol designed within the framework of the Internet protocol suite. Its definition presumes an underlying and reliable transport layer protocol and Transmission Control Protocol (TCP) is commonly used.

Step 1: Open the http Trace

Browser behavior can be quite complex, using more HTTP features than the basic exchange, this trace will show us how much gets transferred.

1. Open the trace file from here: https://kevincurran.org/com320/labs/wireshark/trace-http.pcap
   You should then see a Wireshark screen like below.


trace-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Time       Source          Destination     Protocol Length Info
   1 0.000000   192.168.1.122   18.9.22.169     HTTP      802 GET /img/MIT_logo.gif HTTP/1.1
   2 0.091557   18.9.22.169     192.168.1.122   TCP      1414 80 → 64123 [PSH, ACK] Seq=1 Ack=737 Win=6792 Len=1348 TSval=2386688720 TSecr=549868767 [TCP segmen…
   3 0.091645   192.168.1.122   18.9.22.169     TCP        66 64123 → 80 [ACK] Seq=737 Ack=1349 Win=65495 Len=0 TSval=549868858 TSecr=2386688720
   4 0.092632   18.9.22.169     192.168.1.122   HTTP      591 HTTP/1.1 200 OK  (GIF89a)
   5 0.092655   192.168.1.122   18.9.22.169     TCP        66 64123 → 80 [ACK] Seq=737 Ack=1874 Win=65535 Len=0 TSval=549868859 TSecr=2386688720
   6 9.429071   192.168.1.122   18.9.22.169     HTTP      892 GET /img/MIT_logo.gif HTTP/1.1
   7 9.517857   18.9.22.169     192.168.1.122   HTTP      312 HTTP/1.1 304 Not Modified
   8 9.517915   192.168.1.122   18.9.22.169     TCP        66 64123 → 80 [ACK] Seq=1563 Ack=2120 Win=65535 Len=0 TSval=549878267 TSecr=2386698147
   9 9.532006   192.168.1.122   18.9.22.169     HTTP      711 GET /favicon.ico HTTP/1.1
  10 9.623756   18.9.22.169     192.168.1.122   TCP      1414 80 → 64123 [PSH, ACK] Seq=2120 Ack=2208 Win=8263 Len=1348 TSval=2386698252 TSecr=549878280 [TCP se…
  11 9.623906   192.168.1.122   18.9.22.169     TCP        66 64123 → 80 [ACK] Seq=2208 Ack=3468 Win=65495 Len=0 TSval=549878371 TSecr=2386698252
  12 9.624140   18.9.22.169     192.168.1.122   TCP      1434 80 → 64123 [ACK] Seq=3468 Ack=2208 Win=8263 Len=1368 TSval=2386698252 TSecr=549878280 [TCP segment…
  13 9.624748   18.9.22.169     192.168.1.122   TCP      1434 80 → 64123 [ACK] Seq=4836 Ack=2208 Win=8263 Len=1368 TSval=2386698252 TSecr=549878280 [TCP segment…
  14 9.624792   192.168.1.122   18.9.22.169     TCP        66 64123 → 80 [ACK] Seq=2208 Ack=6204 Win=65322 Len=0 TSval=549878371 TSecr=2386698252
  15 9.625105   18.9.22.169     192.168.1.122   TCP      1434 80 → 64123 [ACK] Seq=6204 Ack=2208 Win=8263 Len=1368 TSval=2386698252 TSecr=549878280 [TCP segment…
  16 9.626328   192.168.1.122   18.9.22.169     TCP        66 64123 → 80 [ACK] Seq=2208 Ack=7572 Win=65535 Len=0 TSval=549878373 TSecr=2386698252
  17 9.630191   18.9.22.169     192.168.1.122   TCP      1434 80 → 64123 [ACK] Seq=7572 Ack=2208 Win=8263 Len=1368 TSval=2386698253 TSecr=549878280 [TCP segment…
  18 9.630692   18.9.22.169     192.168.1.122   TCP      1434 80 → 64123 [ACK] Seq=8940 Ack=2208 Win=8263 Len=1368 TSval=2386698253 TSecr=549878280 [TCP segment…
  19 9.630718   192.168.1.122   18.9.22.169     TCP        66 64123 → 80 [ACK] Seq=2208 Ack=10308 Win=65493 Len=0 TSval=549878377 TSecr=2386698253
```

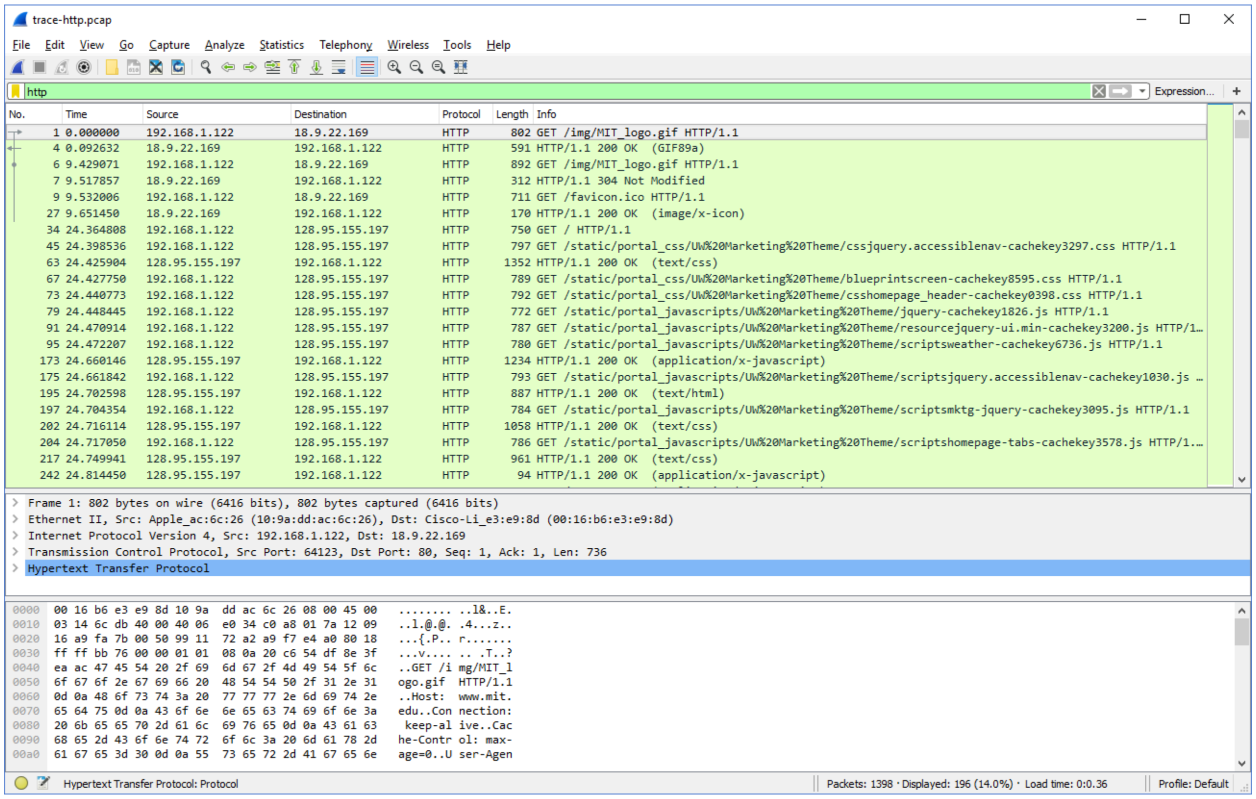
```
> Frame 1: 802 bytes on wire (6416 bits), 802 bytes captured (6416 bits)
> Ethernet II, Src: Apple_ac:6c:26 (10:9a:dd:ac:6c:26), Dst: Cisco-Li_e3:e9:8d (00:16:b6:e3:e9:8d)
> Internet Protocol Version 4, Src: 192.168.1.122, Dst: 18.9.22.169
> Transmission Control Protocol, Src Port: 64123, Dst Port: 80, Seq: 1, Ack: 1, Len: 736
> Hypertext Transfer Protocol
```

Figure 1: The Wireshark screen after opening the http trace file

Step 2: Inspect the Trace

2. To focus on HTTP traffic, *enter and apply a filter expression of* "`http`".

This filter will show HTTP requests and responses, but not the individual packets that are involved. You should see a screen like below.


trace-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
Expression...
Hypertext Transfer Protocol: Protocol
Packets: 1398 · Displayed: 196 (14.0%) · Load time: 0:0.36
Profile: Default

```
No.  Time        Source          Destination     Protocol Length Info
  1 0.000000    192.168.1.122   18.9.22.169     HTTP      802 GET /img/MIT_logo.gif HTTP/1.1
  4 0.092632    18.9.22.169     192.168.1.122   HTTP      591 HTTP/1.1 200 OK  (GIF89a)
  6 9.429071    192.168.1.122   18.9.22.169     HTTP      892 GET /img/MIT_logo.gif HTTP/1.1
  7 9.517857    18.9.22.169     192.168.1.122   HTTP      312 HTTP/1.1 304 Not Modified
  9 9.532006    192.168.1.122   18.9.22.169     HTTP      711 GET /favicon.ico HTTP/1.1
 27 9.651450    18.9.22.169     192.168.1.122   HTTP      170 HTTP/1.1 200 OK  (image/x-icon)
 34 24.364808   192.168.1.122   128.95.155.197  HTTP      750 GET / HTTP/1.1
 45 24.398536   192.168.1.122   128.95.155.197  HTTP      797 GET /static/portal_css/UW%20Marketing%20Theme/cssjquery.accessiblenav-cachekey3297.css HTTP/1.1
 63 24.425904   128.95.155.197  192.168.1.122   HTTP     1352 HTTP/1.1 200 OK  (text/css)
 67 24.427750   192.168.1.122   128.95.155.197  HTTP      789 GET /static/portal_css/UW%20Marketing%20Theme/blueprintscreen-cachekey8595.css HTTP/1.1
 73 24.440773   192.168.1.122   128.95.155.197  HTTP      792 GET /static/portal_css/UW%20Marketing%20Theme/csshomepage_header-cachekey0398.css HTTP/1.1
 79 24.448445   192.168.1.122   128.95.155.197  HTTP      772 GET /static/portal_javascripts/UW%20Marketing%20Theme/jquery-cachekey1826.js HTTP/1.1
 91 24.470914   192.168.1.122   128.95.155.197  HTTP      787 GET /static/portal_javascripts/UW%20Marketing%20Theme/resourcejquery-ui.min-cachekey3200.js HTTP/1…
 95 24.472207   192.168.1.122   128.95.155.197  HTTP      780 GET /static/portal_javascripts/UW%20Marketing%20Theme/scriptsweather-cachekey6736.js HTTP/1.1
173 24.660146   128.95.155.197  192.168.1.122   HTTP     1234 HTTP/1.1 200 OK  (application/x-javascript)
175 24.661842   192.168.1.122   128.95.155.197  HTTP      793 GET /static/portal_javascripts/UW%20Marketing%20Theme/scriptsjquery.accessiblenav-cachekey1030.js …
195 24.702598   128.95.155.197  192.168.1.122   HTTP      887 HTTP/1.1 200 OK  (text/html)
197 24.704354   192.168.1.122   128.95.155.197  HTTP      784 GET /static/portal_javascripts/UW%20Marketing%20Theme/scriptsmktg-jquery-cachekey3095.js HTTP/1.1
202 24.716114   128.95.155.197  192.168.1.122   HTTP     1058 HTTP/1.1 200 OK  (text/css)
204 24.717050   192.168.1.122   128.95.155.197  HTTP      786 GET /static/portal_javascripts/UW%20Marketing%20Theme/scriptshomepage-tabs-cachekey3578.js HTTP/1.…
217 24.749941   128.95.155.197  192.168.1.122   HTTP      961 HTTP/1.1 200 OK  (text/css)
242 24.814450   128.95.155.197  192.168.1.122   HTTP       94 HTTP/1.1 200 OK  (application/x-javascript)
```

```
> Frame 1: 802 bytes on wire (6416 bits), 802 bytes captured (6416 bits)
> Ethernet II, Src: Apple_ac:6c:26 (10:9a:dd:ac:6c:26), Dst: Cisco-Li_e3:e9:8d (00:16:b6:e3:e9:8d)
> Internet Protocol Version 4, Src: 192.168.1.122, Dst: 18.9.22.169
> Transmission Control Protocol, Src Port: 64123, Dst Port: 80, Seq: 1, Ack: 1, Len: 736
> Hypertext Transfer Protocol
```

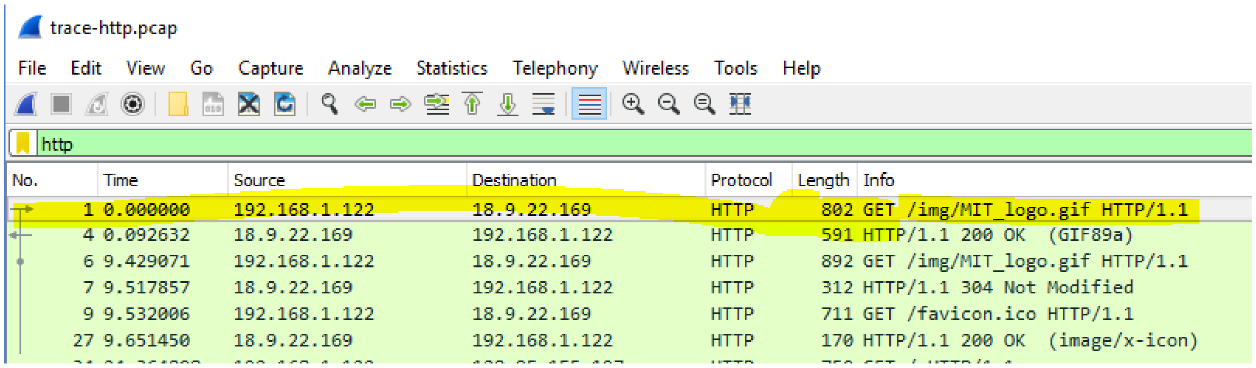
```
0000  00 16 b6 e3 e9 8d 10 9a  dd ac 6c 26 08 00 45 00   ........ ..l&..E.
0010  03 14 6c db 40 00 40 06  e0 34 c0 a8 01 7a 12 09   ..l.@.@. .4...z..
0020  16 a9 fa 7b 00 50 99 11  72 a2 a9 f7 e4 a0 80 18   ...{.P.. r.......
0030  ff ff bb 76 00 00 01 01  08 0a 20 c6 54 df 8e 3f   ...v.... .. .T..?
0040  ea ac 47 45 54 20 2f 69  6d 67 2f 4d 49 54 5f 6c   ..GET /i mg/MIT_l
0050  6f 67 6f 2e 67 69 66 20  48 54 54 50 2f 31 2e 31   ogo.gif  HTTP/1.1
0060  0d 0a 48 6f 73 74 3a 20  77 77 77 2e 6d 69 74 2e   ..Host:  www.mit.
0070  65 64 75 0d 0a 43 6f 6e  6e 65 63 74 69 6f 6e 3a   edu..Con nection:
0080  20 6b 65 65 70 2d 61 6c  69 76 65 0d 0a 43 61 63    keep-al ive..Cac
0090  68 65 2d 43 6f 6e 74 72  6f 6c 3a 20 6d 61 78 2d   he-Contr ol: max-
00a0  61 67 65 3d 30 0d 0a 55  73 65 72 2d 41 67 65 6e   age=0..U ser-Agen
```

Figure 2: Trace of HTTP traffic

Recall that an HTTP response carrying content will normally be spread across multiple packets. When the last packet in the response arrives, Wireshark assembles the complete response and tags the packet with protocol HTTP. The earlier packets are simply TCP segments carrying data; the last packet tagged HTTP includes a list of all the earlier packets used to make the response. A similar process occurs for the request, but in this case it is common for a request to fit in a single packet. With the filter expression of "http" we will hide the intermediate TCP packets and see only the HTTP requests and responses.

3. Select the first GET in the trace and expand its HTTP block.

(This will be 802 GET /img/MIT_logo.gif HTTP/1.1 as shown below).

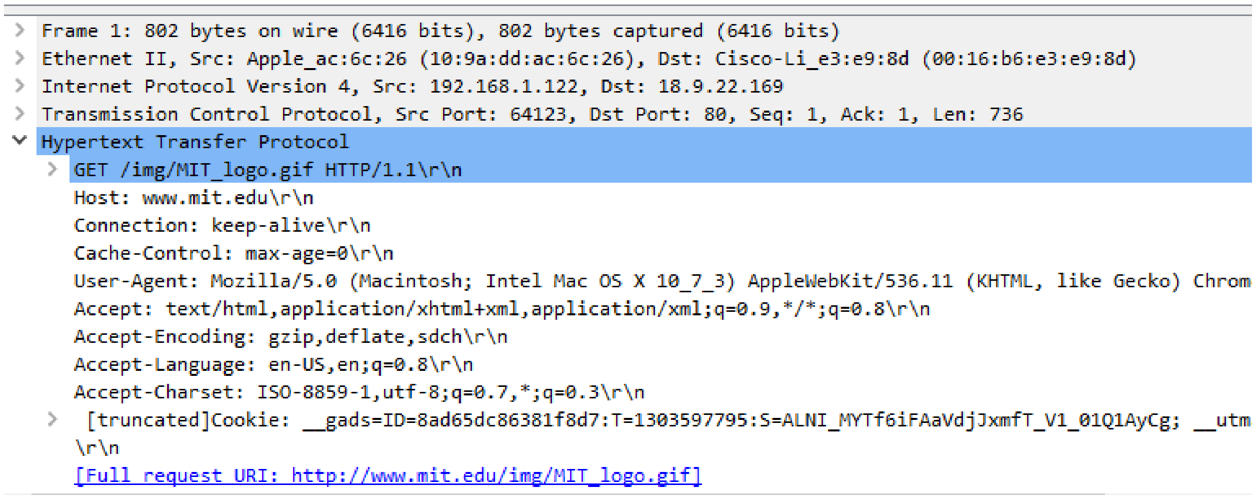

trace-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

```
No.   Time       Source          Destination     Protocol  Length Info
    1 0.000000   192.168.1.122   18.9.22.169     HTTP      802 GET /img/MIT_logo.gif HTTP/1.1
    4 0.092632   18.9.22.169     192.168.1.122   HTTP      591 HTTP/1.1 200 OK  (GIF89a)
    6 9.429071   192.168.1.122   18.9.22.169     HTTP      892 GET /img/MIT_logo.gif HTTP/1.1
    7 9.517857   18.9.22.169     192.168.1.122   HTTP      312 HTTP/1.1 304 Not Modified
    9 9.532006   192.168.1.122   18.9.22.169     HTTP      711 GET /favicon.ico HTTP/1.1
   27 9.651450   18.9.22.169     192.168.1.122   HTTP      170 HTTP/1.1 200 OK  (image/x-icon)
```

Figure 3: The first GET packet in trace: get /img/MIT_logo.gif HTTP/1.1

This will let us inspect the details of an HTTP request.

4. Inspect the HTTP header by expanding the down arrow beside “Hypertext Transfer Protocol” In the middle pane (as shown below).



```
> Frame 1: 802 bytes on wire (6416 bits), 802 bytes captured (6416 bits)
> Ethernet II, Src: Apple_ac:6c:26 (10:9a:dd:ac:6c:26), Dst: Cisco-Li_e3:e9:8d (00:16:b6:e3:e9:8d)
> Internet Protocol Version 4, Src: 192.168.1.122, Dst: 18.9.22.169
> Transmission Control Protocol, Src Port: 64123, Dst Port: 80, Seq: 1, Ack: 1, Len: 736
v Hypertext Transfer Protocol
  > GET /img/MIT_logo.gif HTTP/1.1\r\n
    Host: www.mit.edu\r\n
    Connection: keep-alive\r\n
    Cache-Control: max-age=0\r\n
    User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/536.11 (KHTML, like Gecko) Chrom
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Encoding: gzip,deflate,sdch\r\n
    Accept-Language: en-US,en;q=0.8\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3\r\n
  >  [truncated]Cookie: __gads=ID=8ad65dc86381f8d7:T=1303597795:S=ALNI_MYTf6iFAaVdjJxmfT_V1_01Q1AyCg; __utm
    \r\n
    [Full request URI: http://www.mit.edu/img/MIT_logo.gif]
```

Figure 4: The Hypertext transfer protocol for the first http request

Observe that the HTTP header follows the TCP and IP headers, as HTTP is an application protocol that is transported using TCP/IP. To view it, select the packet, find the HTTP block in the middle panel, and expand it (by using the “v” expander or icon). This block is expanded in figure above.

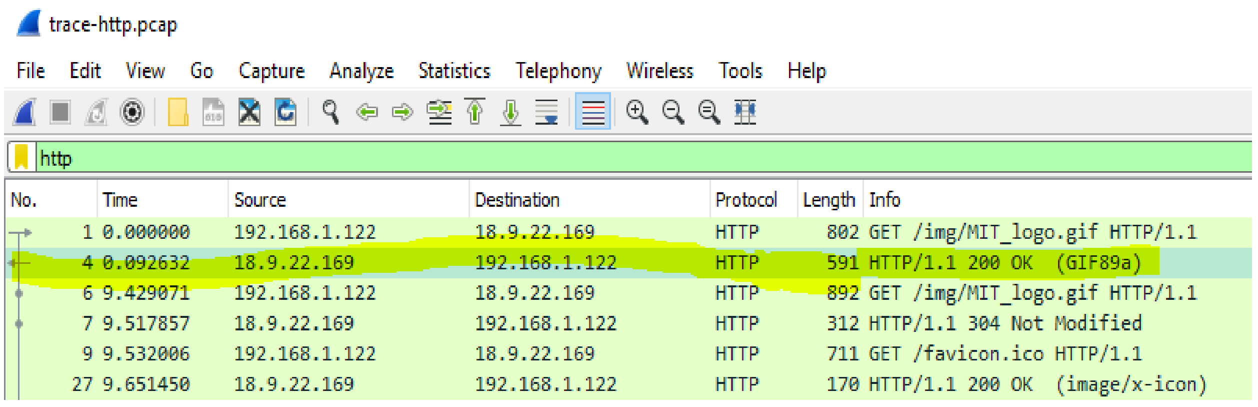
Explore the headers that are sent along with the request. First, you will see the GET method at the start of the request, including details such as the path. Then you will see a series of headers in the form of tagged parameters. There may be many headers, and the choice of headers and their values vary from browser to browser. See if you have any of these common headers:

- Host. A mandatory header, it identifies the name (and port) of the server.

- User-Agent. The kind of browser and its capabilities.
- Accept, Accept-Encoding, Accept-Charset, Accept-Language. Descriptions of the formats that will be accepted in the response, e.g., text/html, including its encoding, e.g., gzip, and language.
- Cookie. The name and value of cookies the browser holds for the website.
- Cache-Control. Information about how the response can be cached.

The request information is sent in a simple text and line-based format. If you look in the bottom panel you can read much of the request directly from the packet itself.

5. *Select the response that corresponds to the first GET in the trace and expand its HTTP block* (see below).


trace-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.122	18.9.22.169	HTTP	802	GET /img/MIT_logo.gif HTTP/1.1
4	0.092632	18.9.22.169	192.168.1.122	HTTP	591	HTTP/1.1 200 OK (GIF89a)
6	9.429071	192.168.1.122	18.9.22.169	HTTP	892	GET /img/MIT_logo.gif HTTP/1.1
7	9.517857	18.9.22.169	192.168.1.122	HTTP	312	HTTP/1.1 304 Not Modified
9	9.532006	192.168.1.122	18.9.22.169	HTTP	711	GET /favicon.ico HTTP/1.1
27	9.651450	18.9.22.169	192.168.1.122	HTTP	170	HTTP/1.1 200 OK (image/x-icon)

Figure 5: The response that corresponds to the first GET in the trace

The Info for this packet will indicate “200 OK” in the case of a normal, successful transfer. You will see that the response is similar to the request, with a series of headers that follow the “200 OK” status code. However, different headers will be used, and the headers will be followed by the requested content.

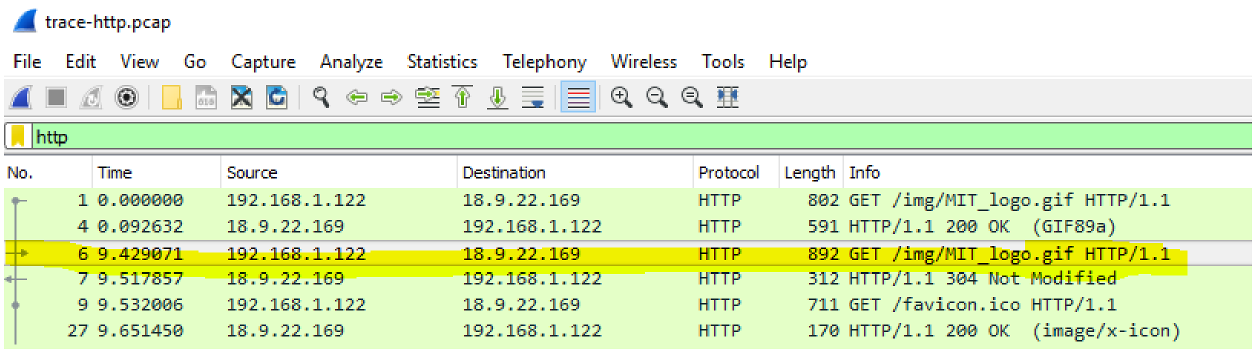
Examine the common headers such as:

- Server. The kind of server and its capabilities.
- Date, Last-Modified. The time of the response and the time the content last changed.
- Cache-Control, Expires, Etag. Information about how the response can be cached.

Step 3: Content Caching

The second fetch in the trace is a re-fetch of the first URL. This fetch presents an opportunity for us to look at caching in action, since it is highly likely that the image or document has not changed and therefore does not need to be downloaded again. HTTP caching mechanisms should identify this opportunity. We will now see how they work.

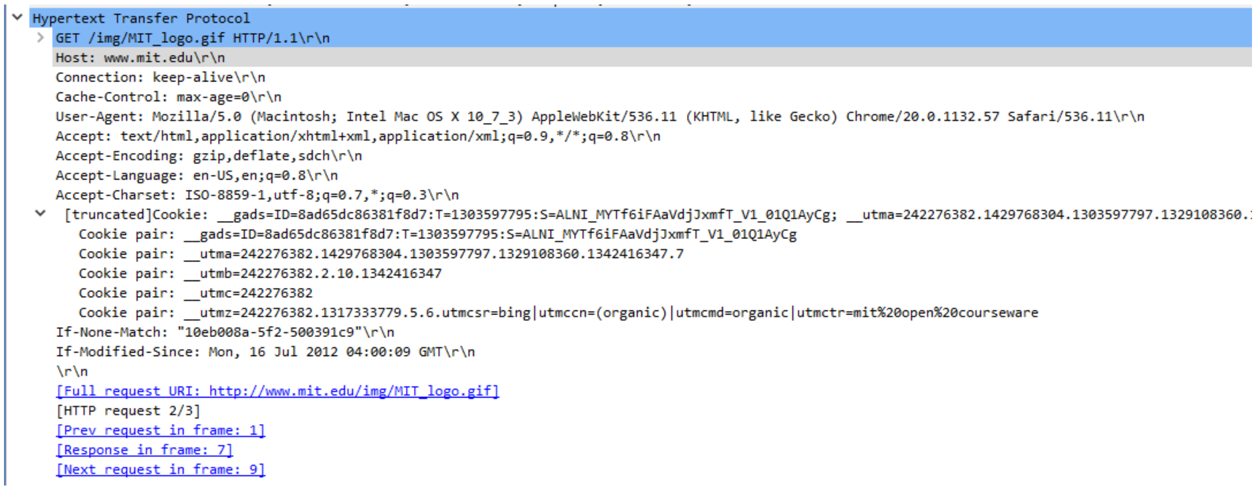
6. *Click on the third line* in the sample trace i.e. 892 GET /img/MIT_logo.gif HTTP 1.1


trace-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

```
No.   Time       Source          Destination     Protocol  Length  Info
    1 0.000000   192.168.1.122   18.9.22.169     HTTP         802  GET /img/MIT_logo.gif HTTP/1.1
    4 0.092632   18.9.22.169     192.168.1.122   HTTP         591  HTTP/1.1 200 OK  (GIF89a)
    6 9.429071   192.168.1.122   18.9.22.169     HTTP         892  GET /img/MIT_logo.gif HTTP/1.1
    7 9.517857   18.9.22.169     192.168.1.122   HTTP         312  HTTP/1.1 304 Not Modified
    9 9.532006   192.168.1.122   18.9.22.169     HTTP         711  GET /favicon.ico HTTP/1.1
   27 9.651450   18.9.22.169     192.168.1.122   HTTP         170  HTTP/1.1 200 OK  (image/x-icon)
```

Figure 6: The second fetch - 892 GET /img/MIT_logo.gif HTTP 1.1

7. *Expand its HTTP block (see below)*

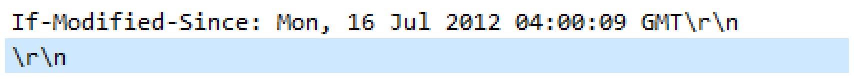


```
Hypertext Transfer Protocol
  GET /img/MIT_logo.gif HTTP/1.1\r\n
  Host: www.mit.edu\r\n
  Connection: keep-alive\r\n
  Cache-Control: max-age=0\r\n
  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
  Accept-Encoding: gzip,deflate,sdch\r\n
  Accept-Language: en-US,en;q=0.8\r\n
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3\r\n
   [truncated]Cookie: __gads=ID=8ad65dc86381f8d7:T=1303597795:S=ALNI_MYTf6iFAaVdjJxmfT_V1_01Q1AyCg; __utma=242276382.1429768304.1303597797.1329108360.
    Cookie pair: __gads=ID=8ad65dc86381f8d7:T=1303597795:S=ALNI_MYTf6iFAaVdjJxmfT_V1_01Q1AyCg
    Cookie pair: __utma=242276382.1429768304.1303597797.1329108360.1342416347.7
    Cookie pair: __utmb=242276382.2.10.1342416347
    Cookie pair: __utmc=242276382
    Cookie pair: __utmz=242276382.1317333779.5.6.utmcsr=bing|utmccn=(organic)|utmcmd=organic|utmctr=mit%20open%20courseware
  If-None-Match: "10eb008a-5f2-500391c9"\r\n
  If-Modified-Since: Mon, 16 Jul 2012 04:00:09 GMT\r\n
  \r\n
  [Full request URI: http://www.mit.edu/img/MIT_logo.gif]
  [HTTP request 2/3]
  [Prev request in frame: 1]
  [Response in frame: 7]
  [Next request in frame: 9]
```

Figure 7: The HTTP block of 892 GET /img/MIT_logo.gif HTTP 1.1

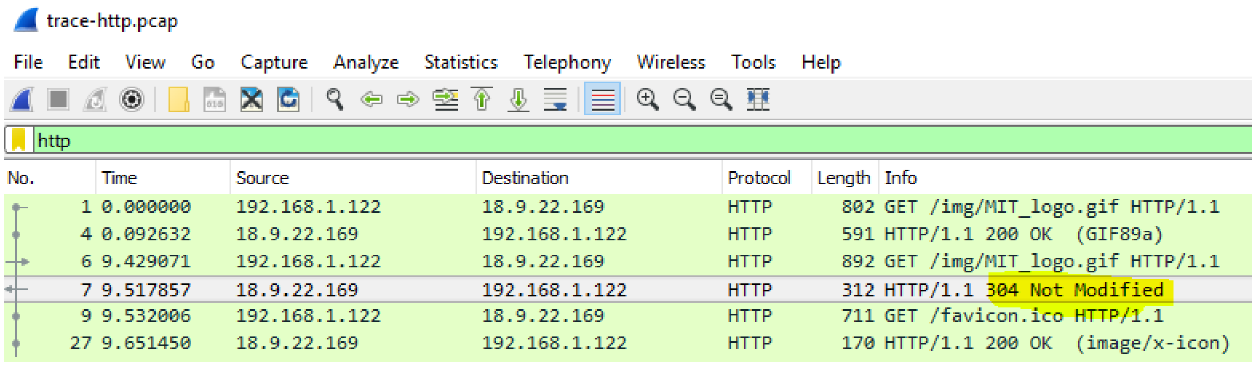
Now find the header that will let the server work out whether it needs to send fresh content. The server will need to send fresh content only if the content has changed since the browser last downloaded it. To work this out, the browser includes a timestamp taken from the previous download for the content that it has cached. This header was not present on the first GET since we cleared the browser cache so the browser had no previous download of the content that it could use. In most other respects, this request will be the same as the first-time request.

This header is called "If-Modified-Since", i.e., it asks the server to send the content if it has been modified since a given time (see the header below).



```
If-Modified-Since: Mon, 16 Jul 2012 04:00:09 GMT\r\n
\r\n
```

8. *Now select the response to the re-fetch, and expand its HTTP block.*


trace-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.122	18.9.22.169	HTTP	802	GET /img/MIT_logo.gif HTTP/1.1
4	0.092632	18.9.22.169	192.168.1.122	HTTP	591	HTTP/1.1 200 OK (GIF89a)
6	9.429071	192.168.1.122	18.9.22.169	HTTP	892	GET /img/MIT_logo.gif HTTP/1.1
7	9.517857	18.9.22.169	192.168.1.122	HTTP	312	HTTP/1.1 304 Not Modified
9	9.532006	192.168.1.122	18.9.22.169	HTTP	711	GET /favicon.ico HTTP/1.1
27	9.651450	18.9.22.169	192.168.1.122	HTTP	170	HTTP/1.1 200 OK (image/x-icon)

Figure 8: The response to cached image

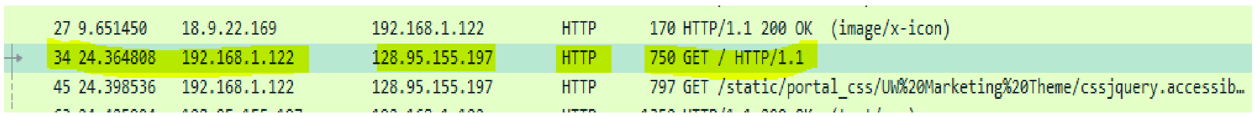
We can see that caching worked as expected, this response will not contain the content. Instead, the status code of the response is "304 Not Modified". This tells the browser that the content is unchanged from its previous copy, and the cached content can then be displayed.

Note the timestamp value comes from the "Last-Modified" header of the most recent download of the content. It is a server timestamp for when the content last changed – it is not a timestamp according to the browser clock, and it is not a timestamp of the time of the download

Step 4: Complex Pages

Now we will examine the fourth fetch in the trace. This fetch was for a more complex web page that will likely have embedded resources. So the browser will download the initial HTML plus all of the embedded resources needed to render the page, plus other resources that are requested during the execution of page scripts. As we will see, a single page can involve many GETs.

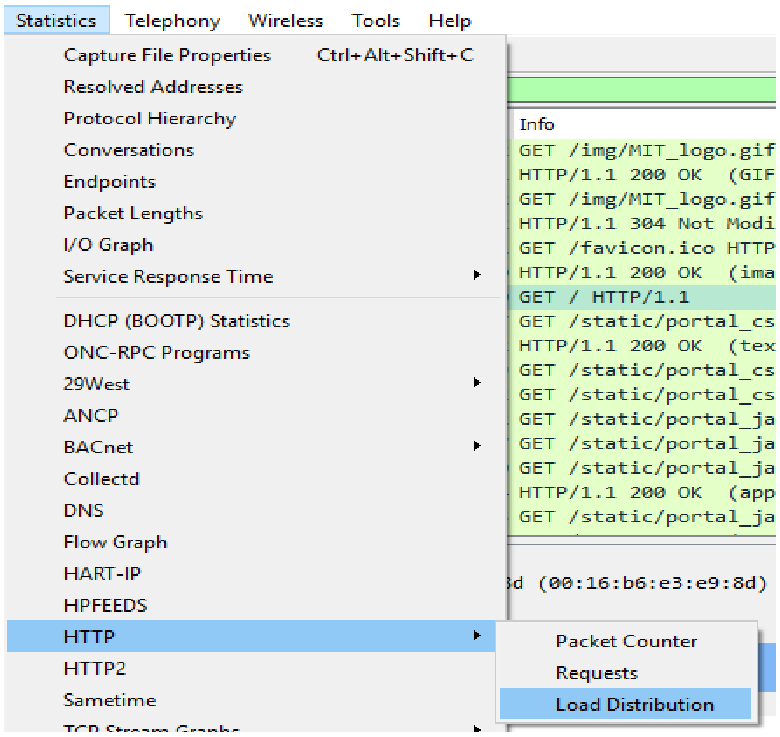
9. Click on no 34 which is 750 GET / HTTP/1.1 (as shown below).



```
27 9.651450   18.9.22.169     192.168.1.122    HTTP    170 HTTP/1.1 200 OK  (image/x-icon)
34 24.364808  192.168.1.122   128.95.155.197   HTTP    750 GET / HTTP/1.1
45 24.398536  192.168.1.122   128.95.155.197   HTTP    797 GET /static/portal_css/UW%20Marketing%20Theme/cssjquery.accessib…
```

Figure 8: The large fourth fetch in http trace

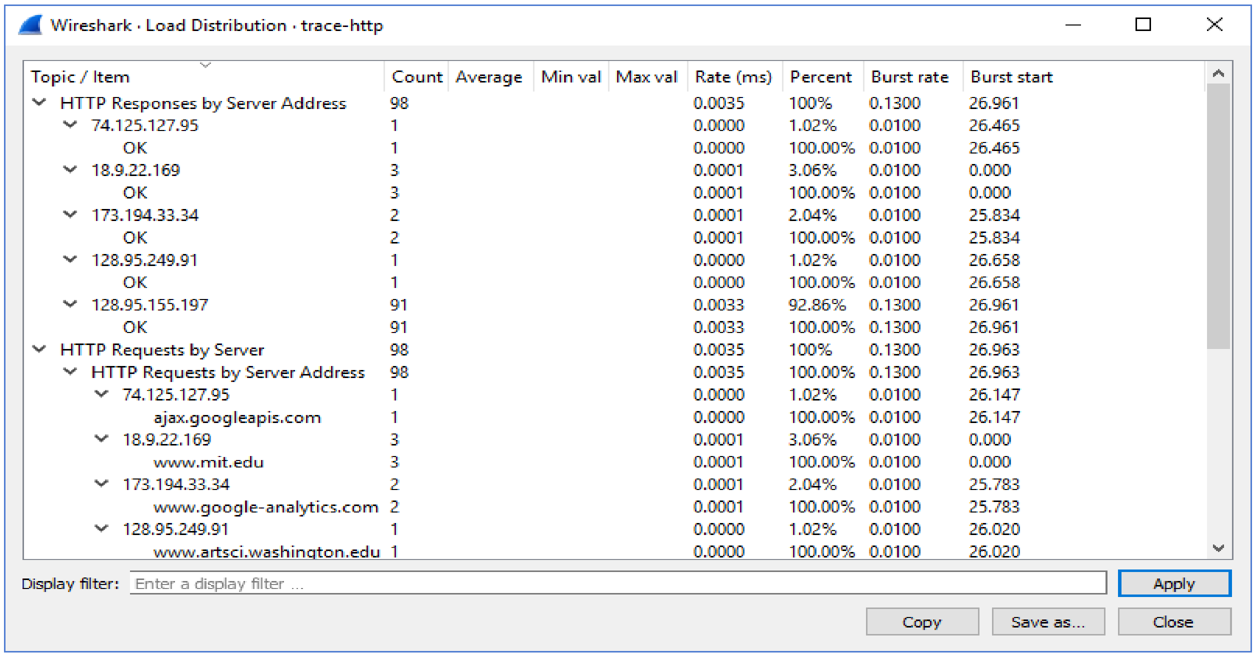
10. *Summarize the GETs for this page by bringing up a HTTP Load Distribution panel.* You will find this panel under "Statistics" and "HTTP".



```
Info
GET /img/MIT_logo.gif
HTTP/1.1 200 OK  (GIF
GET /img/MIT_logo.gif
HTTP/1.1 304 Not Modi
GET /favicon.ico HTTP
HTTP/1.1 200 OK  (ima
GET / HTTP/1.1
GET /static/portal_cs
HTTP/1.1 200 OK  (tex
GET /static/portal_cs
GET /static/portal_cs
GET /static/portal_ja
GET /static/portal_ja
GET /static/portal_ja
HTTP/1.1 200 OK  (app
GET /static/portal_ja
3d (00:16:b6:e3:e9:8d)
```

Figure 9: Statistics → HTTP → Load Distribution

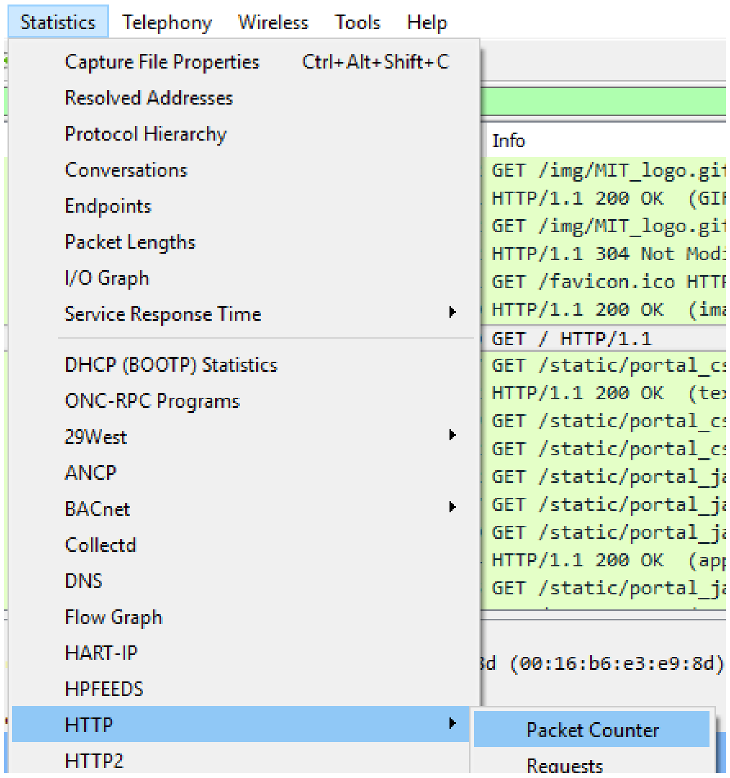
Looking at this panel will tell you how many requests were made to which servers. This is typical, when surfing, chances are that your fetch will request content from other servers you might not have suspected to build the page. These other servers may include third parties such as content distribution networks, ad networks, and analytics networks. Our panel is shown below – the page fetch involved 98 requests to 5 different servers.



Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
HTTP Responses by Server Address	98				0.0035	100%	0.1300	26.961
74.125.127.95	1				0.0000	1.02%	0.0100	26.465
OK	1				0.0000	100.00%	0.0100	26.465
18.9.22.169	3				0.0001	3.06%	0.0100	0.000
OK	3				0.0001	100.00%	0.0100	0.000
173.194.33.34	2				0.0001	2.04%	0.0100	25.834
OK	2				0.0001	100.00%	0.0100	25.834
128.95.249.91	1				0.0000	1.02%	0.0100	26.658
OK	1				0.0000	100.00%	0.0100	26.658
128.95.155.197	91				0.0033	92.86%	0.1300	26.961
OK	91				0.0033	100.00%	0.1300	26.961
HTTP Requests by Server	98				0.0035	100%	0.1300	26.963
HTTP Requests by Server Address	98				0.0035	100.00%	0.1300	26.963
74.125.127.95	1				0.0000	1.02%	0.0100	26.147
ajax.googleapis.com	1				0.0000	100.00%	0.0100	26.147
18.9.22.169	3				0.0001	3.06%	0.0100	0.000
www.mit.edu	3				0.0001	100.00%	0.0100	0.000
173.194.33.34	2				0.0001	2.04%	0.0100	25.783
www.google-analytics.com	2				0.0001	100.00%	0.0100	25.783
128.95.249.91	1				0.0000	1.02%	0.0100	26.020
www.artsci.washington.edu	1				0.0000	100.00%	0.0100	26.020

Figure 10: HTTP Load Distribution panel

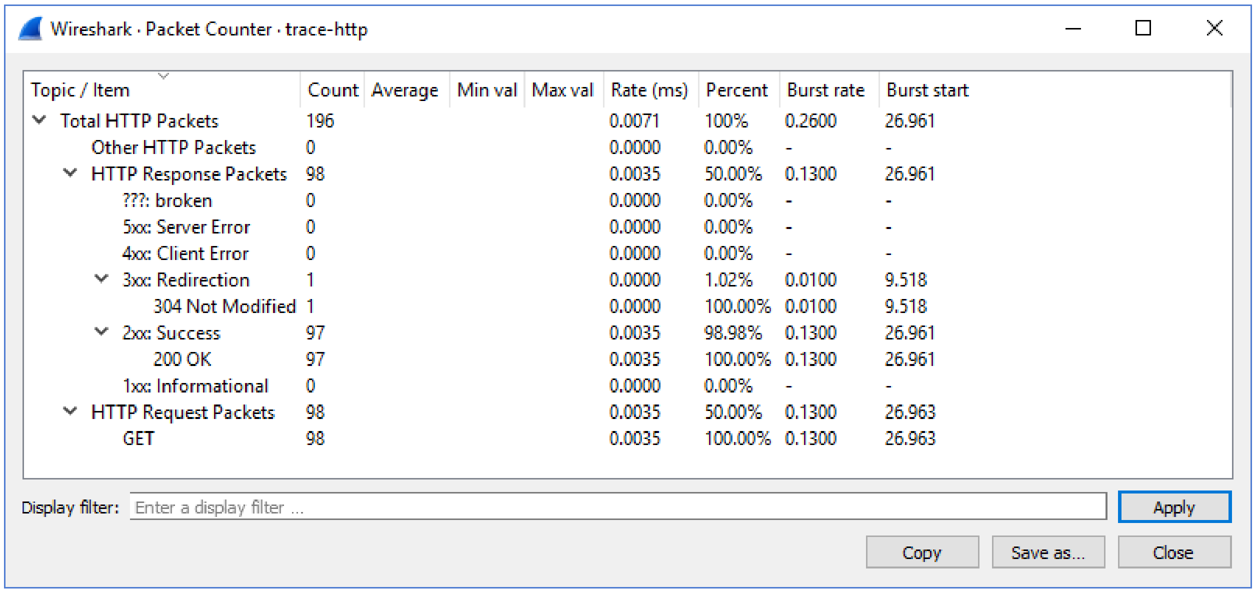
11. For a different kind of summary of the GETs, *bring up a HTTP Packet Counter panel.*



```
GET /img/MIT_logo.gi
HTTP/1.1 200 OK  (GI
GET /img/MIT_logo.gi
HTTP/1.1 304 Not Mod
GET /favicon.ico HTT
HTTP/1.1 200 OK  (ima
GET / HTTP/1.1
GET /static/portal_c
HTTP/1.1 200 OK  (te
GET /static/portal_c
GET /static/portal_c
GET /static/portal_j
GET /static/portal_j
GET /static/portal_j
HTTP/1.1 200 OK  (ap
GET /static/portal_j
d (00:16:b6:e3:e9:8d)
```

Figure 11: Statistics → HTTP → Packet Counter

You will also find this panel under “Statistics” and “HTTP”, and you should filter for the packets that are part of the fetch as before. This panel will tell you the kinds of request and responses. Our panel is shown in the figure below. You can see that it consists of GET requests that are matched by 200 OK responses and 304 Not Modified. However, there are a variety of other response codes that you might observe in your trace, such as when the resource is already cached.


Wireshark · Packet Counter · trace-http
Display filter:
Enter a display filter ...
Apply
Copy
Save as...
Close

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Total HTTP Packets	196				0.0071	100%	0.2600	26.961
Other HTTP Packets	0				0.0000	0.00%	-	-
HTTP Response Packets	98				0.0035	50.00%	0.1300	26.961
???: broken	0				0.0000	0.00%	-	-
5xx: Server Error	0				0.0000	0.00%	-	-
4xx: Client Error	0				0.0000	0.00%	-	-
3xx: Redirection	1				0.0000	1.02%	0.0100	9.518
304 Not Modified	1				0.0000	100.00%	0.0100	9.518
2xx: Success	97				0.0035	98.98%	0.1300	26.961
200 OK	97				0.0035	100.00%	0.1300	26.961
1xx: Informational	0				0.0000	0.00%	-	-
HTTP Request Packets	98				0.0035	50.00%	0.1300	26.963
GET	98				0.0035	100.00%	0.1300	26.963

Figure 12: HTTP Packet Counter panel

You might be curious to know what content is being downloaded by all these requests. As well as seeing the URLs in the Info column, you can get a summary of the URLs in a HTTP Request panel under “Statistics” and “HTTP”. Each of the individual requests and responses has the same form we saw in an earlier step. Collectively, they are performed in the process of fetching a complete page with a given URL.

For a more detailed look at the overall page load process, use a site such as Google’s PageSpeed or https://www.webpagetest.org/. These sites will test a URL of your choice and generate a report of the page load activity, telling what requests were fetched at what times and giving tips for decreasing the overall page load time.

The beginning of the “waterfall” diagram for the page load corresponding to a trace on www.ulster.ac.uk is shown in the next figure. After the initial HTML resource is fetched there are many subsequent quick fetches for embedded resources such as JavaScript scripts, CSS stylesheets, images, and more.

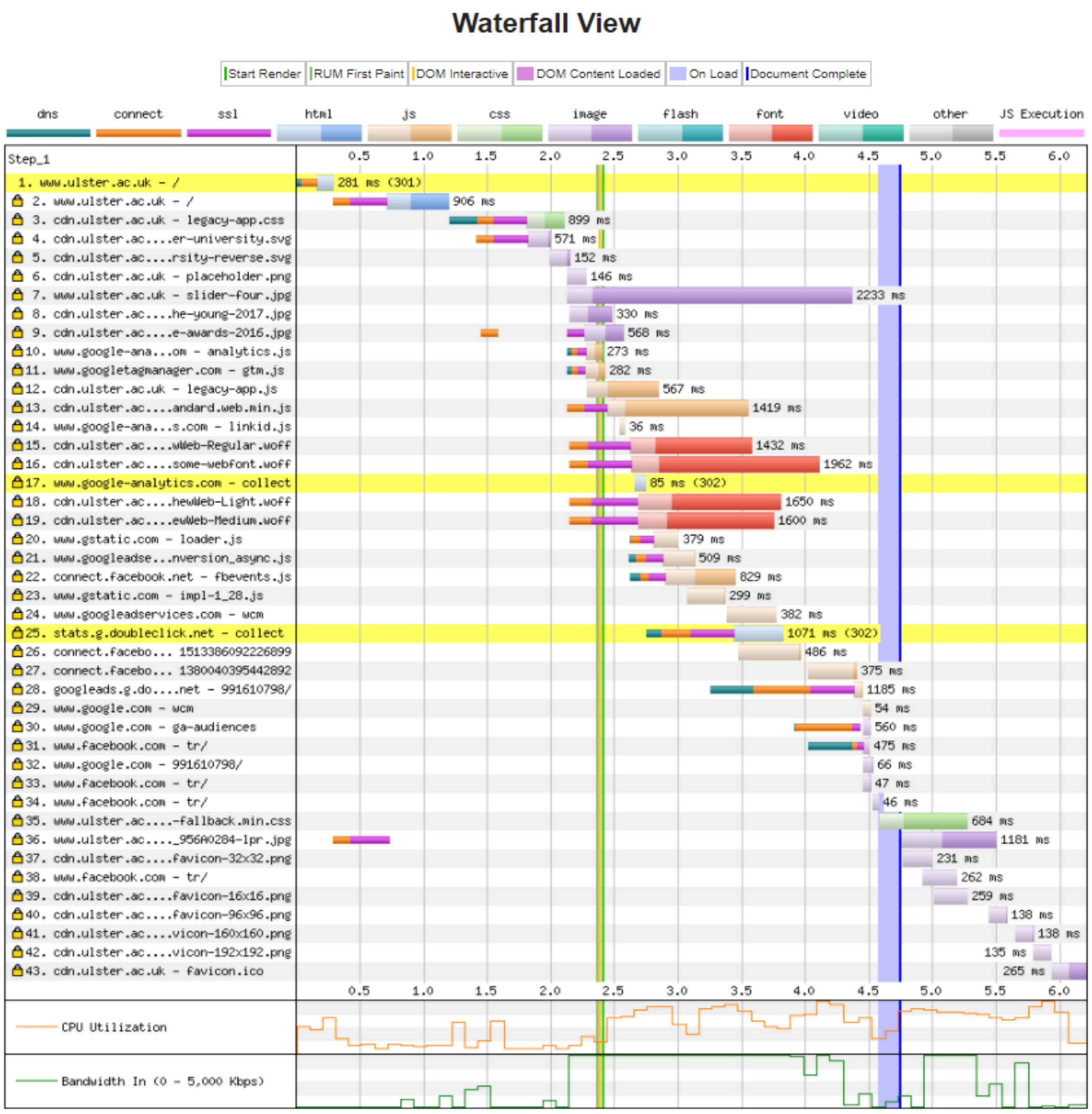
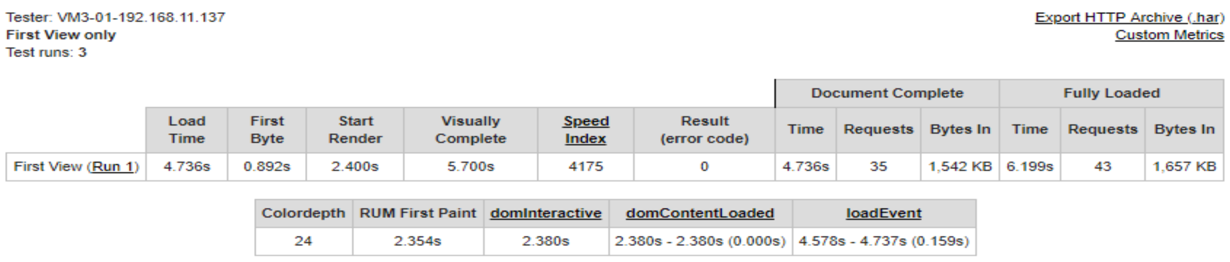


Figure 13: Start of waterfall graph for https://www.ulster.ac.uk/ (from pageloadtest.org)



Tester: VM3-01-192.168.11.137
First View only
Test runs: 3

Export HTTP Archive (.har)
Custom Metrics

	Load Time	First Byte	Start Render	Visually Complete	Speed Index	Result (error code)	Document Complete Time	Document Complete Requests	Document Complete Bytes In	Fully Loaded Time	Fully Loaded Requests	Fully Loaded Bytes In
First View (Run 1)	4.736s	0.892s	2.400s	5.700s	4175	0	4.736s	35	1,542 KB	6.199s	43	1,657 KB

Colordepth	RUM First Paint	domInteractive	domContentLoaded	loadEvent
24	2.354s	2.380s	2.380s - 2.380s (0.000s)	4.578s - 4.737s (0.159s)

Figure 14: Summary stats for https://www.ulster.ac.uk/ (from pageloadtest.org)