

Lab 2

Exercise 3

[Question 1](#)

[Question 2](#)

[Question 3](#)

[Question 4](#)

[Question 5](#)

Exercise 4

[Question 1](#)

[Question 2](#)

[Question 3](#)

[Question 4](#)

[Question 5](#)

Exercise 5

Exercise 3

Question 1

What is the status code and phrase returned from the server to the client browser?

10	4.694850	192.168.1.102	128.119.245.12	HTTP	555 GET /ethereal-labs/lab2-1.html HTTP/1.
12	4.718993	128.119.245.12	192.168.1.102	HTTP	439 HTTP/1.1 200 OK (text/html)
13	4.724332	192.168.1.102	128.119.245.12	HTTP	541 GET /favicon.ico HTTP/1.1
14	4.750366	128.119.245.12	192.168.1.102	HTTP	1395 HTTP/1.1 404 Not Found (text/html)

200 OK

Question 2

When was the HTML file that the browser is retrieving last modified at the server?

2003-09-23, 05:29:00

```

▼ Hypertext Transfer Protocol
  ▶ HTTP/1.1 200 OK\r\n
    Date: Tue, 23 Sep 2003 05:29:50 GMT\r\n
    Server: Apache/2.0.40 (Red Hat Linux)\r\n
    Last-Modified: Tue, 23 Sep 2003 05:29:00 GMT\r\n
    ETag: "1bfed-49-79d5bf00"\r\n
    Accept-Ranges: bytes\r\n
    ▶ Content-Length: 73\r\n

```

Does the response also contain a DATE header?

Yes

How are these two fields different?

According to mozilla.org, last modified is when the resource was last modified, whereas date is when the message originated

“The **last-modified** response HTTP header contains a date and time when the origin server believes the resource was last modified. It is used as a validator to determine if the resource is the same as the previously stored one

The **date** general HTTP header contains the date and time at which the message was originated.” ~ mozilla.org

Question 3

Is the connection established between the browser and the server persistent or non-persistent? How can you infer this?

Persistent as the connection is a keep-alive connection.

```

▼ Hypertext Transfer Protocol
  ▶ GET /ethereal-labs/lab2-1.html HTTP/1.1\r\n
    Host: gaia.cs.umass.edu\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.0.2) Gecko/20060208 Firefox/1.0.2\r\n
    Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,application/javascript;q=0.7\r\n
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate, compress;q=0.9\r\n
    Accept-Charset: ISO-8859-1, utf-8;q=0.66, *;q=0.66\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
\r\n
[Full request URI: http://gaia.cs.umass.edu/ethereal-labs/lab2-1.html]
[HTTP request 1/2]
[Response in frame: 12]
[Next request in frame: 13]

```

Question 4

How many bytes of content are being returned to the browser?

73 bytes

```
▼ Hypertext Transfer Protocol
  ▶ HTTP/1.1 200 OK\r\n
    Date: Tue, 23 Sep 2003 05:29:50 GMT\r\n
    Server: Apache/2.0.40 (Red Hat Linux)\r\n
    Last-Modified: Tue, 23 Sep 2003 05:29:00 GMT\r\n
    ETag: "1bfed-49-79d5bf00"\r\n
    Accept-Ranges: bytes\r\n
    ▶ Content-Length: 73\r\n
    Keep-Alive: timeout=10, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=ISO-8859-1\r\n
    \r\n
    [HTTP response 1/2]
    [Time since request: 0.024143000 seconds]
    \[Request in frame: 10\]
    \[Next request in frame: 13\]
    \[Next response in frame: 14\]
    [Request URI: http://gaia.cs.umass.edu/ethereal-labs/lab2-1.html]
    File Data: 73 bytes
  ▶ Line-based text data: text/html (3 lines)
```

Question 5

What is the data contained inside the HTTP response packet?

Text/HTML, as below

```
▼ Line-based text data: text/html (3 lines)
  <html>\n
  Congratulations.  You've downloaded the file lab2-1.html!\n
  </html>\n
```

Exercise 4

Question 1

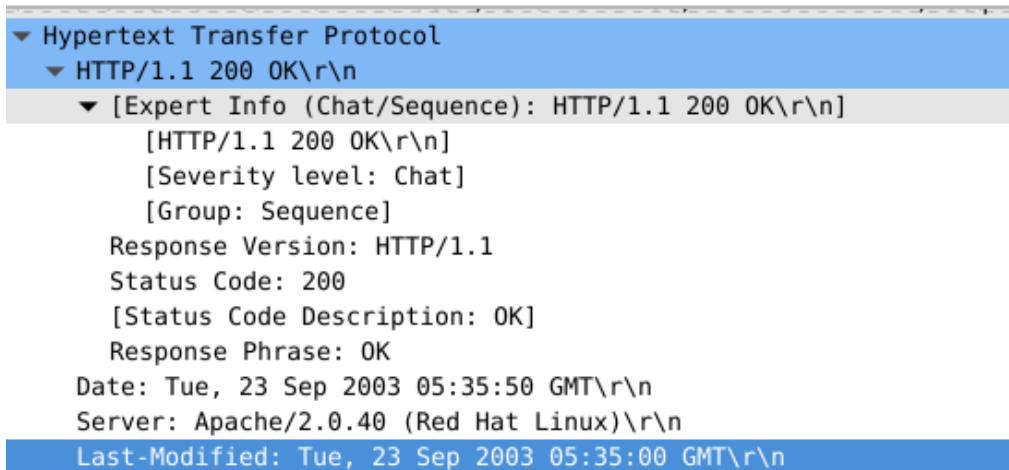
Inspect the contents of the first HTTP GET request from the browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET?

No

Question 2

Does the response indicate the last time that the requested file was modified?

Yes



```
▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 200 OK\r\n
    ▼ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      [HTTP/1.1 200 OK\r\n]
      [Severity level: Chat]
      [Group: Sequence]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
      Date: Tue, 23 Sep 2003 05:35:50 GMT\r\n
      Server: Apache/2.0.40 (Red Hat Linux)\r\n
      Last-Modified: Tue, 23 Sep 2003 05:35:00 GMT\r\n
```

Question 3

Now inspect the contents of the second HTTP GET request from the browser to the server. Do you see an “IF-MODIFIED-SINCE:” and “IF-NONE-MATCH” lines in the HTTP GET? If so, what information is contained in these header lines?

Yes

```

HyperText Transfer Protocol
  GET /ethereal-labs/lab2-2.html HTTP/1.1\r\n
  [Expert Info (Chat/Sequence): GET /ethereal-labs/lab2-2.html HTTP/1.1\r\n]
    [GET /ethereal-labs/lab2-2.html HTTP/1.1\r\n]
    [Severity level: Chat]
    [Group: Sequence]
    Request Method: GET
    Request URI: /ethereal-labs/lab2-2.html
    Request Version: HTTP/1.1
    Host: gaia.cs.umass.edu\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.0.2) Gecko/20021120 Netscape/7.01\r\n
    Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-mng,ir
    Accept-Language: en-us, en;q=0.50\r\n
    Accept-Encoding: gzip, deflate, compress;q=0.9\r\n
    Accept-Charset: ISO-8859-1, utf-8;q=0.66, *;q=0.66\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    If-Modified-Since: Tue, 23 Sep 2003 05:35:00 GMT\r\n
    If-None-Match: "1bfef-173-8f4ae900"\r\n

```

Question 4

What is the HTTP status code and phrase returned from the server in response to this second HTTP GET?

304 Not Modified

Did the server explicitly return the contents of the file? Explain.

A 304 response got send back without any body.

Question 5

What is the value of the Etag field in the 2nd response message and how it is used?

An entity tag is an identifier of a specific version of a resource. It is useful for caching as they can be though of as a “fingerprint” of a version of the resource. If the resource has been updated or modified, the Etag/fingerprint would change. This is useful to work out if the cache is up to date or not.

Etag: "1bfef-173-8f4ae900"

```

▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 304 Not Modified\r\n
    ▼ [Expert Info (Chat/Sequence): HTTP/1.1 304 Not Modified\r\n]
      [HTTP/1.1 304 Not Modified\r\n]
      [Severity level: Chat]
      [Group: Sequence]
      Response Version: HTTP/1.1
      Status Code: 304
      [Status Code Description: Not Modified]
      Response Phrase: Not Modified
      Date: Tue, 23 Sep 2003 05:35:53 GMT\r\n
      Server: Apache/2.0.40 (Red Hat Linux)\r\n
      Connection: Keep-Alive\r\n
      Keep-Alive: timeout=10, max=99\r\n
      ETag: "1bfef-173-8f4ae900"\r\n

```

Has this value changed since the 1st response message was received?

No, the ETag value is still ETag: "1bfef-173-8f4ae900"

Exercise 5

Screenshot of output on client side

```

z3330164@vx5:/tmp_ amd/ravel/export/ravel/2/z3330164/COMP9331/COMP9331/lab2$ python3
PingClient.py 127.0.0.1 5000
ping to 127.0.0.1, seq = 1, rtt = 114
ping to 127.0.0.1, seq = 2, rtt = 82
ping to 127.0.0.1, seq = 3, rtt = 103
ping to 127.0.0.1, seq = 4, rtt = 161
ping to 127.0.0.1, seq = 5, rtt = 13
ping to 127.0.0.1, seq = 6, rtt = time out
ping to 127.0.0.1, seq = 7, rtt = 38
ping to 127.0.0.1, seq = 8, rtt = time out
ping to 127.0.0.1, seq = 9, rtt = 165
ping to 127.0.0.1, seq = 10, rtt = time out
ping to 127.0.0.1, seq = 11, rtt = 66
ping to 127.0.0.1, seq = 12, rtt = 135
ping to 127.0.0.1, seq = 13, rtt = 48
ping to 127.0.0.1, seq = 14, rtt = 188
ping to 127.0.0.1, seq = 15, rtt = time out
*****
Attempted to send 15 messages to 127.0.0.1 on port 5000
Of the 11 successful packets received:
    the minimum RTT was 13ms
    the maximum RTT was 188ms
    the average RTT was 101ms

```

Screenshot of output on server side

```
Received from 127.0.0.1: PING 3331 19:49:46
  Reply sent.
Received from 127.0.0.1: PING 3332 19:49:46
  Reply sent.
Received from 127.0.0.1: PING 3333 19:49:46
  Reply sent.
Received from 127.0.0.1: PING 3334 19:49:46
  Reply sent.
Received from 127.0.0.1: PING 3335 19:49:46
  Reply sent.
Received from 127.0.0.1: PING 3336 19:49:46
  Reply not sent.
Received from 127.0.0.1: PING 3337 19:49:47
  Reply sent.
Received from 127.0.0.1: PING 3338 19:49:47
  Reply not sent.
Received from 127.0.0.1: PING 3339 19:49:47
  Reply sent.
Received from 127.0.0.1: PING 3340 19:49:48
  Reply not sent.
Received from 127.0.0.1: PING 3341 19:49:48
  Reply sent.
Received from 127.0.0.1: PING 3342 19:49:48
  Reply sent.
Received from 127.0.0.1: PING 3343 19:49:48
  Reply sent.
Received from 127.0.0.1: PING 3344 19:49:48
  Reply sent.
Received from 127.0.0.1: PING 3345 19:49:49
  Reply not sent.
```

Code below (also submitted as a .py file)

```

#!/usr/bin/python3
#coding: utf-8
"""
COMP9331 Lab2
Andrew Lau z3330164

Usage:
    python3 PingClient.py [HOST] [PORT]

Note that I have used the below template as a starting point:
https://webcms3.cse.unsw.edu.au/COMP3331/22T1/resources/70208
"""
from socket import *
from datetime import datetime
import time
import sys

# allow command line arguments, otherwise take defaults
if len(sys.argv) == 3:
    SERVER_NAME = sys.argv[1]
    SERVER_PORT = int(sys.argv[2])
else:
    SERVER_NAME = '127.0.0.1' # equivalent to 'localhost'
    SERVER_PORT = 2000 #change this port number if required

SEQUENCE_NUMBER_START = 3331
MAX_MESSAGES = 15

# This line creates the client's socket. The first parameter indicates the
# address family; in particular,AF_INET indicates that the underlying network
# is using IPv4.The second parameter indicates that the socket is of type
# SOCK_DGRAM,which means it is a UDP socket (rather than a TCP socket, where we
# use SOCK_STREAM).
client_socket = socket(AF_INET, SOCK_DGRAM)
# setting timeout to 600ms
client_socket.settimeout(0.6)

# list of RTTs for reporting at the end
rtt_list = []

# send MAX_MESSAGES number of messages
for message_no in range(MAX_MESSAGES):
    message = f'PING {SEQUENCE_NUMBER_START + message_no} {datetime.now().strftime("%H:%M:%S")}\r\n'

    time_start = time.time() # record when we went the message
    # Now that we have a socket and a message, we will want to send the message
    # through the socket to the destination host.
    client_socket.sendto(message.encode(),(SERVER_NAME, SERVER_PORT))
    # Note the difference between UDP sendto() and TCP send() calls. In TCP we do
    # not need to attach the destination address to the packet, while in UDP we
    # explicitly specify the destination address + Port No for each message

    # try to listen for a message
    try:
        modifiedMessage, serverAddress = client_socket.recvfrom(SERVER_PORT)
        # Note the difference between UDP recvfrom() and TCP recv() calls.
        # print message from server

```



```

        # print(modifiedMessage.decode())
        # print the received message
        # calculate RTT and convert to milliseconds
        rtt = round((time.time() - time_start) * 1000)
        rtt_list.append(rtt)
    except timeout: # timeout, packet loss
        rtt = 'time out'

    print(f'ping to {SERVER_NAME}, seq = {message_no + 1}, rtt = {rtt}')

client_socket.close()
# Close the socket

# report summary statistics of RTT
print('*' * 80)
print(f'Attempted to send {MAX_MESSAGES} messages to {SERVER_NAME} on port {SERVER_PORT}')
print(f'Of the {len(rtt_list)} successful packets received:')
print(f'\tthe minimum RTT was {min(rtt_list)}ms')
print(f'\tthe maximum RTT was {max(rtt_list)}ms')
print(f'\tthe average RTT was {round(sum(rtt_list) / len(rtt_list))}ms')

```