CS 6043/5143: Computer Networking

FALL 2017

PROJECT 1

Given: Sept. 20, 2017

Due: Oct. 6 (Friday), 2017 (NO LATER THAN 11:59PM)

Group Members:

Anshul Gautam

Submission Instructions:

1. Submit only on-line files on Blackboard before midnight. No hard copy will be accepted.

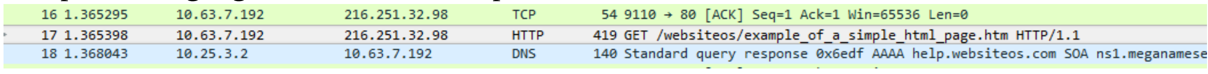
2. For students who are working in a team, *one* submission for the team is sufficient.

3. Wireshark files for this project can be found in the zip file "Project_1_Wireshark_Traces_Fall2017.zip".

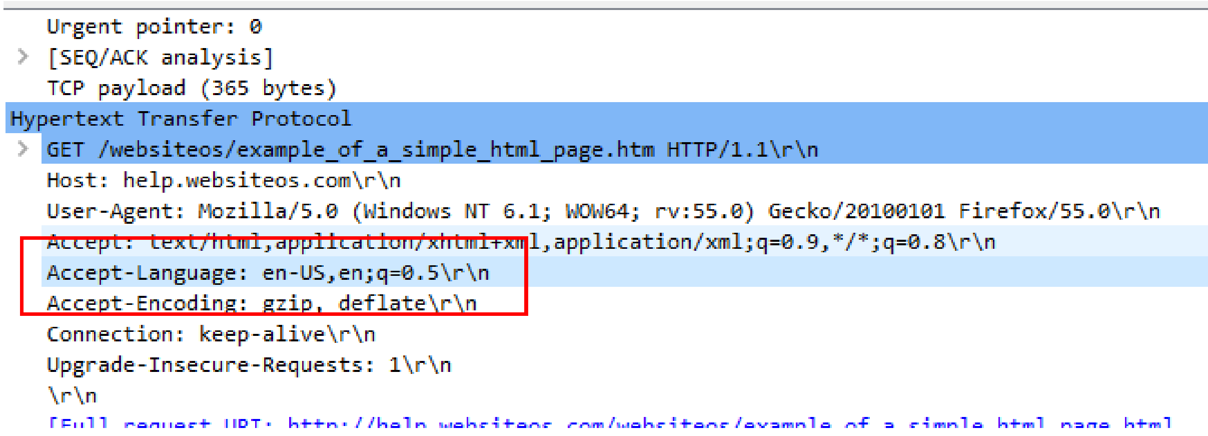
Total possible points: 10

Part I: HTTP

1. (0.5 pts) Load the file 'http-trace-1.pcapng' in Wireshark and answer the following questions. The traces were collected when a simple and very short HTML file was downloaded by the client (10.63.7.192).
   a) What languages (if any) does the client browser indicate that it can accept to the server?
   Answer
   In the HTTP request sent to the server, the client indicates "En-us, en" as the acceptable language as seen in the snapshots below.



```
16 1.365295    10.63.7.192    216.251.32.98    TCP     54 9110 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
17 1.365398    10.63.7.192    216.251.32.98    HTTP   419 GET /websiteos/example_of_a_simple_html_page.htm HTTP/1.1
18 1.368043    10.25.3.2      10.63.7.192      DNS    140 Standard query response 0x6edf AAAA help.websiteos.com SOA ns1.meganamese
```

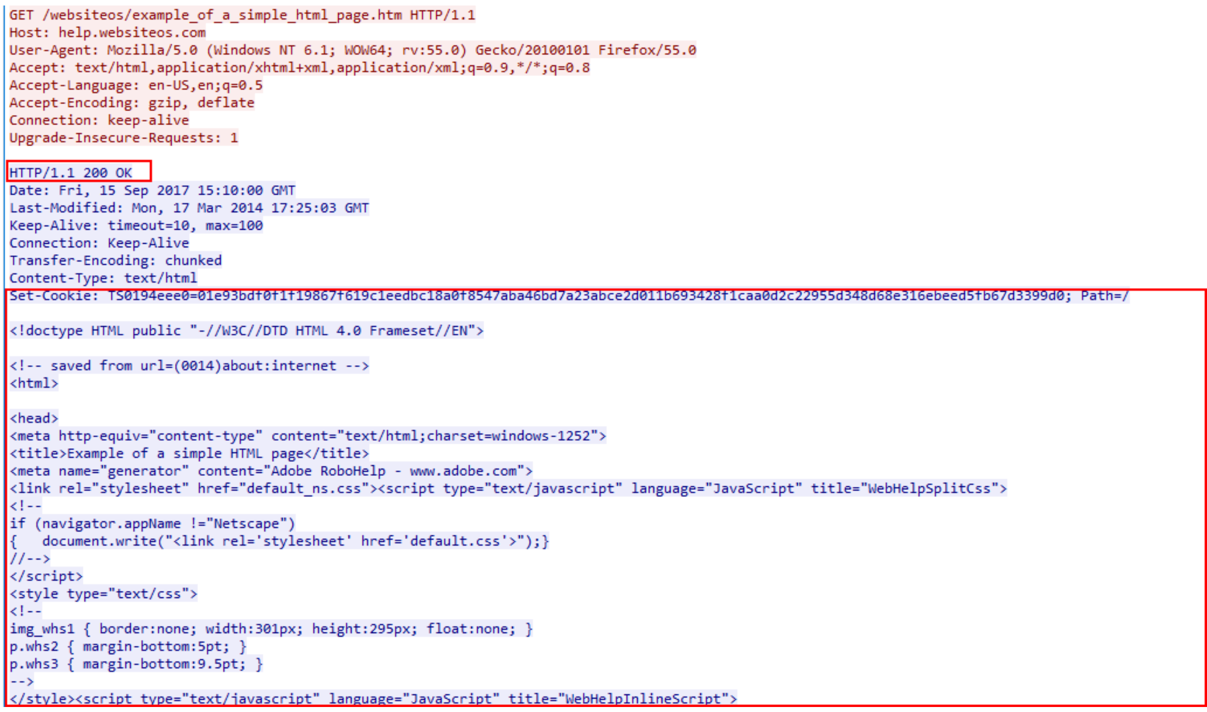


```
    Urgent pointer: 0
  > [SEQ/ACK analysis]
    TCP payload (365 bytes)
Hypertext Transfer Protocol
  > GET /websiteos/example_of_a_simple_html_page.htm HTTP/1.1\r\n
    Host: help.websiteos.com\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:55.0) Gecko/20100101 Firefox/55.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Full request URI: http://help.websiteos.com/websiteos/example_of_a_simple_html_page.html
```

b) What is the status code returned from the server to the client browser for the first GET request? What does the code mean?

Answer

The status code returned by the server to the client is 200 OK which means the requested file was found and retrieved successfully, and the HTML code is present in the response HTTP packet.



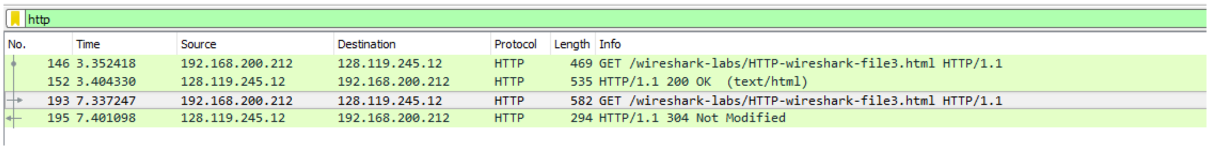
```
GET /websiteos/example_of_a_simple_html_page.htm HTTP/1.1
Host: help.websiteos.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:55.0) Gecko/20100101 Firefox/55.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Fri, 15 Sep 2017 15:10:00 GMT
Last-Modified: Mon, 17 Mar 2014 17:25:03 GMT
Keep-Alive: timeout=10, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html
Set-Cookie: TS0194eee0=01e93bdf0f1f19867f619c1eedbc18a0f8547aba46bd7a23abce2d011b693428f1caa0d2c22955d348d68e316ebeed5fb67d3399d0; Path=/

<!doctype HTML public "-//W3C//DTD HTML 4.0 Frameset//EN">

<!-- saved from url=(0014)about:internet -->
<html>

<head>
<meta http-equiv="content-type" content="text/html;charset=windows-1252">
<title>Example of a simple HTML page</title>
<meta name="generator" content="Adobe RoboHelp - www.adobe.com">
<link rel="stylesheet" href="default_ns.css"><script type="text/javascript" language="JavaScript" title="WebHelpSplitCss">
<!--
if (navigator.appName !="Netscape")
{   document.write("<link rel='stylesheet' href='default.css'>");}
//-->
</script>
<style type="text/css">
<!--
img_whs1 { border:none; width:301px; height:295px; float:none; }
p.whs2 { margin-bottom:5pt; }
p.whs3 { margin-bottom:9.5pt; }
-->
</style><script type="text/javascript" language="JavaScript" title="WebHelpInlineScript">
```

2. (0.5 pts) Load the file ‘http-trace-2.pcapng’ in Wireshark and answer the following questions. The traces were collected when a particular web page is accessed twice from the browser within a short interval.

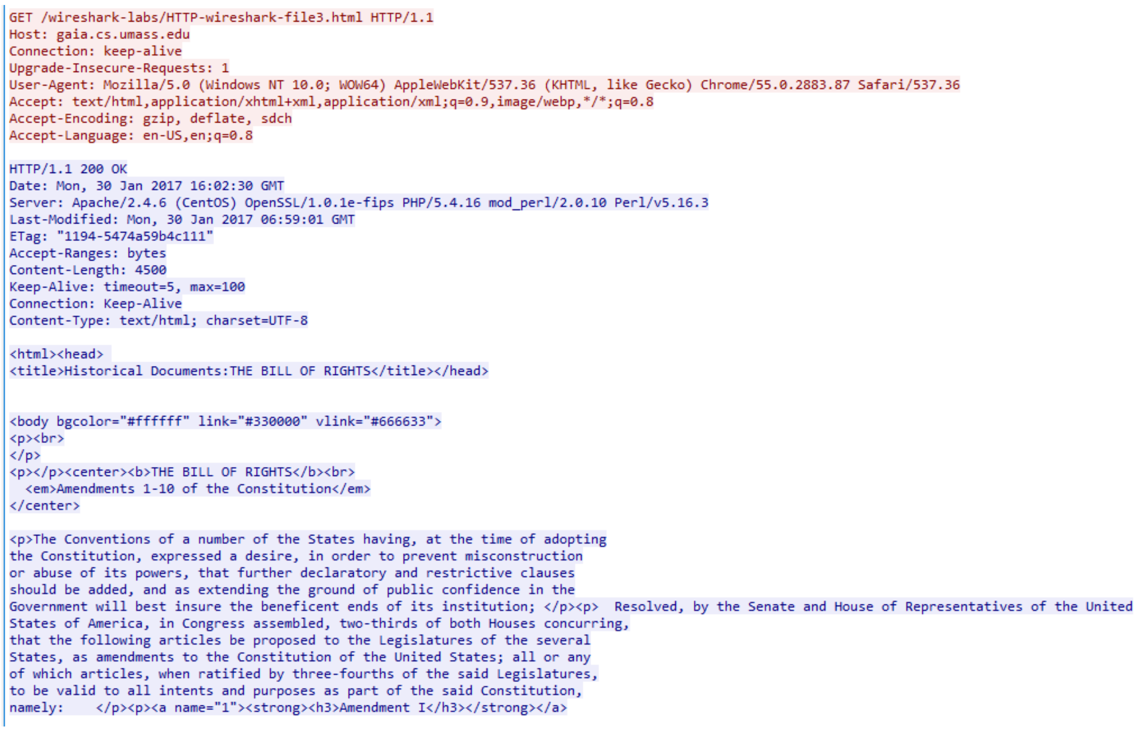
   a) Inspect the contents of the first server response. Did the server explicitly return the contents of the file? If so, what was returned? (provide a screenshot)

Answer

In the first response, the server responded with 200 OK code and the requested HTML content as seen in snapshots below.



```
http
No.  Time      Source          Destination     Protocol Length Info
 146 3.352418  192.168.200.212 128.119.245.12  HTTP     469 GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
 152 3.404330  128.119.245.12  192.168.200.212 HTTP     535 HTTP/1.1 200 OK  (text/html)
 193 7.337247  192.168.200.212 128.119.245.12  HTTP     582 GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
 195 7.401098  128.119.245.12  192.168.200.212 HTTP     294 HTTP/1.1 304 Not Modified
```

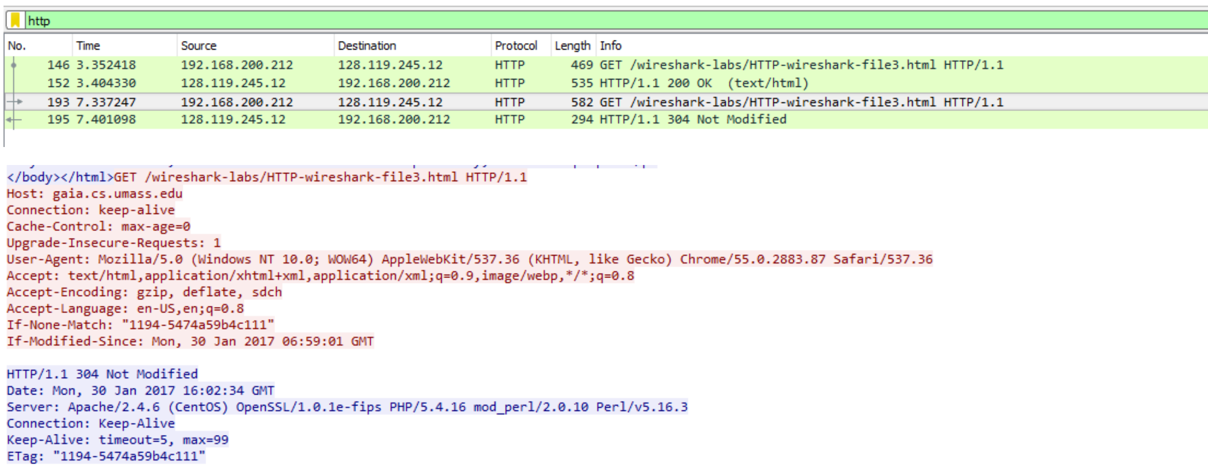


```
GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
Host: gaia.cs.umass.edu
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8

HTTP/1.1 200 OK
Date: Mon, 30 Jan 2017 16:02:30 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/5.4.16 mod_perl/2.0.10 Perl/v5.16.3
Last-Modified: Mon, 30 Jan 2017 06:59:01 GMT
ETag: "1194-5474a59b4c111"
Accept-Ranges: bytes
Content-Length: 4500
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<html><head>
<title>Historical Documents:THE BILL OF RIGHTS</title></head>


<body bgcolor="#ffffff" link="#330000" vlink="#666633">
<p><br>
</p>
<p></p><center><b>THE BILL OF RIGHTS</b><br>
  <em>Amendments 1-10 of the Constitution</em>
</center>

<p>The Conventions of a number of the States having, at the time of adopting
the Constitution, expressed a desire, in order to prevent misconstruction
or abuse of its powers, that further declaratory and restrictive clauses
should be added, and as extending the ground of public confidence in the
Government will best insure the beneficent ends of its institution; </p><p>  Resolved, by the Senate and House of Representatives of the United
States of America, in Congress assembled, two-thirds of both Houses concurring,
that the following articles be proposed to the Legislatures of the several
States, as amendments to the Constitution of the United States; all or any
of which articles, when ratified by three-fourths of the said Legislatures,
to be valid to all intents and purposes as part of the said Constitution,
namely:    </p><p><a name="1"><strong><h3>Amendment I</h3></strong></a>
```

b) What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain why it did or did not.

Answer

For the second request on the same file, the server responded with the 304 response code, which indicates that the file is not modified since the last request from client. The server didn't respond with explicit content as it would be redundant and adds extra load on the server to create the same payload again. This process optimizes the performance of the server. In case if the file on server was modified since the last request from client, the server would respond with the new content of the file as the time stamp is updated on the file.



```
http
No.    Time        Source            Destination       Protocol  Length  Info
  146 3.352418    192.168.200.212   128.119.245.12    HTTP        469 GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
  152 3.404330    128.119.245.12    192.168.200.212   HTTP        535 HTTP/1.1 200 OK  (text/html)
  193 7.337247    192.168.200.212   128.119.245.12    HTTP        582 GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
  195 7.401098    128.119.245.12    192.168.200.212   HTTP        294 HTTP/1.1 304 Not Modified
```

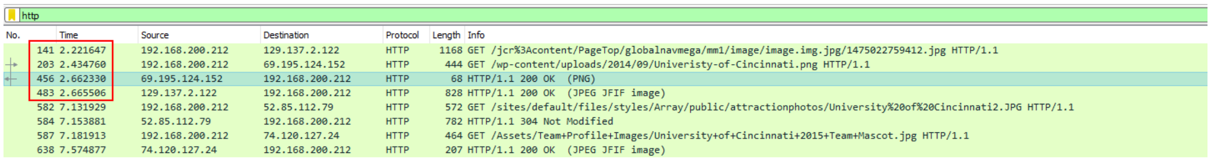
```
</body></html>GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
Host: gaia.cs.umass.edu
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
If-None-Match: "1194-5474a59b4c111"
If-Modified-Since: Mon, 30 Jan 2017 06:59:01 GMT

HTTP/1.1 304 Not Modified
Date: Mon, 30 Jan 2017 16:02:34 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips PHP/5.4.16 mod_perl/2.0.10 Perl/v5.16.3
Connection: Keep-Alive
Keep-Alive: timeout=5, max=99
ETag: "1194-5474a59b4c111"
```

3. (1 pts) Load the file 'http-trace-3.pcapng' in Wireshark and answer the following question.

   The traces were collected when HTTP GET requests were sent for four image files (one PNG, three JPEGs). Can you tell whether the client browser downloaded the first two images (one JPEG and one PNG) serially or in parallel? Explain how you came to your conclusion.

   Answer

   The first two images JPEG and PNG are downloaded serially as both the packets have different time stamp as seen in the snapshot below. The JPEG request sent from 192.168.200.212 to 129.137.2.122 was sent at time 2.221647 and the client received the file at 2.665506. Second request for PNG file is sent from 192.168.200.212 to 69.195.124.152 at time 2.434760 and the file was received at 2.662330.



```
http
No.    Time        Source            Destination       Protocol  Length  Info
  141 2.221647    192.168.200.212   129.137.2.122     HTTP       1168 GET /jcr%3Acontent/PageTop/globalnavmega/mm1/image/image.img.jpg/1475022759412.jpg HTTP/1.1
  203 2.434760    192.168.200.212   69.195.124.152    HTTP        444 GET /wp-content/uploads/2014/09/Univeristy-of-Cincinnati.png HTTP/1.1
  456 2.662330    69.195.124.152    192.168.200.212   HTTP         68 HTTP/1.1 200 OK  (PNG)
  483 2.665506    129.137.2.122     192.168.200.212   HTTP        828 HTTP/1.1 200 OK  (JPEG JFIF image)
  582 7.131929    192.168.200.212   52.85.112.79      HTTP        572 GET /sites/default/files/styles/Array/public/attractionphotos/University%20of%20Cincinnati2.JPG HTTP/1.1
  584 7.153881    52.85.112.79      192.168.200.212   HTTP        782 HTTP/1.1 304 Not Modified
  587 7.181913    192.168.200.212   74.120.127.24     HTTP        464 GET /Assets/Team+Profile+Images/University+of+Cincinnati+2015+Team+Mascot.jpg HTTP/1.1
  638 7.574877    74.120.127.24     192.168.200.212   HTTP        207 HTTP/1.1 200 OK  (JPEG JFIF image)
```

4. (1 pts) Enter the following URL into your browser and type the requested user name and password into the pop up box.

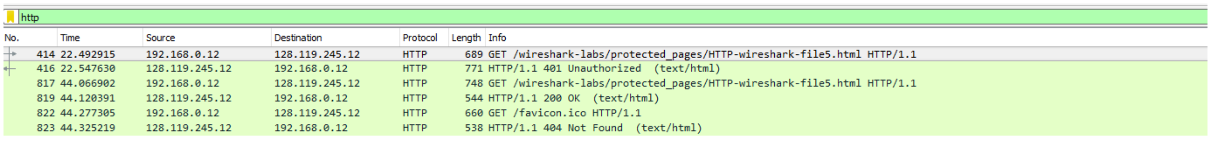
   http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html

   The username is "wireshark-students", and the password is "network" (without the quotes).

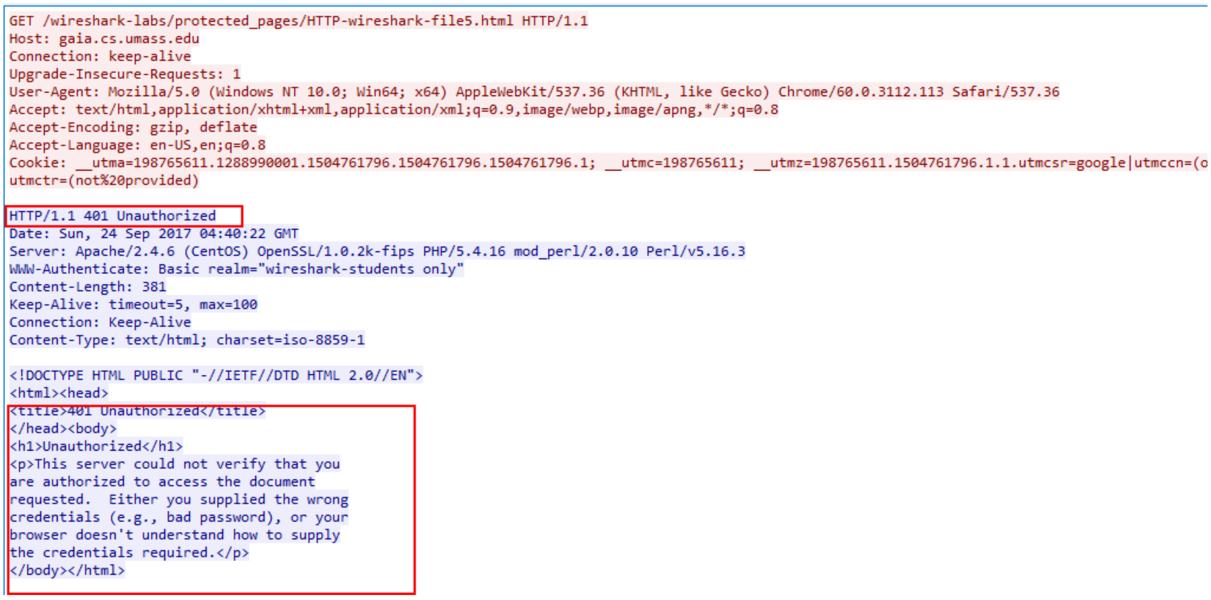
   a) What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?

Answer

The server responded with the 401 “Unauthorized” to the initial HTTP GET message.



```
http
No.   Time       Source          Destination     Protocol Length Info
  414 22.492915  192.168.0.12    128.119.245.12  HTTP     689 GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
  416 22.547630  128.119.245.12  192.168.0.12    HTTP     771 HTTP/1.1 401 Unauthorized  (text/html)
  817 44.066902  192.168.0.12    128.119.245.12  HTTP     748 GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
  819 44.120391  128.119.245.12  192.168.0.12    HTTP     544 HTTP/1.1 200 OK  (text/html)
  822 44.277305  192.168.0.12    128.119.245.12  HTTP     660 GET /favicon.ico HTTP/1.1
  823 44.325219  128.119.245.12  192.168.0.12    HTTP     538 HTTP/1.1 404 Not Found  (text/html)
```

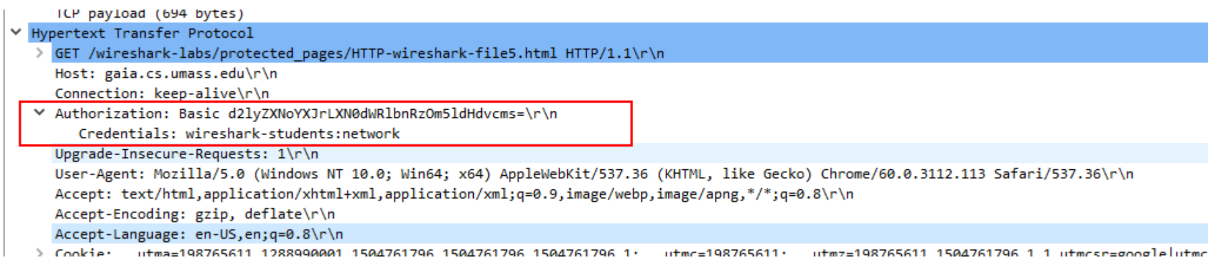


```
GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
Host: gaia.cs.umass.edu
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cookie: __utma=198765611.1288990001.1504761796.1504761796.1504761796.1; __utmc=198765611; __utmz=198765611.1504761796.1.1.utmcsr=google|utmccn=(o
utmctr=(not%20provided)

HTTP/1.1 401 Unauthorized
Date: Sun, 24 Sep 2017 04:40:22 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16 mod_perl/2.0.10 Perl/v5.16.3
WWW-Authenticate: Basic realm="wireshark-students only"
Content-Length: 381
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Unauthorized</title>
</head><body>
<h1>Unauthorized</h1>
<p>This server could not verify that you
are authorized to access the document
requested.  Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
</body></html>
```

b) When your browser sends the HTTP GET message for the second time, what new field is included in the HTTP GET message? Is the login credentials encrypted or is it sent as plain text?

Answer:

For the second HTTP GET message, the login credentials are sent for authorization. The login is not encrypted as it can be seen in plain text in the HTTP request below.



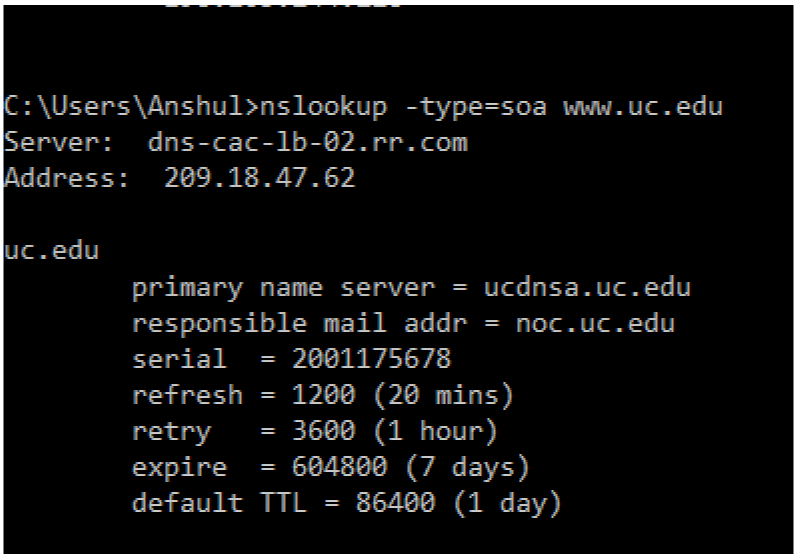
```
      TCP payload (694 bytes)
 ˅ Hypertext Transfer Protocol
   > GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1\r\n
     Host: gaia.cs.umass.edu\r\n
     Connection: keep-alive\r\n
   ˅ Authorization: Basic d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=\r\n
       Credentials: wireshark-students:network
     Upgrade-Insecure-Requests: 1\r\n
     User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36\r\n
     Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8\r\n
     Accept-Encoding: gzip, deflate\r\n
     Accept-Language: en-US,en;q=0.8\r\n
   > Cookie: __utma=198765611.1288990001.1504761796.1504761796.1504761796.1; __utmc=198765611; __utmz=198765611.1504761796.1.1.utmcsr=google|utmc
```

Part II: DNS

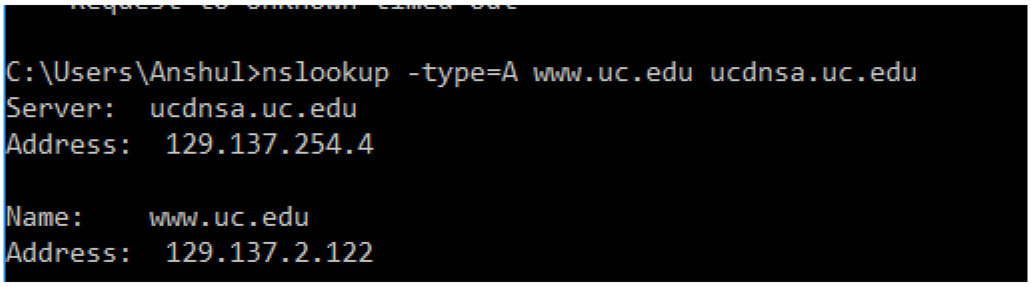
1. (0.5 pts) Run *nslookup* to determine the authoritative DNS server and its IP for www.uc.edu. Include screenshot in your answer.

Answer

The authoritative DNS for www.uc.edu is ucdnsa.uc.edu and its stored IP address for www.uc.edu is 129.137.2.122



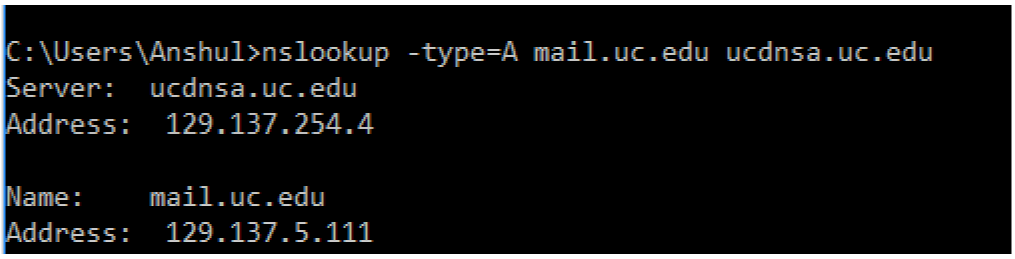
```
C:\Users\Anshul>nslookup -type=soa www.uc.edu
Server:  dns-cac-lb-02.rr.com
Address:  209.18.47.62

uc.edu
        primary name server = ucdnsa.uc.edu
        responsible mail addr = noc.uc.edu
        serial  = 2001175678
        refresh = 1200 (20 mins)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        default TTL = 86400 (1 day)
```



```
C:\Users\Anshul>nslookup -type=A www.uc.edu ucdnsa.uc.edu
Server:  ucdnsa.uc.edu
Address:  129.137.254.4

Name:    www.uc.edu
Address:  129.137.2.122
```

2. (0.5 pts) Run *nslookup* so that the authoritative DNS server obtained in Question 1 is queried for mail.uc.edu. Mention the IP address(es) for mail.uc.edu. Include screenshot.

   Answer:
   The IP address of mail.uc.edu is 129/137.5.111



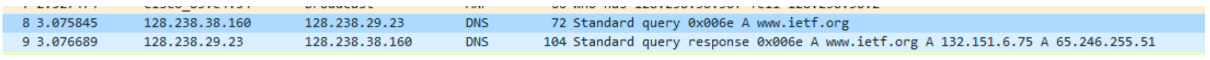
```
C:\Users\Anshul>nslookup -type=A mail.uc.edu ucdnsa.uc.edu
Server:  ucdnsa.uc.edu
Address:  129.137.254.4

Name:    mail.uc.edu
Address:  129.137.5.111
```

3. (0.5 pts) Load the file 'dns-trace-1.pcapng' in Wireshark and answer the following questions.

a) Locate the DNS query and response messages. Are they sent over UDP or TCP? What is the destination port for the DNS query message and the source port of DNS response message?

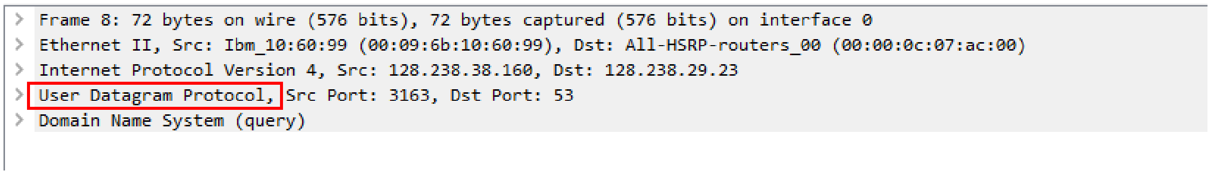
Answer:

Image below shows the DNS query. The request is sent over UDP as seen in the 2nd snapshot. The destination port for DNS query message is 53 and the source port of DNS response message is 53 as seen in the snapshots below



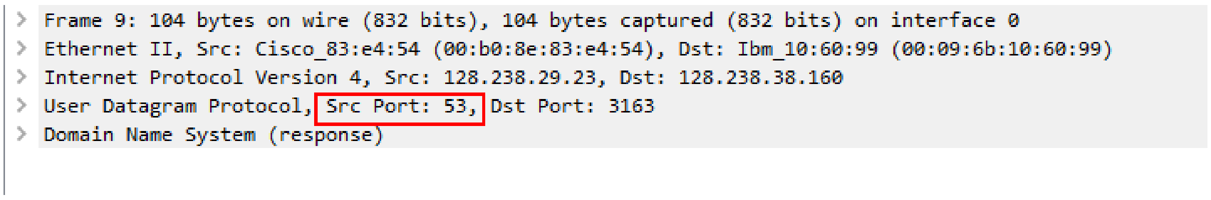
```
8 3.075845    128.238.38.160    128.238.29.23     DNS     72 Standard query 0x006e A www.ietf.org
9 3.076689    128.238.29.23     128.238.38.160    DNS    104 Standard query response 0x006e A www.ietf.org A 132.151.6.75 A 65.246.255.51
```

DNS Request



```
> Frame 8: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface 0
> Ethernet II, Src: Ibm_10:60:99 (00:09:6b:10:60:99), Dst: All-HSRP-routers_00 (00:00:0c:07:ac:00)
> Internet Protocol Version 4, Src: 128.238.38.160, Dst: 128.238.29.23
> User Datagram Protocol, Src Port: 3163, Dst Port: 53
> Domain Name System (query)
```

DNS Response

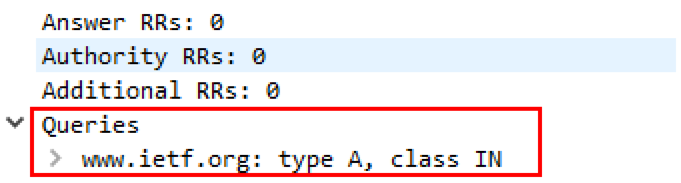


```
> Frame 9: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
> Ethernet II, Src: Cisco_83:e4:54 (00:b0:8e:83:e4:54), Dst: Ibm_10:60:99 (00:09:6b:10:60:99)
> Internet Protocol Version 4, Src: 128.238.29.23, Dst: 128.238.38.160
> User Datagram Protocol, Src Port: 53, Dst Port: 3163
> Domain Name System (response)
```

b) Examine the DNS query message. What "Type" of DNS query is it?

Answer:

The DNS query is Type A as seen in the snapshot

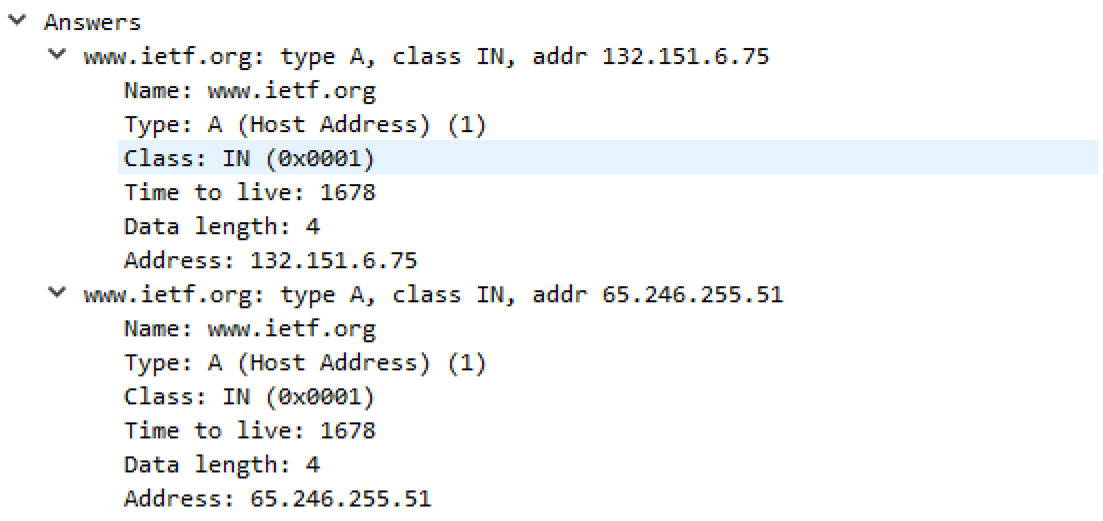


```
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
v Queries
  > www.ietf.org: type A, class IN
```

c) Examine the DNS response message. How many "answers" are provided? What does each of these answers contain?

Answer:

Two answers are provided in the DNS response. Both answers contain the IP address to access www.ietf.org



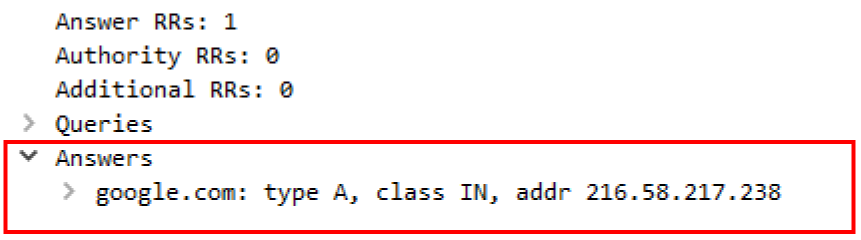
```
Answers
  www.ietf.org: type A, class IN, addr 132.151.6.75
    Name: www.ietf.org
    Type: A (Host Address) (1)
    Class: IN (0x0001)
    Time to live: 1678
    Data length: 4
    Address: 132.151.6.75
  www.ietf.org: type A, class IN, addr 65.246.255.51
    Name: www.ietf.org
    Type: A (Host Address) (1)
    Class: IN (0x0001)
    Time to live: 1678
    Data length: 4
    Address: 65.246.255.51
```

4. (0.5 pts) Load the file 'dns-trace-2.pcapng' in Wireshark and answer the following questions.

   a) Examine the fourth DNS response message (serial no. 19). What are the type, class, and address received for google.com?

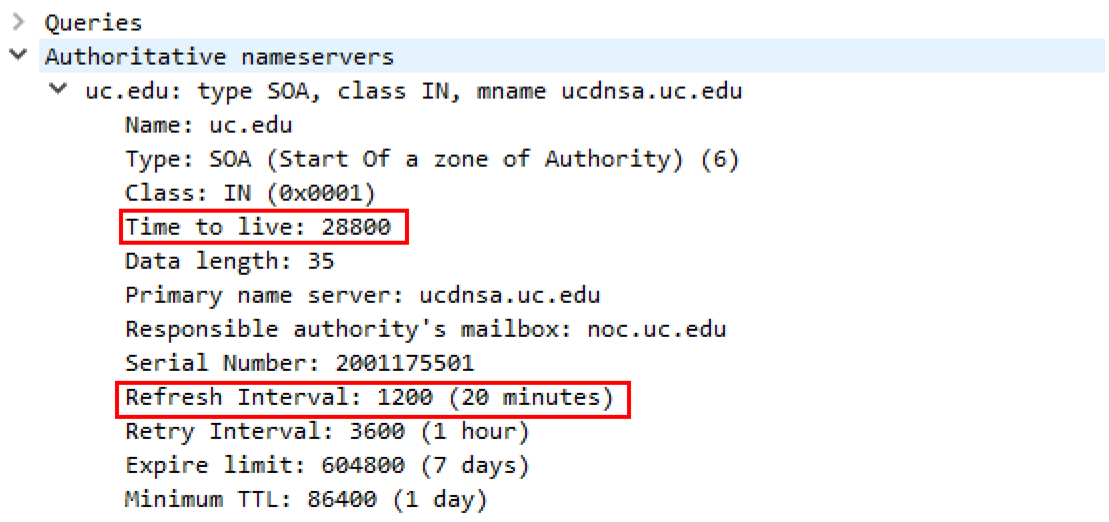
   Answer

   Type is A, class is IN, and the address is 216.58.217.238 as seen in the snapshot below



```
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  Queries
  Answers
    google.com: type A, class IN, addr 216.58.217.238
```

   b) Examine the tenth DNS response message (serial no. 39). What are the refresh interval and minimum TTL for UC mail server?
   The refresh interval is 20 min and TTL is 28800 as seen in the snapshot below.



```
> Queries
v Authoritative nameservers
  v uc.edu: type SOA, class IN, mname ucdnsa.uc.edu
      Name: uc.edu
      Type: SOA (Start Of a zone of Authority) (6)
      Class: IN (0x0001)
      Time to live: 28800
      Data length: 35
      Primary name server: ucdnsa.uc.edu
      Responsible authority's mailbox: noc.uc.edu
      Serial Number: 2001175501
      Refresh Interval: 1200 (20 minutes)
      Retry Interval: 3600 (1 hour)
      Expire limit: 604800 (7 days)
      Minimum TTL: 86400 (1 day)
```

Part III: Socket Programming

1. (2.5 pts) Web Server

You will develop a web server that handles one HTTP request at a time. Your web server should accept and parse the HTTP request, get the requested file from the server's file system, create an HTTP response message consisting of the requested file preceded by header lines, and then send the response directly to the client. If the requested file is not present in the server, the server should send an HTTP "404 Not Found" message back to the client.

Below you will find the skeleton code for the Web server. You are to complete the skeleton code. The places where you need to fill in code are marked with #Fill in start and #Fill in end. Each place may require one or more lines of code.

Put an HTML file (e.g., HelloWorld.html) in the same directory that the server is in. Run the server program. Determine the IP address of the host that is running the server (e.g., 128.238.251.26). From another host, open a browser and provide the corresponding URL. For example:

http://128.238.251.26:6789/HelloWorld.html

'HelloWorld.html' is the name of the file you placed in the server directory. Note also the use of the port number after the colon. You need to replace this port number with whatever port you have used in the server code. In the above example, we have used the port number 6789. The browser should then display the contents of HelloWorld.html. If you omit ":6789", the browser will assume port 80 and you will get the web page from the server only if your server is listening at port 80.

Then try to get a file that is not present at the server. You should get a “404 Not Found” message.

You will hand in the complete server code along with the screen shots of your client browser, verifying that you actually receive the contents of the HTML file from the server.

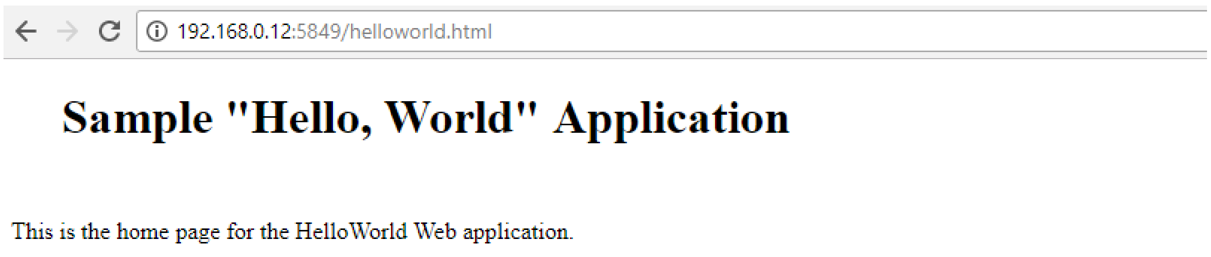
Answer:
Please see the attached python code and the sample html file used to test the code.
Here are the steps to run the code successfully

1. Place the html file in the directory where the Python code is running. If you’re unsure about the current working directory of the code, there is a cwd code added in the file which shows you the current working directory. Run the code without placing the html file in order to see the CWD.
2. Change the IP address and port number fields to your computers IP address and any desired port number.
3. Once the server is running, load the file into browser using the URL http://192.168.1.12:5849/helloworld.html. You should see the file loaded.
4. Change the file name to receive the 404 error code.

Skeleton Python Code for the Web Server:

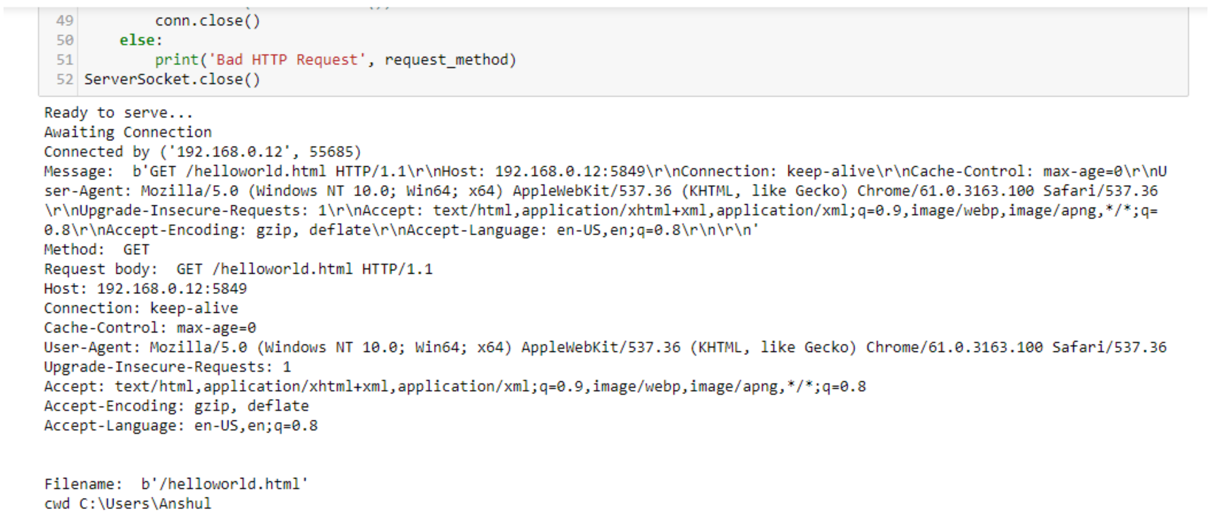
```
#import socket module
from socket import *
HOST =
serverSocket = socket(AF_INET, SOCK_STREAM)

#Prepare a sever socket
#Fill in start
#Fill in end
while True:
    #Establish the connection
    print('Ready to serve...')
    connectionSocket, addr =   #Fill in start              #Fill in end
    try:
        message =   #Fill in start          #Fill in end
        filename = message.split()[1]
        f     = open(filename[1:])
        outputdata = #Fill in start       #Fill in end

        #Send one HTTP header line into socket
        #Fill in start
        #Fill in end

        #Send the content of the requested file to the client
        for i in range(0, len(outputdata)):
            connectionSocket.send(outputdata[i].encode())
        connectionSocket.send("\r\n".encode())
        connectionSocket.close()
    except IOError:
        #Send response message for file not found
        #Fill in start
        #Fill in end

        #Close client socket
        #Fill in start
        #Fill in end
serverSocket.close()
```

Output for successful HTTP Request


```
192.168.0.12:5849/helloworld.html
```

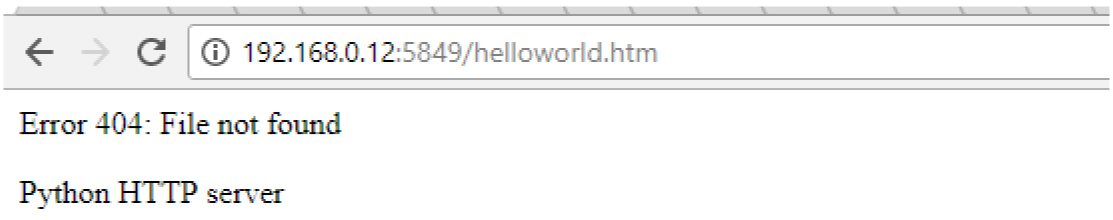
Sample "Hello, World" Application

This is the home page for the HelloWorld Web application.


```
        conn.close()
    else:
        print('Bad HTTP Request', request_method)
ServerSocket.close()
```

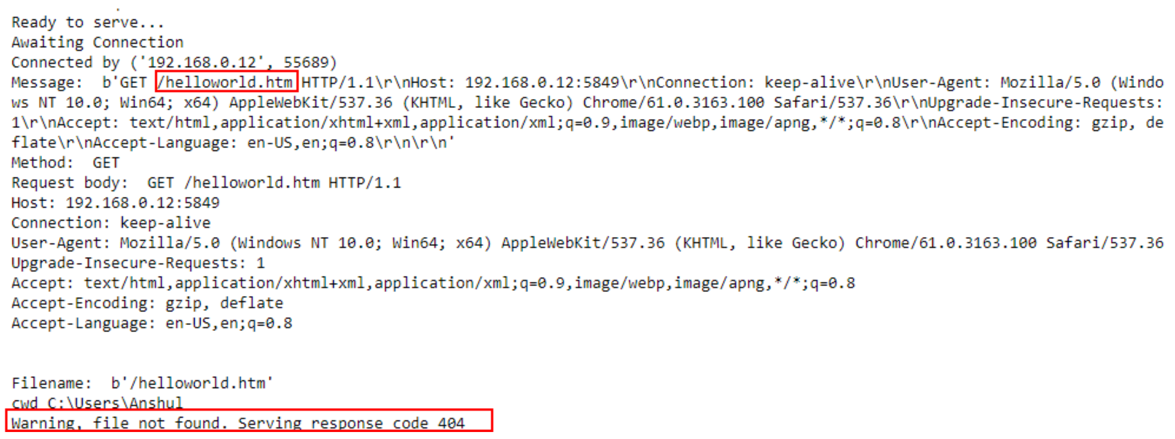
```
Ready to serve...
Awaiting Connection
Connected by ('192.168.0.12', 55685)
Message:  b'GET /helloworld.html HTTP/1.1\r\nHost: 192.168.0.12:5849\r\nConnection: keep-alive\r\nCache-Control: max-age=0\r\nU
ser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36
\r\nUpgrade-Insecure-Requests: 1\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=
0.8\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: en-US,en;q=0.8\r\n\r\n'
Method:  GET
Request body:  GET /helloworld.html HTTP/1.1
Host: 192.168.0.12:5849
Connection: keep-alive
Cache-Control: max-age=0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8


Filename:  b'/helloworld.html'
cwd C:\Users\Anshul
```

Output for wrong HTML file


```
192.168.0.12:5849/helloworld.htm
```

Error 404: File not found

Python HTTP server



```
Ready to serve...
Awaiting Connection
Connected by ('192.168.0.12', 55689)
Message:  b'GET /helloworld.htm HTTP/1.1\r\nHost: 192.168.0.12:5849\r\nConnection: keep-alive\r\nUser-Agent: Mozilla/5.0 (Windo
ws NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36\r\nUpgrade-Insecure-Requests:
1\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8\r\nAccept-Encoding: gzip, de
flate\r\nAccept-Language: en-US,en;q=0.8\r\n\r\n'
Method:  GET
Request body:  GET /helloworld.htm HTTP/1.1
Host: 192.168.0.12:5849
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36
Upgrade-Insecure-Requests: 1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8


Filename:  b'/helloworld.htm'
cwd C:\Users\Anshul
Warning, file not found. Serving response code 404
```

2. (2.5 pts) Simple Mail Client

You will develop a simple mail client that sends email to any recipient. Your client will need to connect to a mail server, dialogue with the mail server using the SMTP protocol, and send an email message to the mail server. Python provides a module, called `smtplib`, which has built in methods to send mail using SMTP protocol. However, we will not be using this module in this lab, because it hide the details of SMTP and socket programming.

In order to limit spam, some mail servers do not accept TCP connection from arbitrary sources. For the experiment described below, you may want to try connecting both to your university mail server and to a popular Webmail server, such as a AOL mail server. You may also try making your connection both from your home and from your university campus.

Below you will find the skeleton code for the client. You are to complete the skeleton code. The places where you need to fill in code are marked with **`#Fill in start`** and **`#Fill in end`**. Each place may require one or more lines of code.

In some cases, the receiving mail server might classify your e-mail as junk. Make sure you check the junk/spam folder when you look for the e-mail sent from your client.

In your submission, you are to provide the complete code for your SMTP mail client as well as a screenshot showing that you indeed receive the e-mail message.

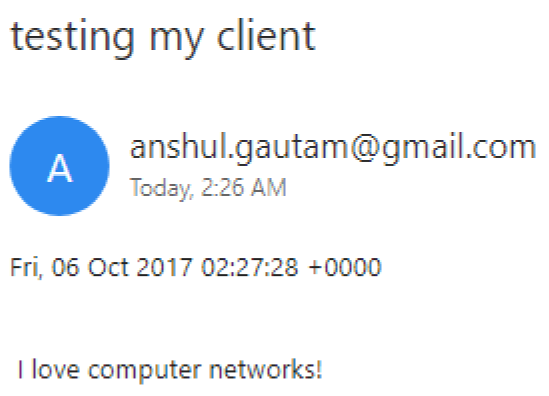
Answer:

Please see the attached python code.
We've used SSL socket created on top of the socket connection to connect with gmail server. Gmail was configured to receive requests from lesser secured apps to send the email. Here is a link to configure Gmail with lower security,
https://support.google.com/accounts/answer/6010255?hl=en
In order to test the code, follow these steps.

1. Lower the security of your gmail
2. Change the username, password, from, and to fields to your desired email ids.

Skeleton Python Code for the Mail Client:

```
from socket import *
msg = "\r\n I love computer networks!"
endmsg = "\r\n.\r\n"

# Choose a mail server (e.g. Google mail server) and call it mailserver
mailserver = #Fill in start   #Fill in end

# Create socket called clientSocket and establish a TCP connection with
mailserver
#Fill in start
#Fill in end
recv = clientSocket.recv(1024).decode()
print(recv)
if recv[:3] != '220':
      print('220 reply not received from server.')

# Send HELO command and print server response.
heloCommand = 'HELO Alice\r\n'
clientSocket.send(heloCommand.encode())
recv1 = clientSocket.recv(1024).decode()
print(recv1)
if recv1[:3] != '250':
    print('250 reply not received from server.')
```

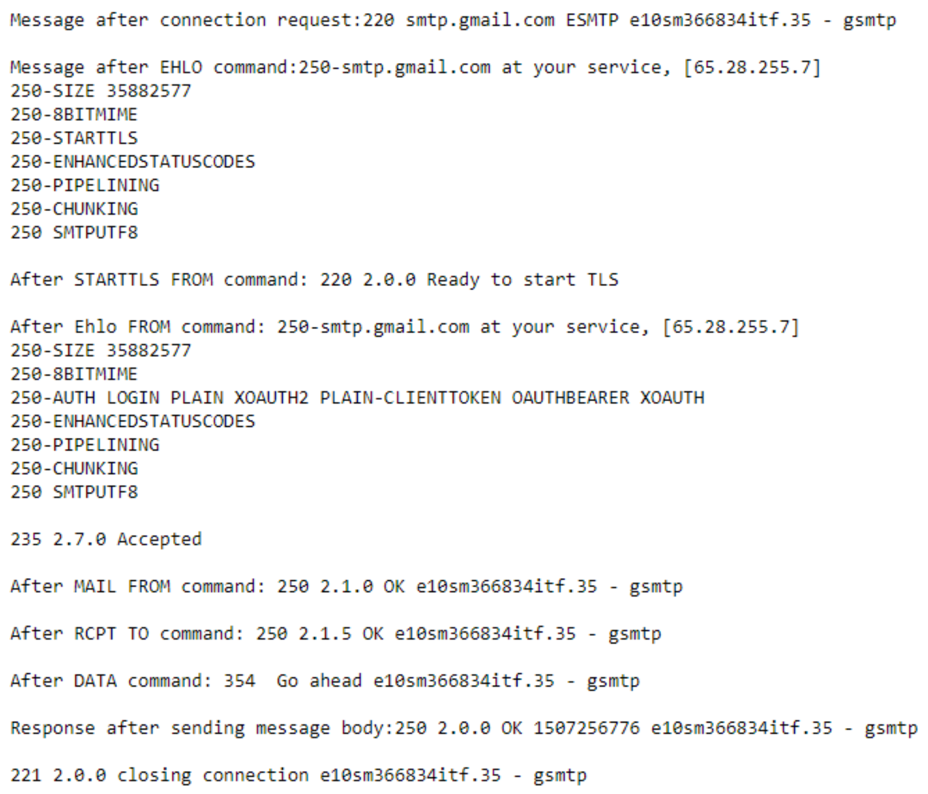


testing my client

anshul.gautam@gmail.com
Today, 2:26 AM

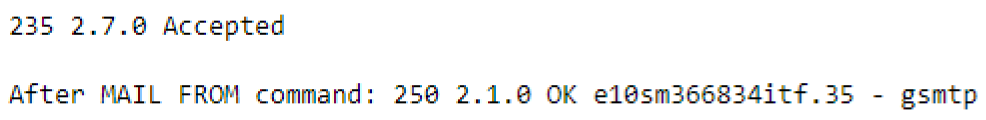
Fri, 06 Oct 2017 02:27:28 +0000

I love computer networks!



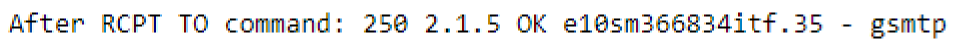
```
Message after connection request:220 smtp.gmail.com ESMTP e10sm366834itf.35 - gsmtp

Message after EHLO command:250-smtp.gmail.com at your service, [65.28.255.7]
250-SIZE 35882577
250-8BITMIME
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8

After STARTTLS FROM command: 220 2.0.0 Ready to start TLS

After Ehlo FROM command: 250-smtp.gmail.com at your service, [65.28.255.7]
250-SIZE 35882577
250-8BITMIME
250-AUTH LOGIN PLAIN XOAUTH2 PLAIN-CLIENTTOKEN OAUTHBEARER XOAUTH
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8

235 2.7.0 Accepted

After MAIL FROM command: 250 2.1.0 OK e10sm366834itf.35 - gsmtp

After RCPT TO command: 250 2.1.5 OK e10sm366834itf.35 - gsmtp

After DATA command: 354  Go ahead e10sm366834itf.35 - gsmtp

Response after sending message body:250 2.0.0 OK 1507256776 e10sm366834itf.35 - gsmtp

221 2.0.0 closing connection e10sm366834itf.35 - gsmtp
```

```
# Send MAIL FROM command and print server response.
# Fill in start
```



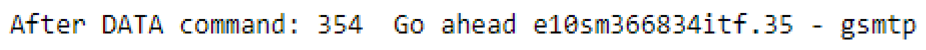
```
235 2.7.0 Accepted

After MAIL FROM command: 250 2.1.0 OK e10sm366834itf.35 - gsmtp
```

```
# Fill in end

# Send RCPT TO command and print server response.
# Fill in start
```



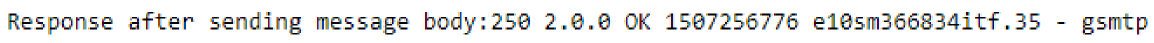
```
After RCPT TO command: 250 2.1.5 OK e10sm366834itf.35 - gsmtp
```

```
# Fill in end

# Send DATA command and print server response.
# Fill in start
```



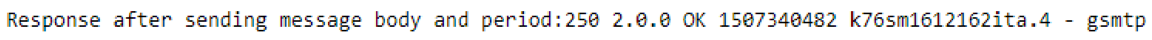
```
After DATA command: 354  Go ahead e10sm366834itf.35 - gsmtp
```

```
# Fill in end

# Send message data.
# Fill in start
```



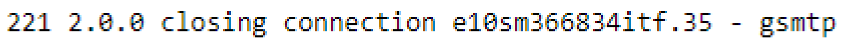
```
Response after sending message body:250 2.0.0 OK 1507256776 e10sm366834itf.35 - gsmtp
```

```
# Fill in end

# Message ends with a single period.
# Fill in start
```



```
Response after sending message body and period:250 2.0.0 OK 1507340482 k76sm1612162ita.4 - gsmtp
```

```
# Fill in end

# Send QUIT command and get server response.
# Fill in start
```



```
221 2.0.0 closing connection e10sm366834itf.35 - gsmtp
```

```
# Fill in end
```