Lab 2

Jingshi Yang z5110579

EXERCISE 3

Question 1

Status Code: 200

Response Phrase: OK

Question 2

Last-Modified: Tue, 23 Sep 2003 05:29:00 GMT\r\n

The response contains the DATE header,

Date: Tue, 23 Sep 2003 05:29:50 GMT\r\n, the difference is the date when the file was created is 50s behind the last-modified date.

Question 3

The connection is consistent because this line is shown in the widow

Connection: Keep-Alive\r\n

This means it is using a sing TCP connection to handle HTTP requests and it is by default on HTTP 1.1

Question 4

73 bytes

Question 5

Text and html data

And the content is:

Congratulations. You've downloaded the file lab2-1.html!\n

Exercise 4

Question 1

No

Question 2

Yes

Last-Modified: Tue, 23 Sep 2003 05:35:00 GMT\r\n

Question 3

Yes

The if-modified-since header is a HTTP header that is sent to a server as a conditional request

If-Modified-Since: Tue, 23 Sep 2003 05:35:00 GMT\r\n

The if-none-match header: For GET and HEAD methods, the server will send back the requested source with a 200 status, only if it doesn't have an ETag matching the given ones

If-None-Match: "1bfef-173-8f4ae900"\r\n

Question 4

Status Code: 304

Response Phrase: Not Modified

No, the reason is that the GET request is only conditional, if the request has been modified after a given date, the server will send back the request resource, but if not modified since that date, then the request won't send back, and will be 304 without any content

Question 5

ETag: "1bfef-173-8f4ae900"\r\n

ETag response HTTP header is an identifier for a specific version of a resource

ETag value is unchanged since the 1st response message was received

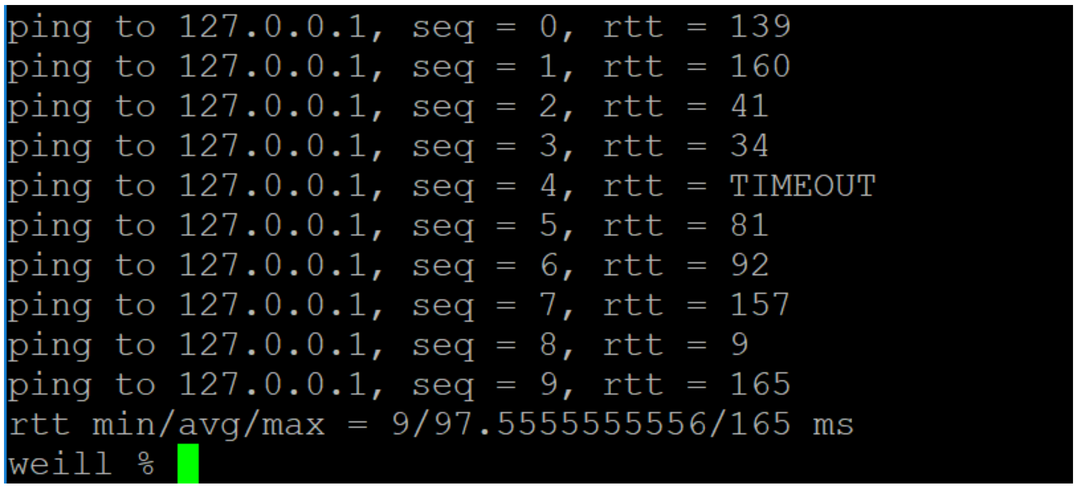
ETag allows caches to be more efficient and saves bandwidth, because if the content is unchanged, the server doesn't need to send a same full request.

EXERCISE 5

```
#!/usr/bin/env python3.6
import time
import sys
import numpy as np
total = len(sys.argv)

seq = 0
rtt_arr = []

def get_time():
        return int(round(time.time() * 1000))

from socket import *
serverName = str(sys.argv[total - 2])
serverPort = int(sys.argv[total - 1])

clientSocket = socket(AF_INET, SOCK_DGRAM)
clientSocket.settimeout(1.0)

while seq < 10:
    clientSocket.connect((serverName, serverPort))
    start = get_time()

    try:
        outmessage = "hello"
        clientSocket.send(outmessage.encode('utf-8'))
        modifiedMessage = clientSocket.recv(1024)

        end = get_time()
        rtt = end - start

        rtt_arr.append(rtt)
        #print(modifiedMessage)

        sentence = "ping to 127.0.0.1, seq = " + str(seq) + ", rtt
= " + str(rtt)
    except timeout:
        sentence = "ping to 127.0.0.1, seq = " + str(seq) + ", rtt
= " + "TIMEOUT"

    print(sentence)
    seq = seq + 1
```

avg_rtt = np.mean(rtt_arr)

print("rtt min/avg/max = " + str(min(rtt_arr)) + "/" + str(avg_rtt) + "/" + str(max(rtt_arr)) + " ms")



```
ping to 127.0.0.1, seq = 0, rtt = 139
ping to 127.0.0.1, seq = 1, rtt = 160
ping to 127.0.0.1, seq = 2, rtt = 41
ping to 127.0.0.1, seq = 3, rtt = 34
ping to 127.0.0.1, seq = 4, rtt = TIMEOUT
ping to 127.0.0.1, seq = 5, rtt = 81
ping to 127.0.0.1, seq = 6, rtt = 92
ping to 127.0.0.1, seq = 7, rtt = 157
ping to 127.0.0.1, seq = 8, rtt = 9
ping to 127.0.0.1, seq = 9, rtt = 165
rtt min/avg/max = 9/97.5555555556/165 ms
weill %
```