The Basic HTTP GET/response interaction

1: HTTP/1.1. from Packet details window or Packet listing window.

2.a: sv, en-GB & en-US
2.b: What other information does the browser supply:
That it has no preference in terms of language.
What browsers it supports.
Connection: keep-alive
What markup languages it accepts (html, XML, etc.)
from Packet details window

3: user IP-address: 130.236.76.90
server IP-address: 128.119.245.12
from Packet listing window

4: 200 OK. from Packet listing window

5: Wed, 18 Jan 2017, 06:59:02 GMT. from Packet details window

6: 293 bytes. from Packet details window

7: Yes, every entry that is in brackets is supplied by wireshark, and not part of the actual packets. (examples: "[HTTP response 1/1]" and "[Time since request: 0.106]".)

The HTTP CONDITIONAL GET/response interaction

8: No, since it's the first time loading the server contents

9: Packet-listing window displays "200 OK (text/html)" which is the file. You can also look in Packet-details window to further inspect the returned contents.

10: Yes, "Mon, 23 Jan 2017 06:59:01 GMT".

11: 304 Not modified. It did not return any contents. This is because the contents are already cached on the computer.

When the web browser doesn't have any contents cached it will request the contents from the server and the server will return it. The received contents are being cached by the web browser/stored on the hard drive. If the cached content are up to date with the server's content, that means the server will not return any files. Instead the server returns the status code 304 Not Modified.

Retrieving Long Documents

12: One GET request was sent

13: Two (amount of bytes varies) TCP segments when we did our own trace (but the alternative trace-files supplied as a part of the course had four TCP segments).

14: Status code: 200. Response phrase: OK

15: The first TCP segment's data contains all of the HTTP-headers.

The web browser sent one GET request. Since the GET-request data according to the TCP-protocol is too big. It separated the data into two packets of 1448 and 3413 bytes each, equaling in a total of 4861 bytes. Since the web browser's cache was cleared the web server sends the requested contents. This means that the response phrase from the server was "OK" and the status code was "200". In the very first TCP segment you can find the HTTP headers, and once the file has reassembled it will look to HTTP as a normal file sent in one go.

HTML Documents with Embedded Objects

16: Four GET-requests were sent by the web browser.
HTML-file was requested from "http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html"
.PNG-file was requested from "http://gaia.cs.umass.edu/pearson.png"
.JPG-file was requested twice from first "http://manic.cs.umass.edu/~kurose/cover_5th_ed.jpg" and then "http://caite.cs.umass.edu/~kurose/cover_5th_ed.jpg".

17: They were downloaded in parallel. If it was serially it would wait with requesting the second image until it had received the first one.

When we did our own trace as per the instructions, we got the above results. The two GET requests for the same image comes from a server-side redirect, this caused a bit of confusion but was ultimately educational. The returned data for the first GET request is in (text/HTML)* while data corresponding to the second GET request is (JPEG JFIF image).
As for the two images, the easiest way to realize that they are in parallel is to notice that they are two connections to two different servers. Had they been on the same server, they would have been sent and received serially.

* Essentially: " The document has moved here. "

HTTP Authentication

18: Status code: 401, Status phrase: Unauthorized/Authorization required

19: An "Authorization"-field containing "credentials"-data.
this authorization sequence is not unlike a second "handshake".
C: "Requesting file"
S: "who are you?"
C: "these are my credentials"
S: "you may now request the file"
C: "Requesting file"... et.c.

Preparation questions for Assignment 2

20: That the connection is "keep-alive" means that the server will be open for more requests from that IP-address for an X amount of time. Connection "closed" means that another "handshake" is needed to open a new connection. "Keep-alive" is preferable the times where a client requests many files, since the "handshake" takes unnecessary time. If the client only needs one file it's beneficial for the server to close the connection as quickly as possible to avoid queuing and decrease the pressure on the server.
Interesting information: The connection "closed" were used in HTTP/1.0 but were eventually changed to connection "keep-alive" in HTTP/1.1.