Prof. Athina Markopoulou
Spring 2016

Intro to Computer Networks
EECS148/CompSci132

Homework 2 Solution

Topics: Application Layer, Socket Programming.

1. *(50 Points)* **HTTP Web server and Mail Client** Sample solution code to the basic part of the assignment are posted on class website for your reference.

   (a) Web server: `https://eee.uci.edu/16s/18105/hws/web.py`

   (b) Mail client: `https://eee.uci.edu/16s/18105/hws/mail.py`

2. *(20 Points)* **Wireshark Lab:HTTP**

   Part1: Basic HTTP GET/Response

   1-Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?Answer: Both are running HTTP 1.1

   2. What languages (if any) does your browser indicate that it can accept to the server? Answer: Accept-Language: en-us, en

   3. What is the IP address of your computer? Of the gaia.cs.umass.edu server? Answer: My IP address is 192.168.1.46 and the server?s is 128.119.245.12

   4. What is the status code returned from the server to your browser? Answer: HTTP/1.1 200 OK (text/html)

   5. When was the HTML file that you are retrieving last modified at the server? Answer: Last-Modified: Thu, 07 Jun 2007 22:09:01 GMT

   6. How many bytes of content are being returned to your browser? Answer: Content-Length: 126

   7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one. Answer: No all of the headers can be found in the raw data.

   Part 4: HTML

   16. How many HTTP GET request messages were sent by your browser? To which Internet addresses were these GET requests sent?

```
No.     Time          Source          Destination     Protocol Info
4       0.048291      192.168.1.46    128.119.245.12  HTTP   GET /wireshark-
labs/HTTP-wireshark-file1.html HTTP/1.1

Hypertext Transfer Protocol
    GET /wireshark-labs/HTTP-wireshark-file1.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:1.8.1.4)
Gecko/20070515 Firefox/2.0.0.4\r\n
    Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8
,image/png,*/*;q=0.5\r\n
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    Cookie: MintUnique=1;
__utmz=198765611.1176212581.8.2.utmccn=(referral)|utmcsr=cs.umass.edu|utmcct=/c
sinfo/news.html|utmcmd=referral;
__utma=198765611.821901841.1145892528.1176212581.1179945703.9;
__utma=267820956.1666738513.1163587262.118
    If-Modified-Since: Thu, 07 Jun 2007 22:07:01 GMT\r\n
    If-None-Match: "d6c95-7e-224fab40"\r\n
    \r\n
```

Figure 1: Example GET

Answer: As you can see from the above screenshot there were 3 HTTP GET requests sent to the following Internet addresses:(This number sometimes depend on your browser)

a. 128.119.245.12

b. 128.119.240.90

c. 165.193.123.218

17. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.

Answer: Looking at the connection open time (SYN, SYN/ACK, ACK packet time stamp) and the connection close time (FIN, FIN/ACK, ACK packets) of TCP connections established for downloading the images we can see if the files were downloaded serially or in parallel. In this case the two images were downloaded in parallel.

NOTE: In order to map TCP connections to to the images being downloaded, we need to look at the HTTP GET requests.

```
No.     Time        Source          Destination  Protocol Info
6       0.155044    128.119.245.12 192.168.1.46 HTTP     HTTP/1.1 200 OK
(text/html)

Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Date: Thu, 07 Jun 2007 22:09:08 GMT\r\n
    Server: Apache/2.0.52 (CentOS)\r\n
    Last-Modified: Thu, 07 Jun 2007 22:09:01 GMT\r\n
    ETag: "d6c95-7e-2976b940"\r\n
    Accept-Ranges: bytes\r\n
    Content-Length: 126
    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
Line-based text data: text/html
    <html>\n
    Congratulations.  You've downloaded the file \n
    http://gaia.cs.umass.edu/ethereal-labs/HTTP-ethereal-file1.html!\n
    </html>\n
```

Figure 2: Example Response

3. *(20 Points)* **HTTP)** (from last year's midterm, and similar to Pr. 7-8 from Ch.2)

   You open your browser and you click on a URL. Assume that the webpage associated with the URL consists of a small HTML file and 2 small images, all stored on the same webserver. Let RTT_{cs} be the RTT from the client host to the webserver containing the URL. The IP address of the webserver is cached at your local DNS server. Let RTT_{DNS} be the RTT from the client host to the local DNS server.

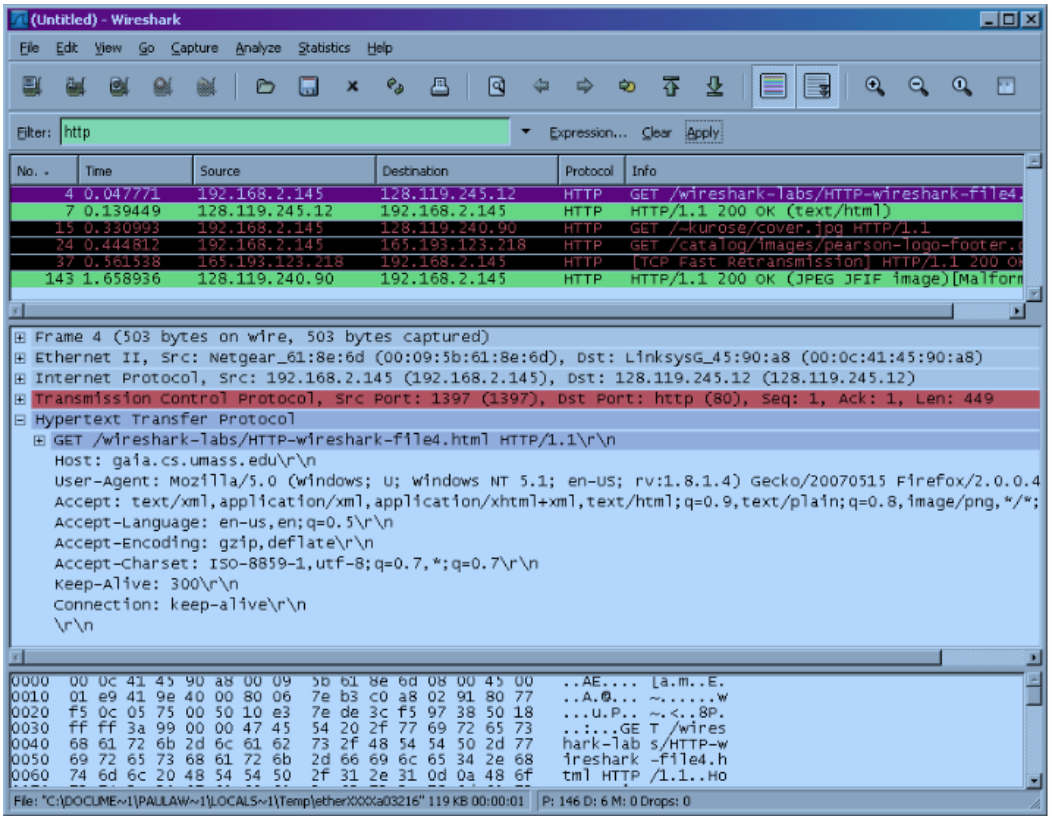
   What is the total delay from the time you click on the webpage until the time that the entire webpage is displayed on your browser? Take into account *all* messages sent/received by the client host, including DNS, HTTP and TCP-related messages. Consider two cases:

   (a) Persistent HTTP without pipelining.

   (b) Non-persistent HTTP with 2 parallel connections.

   Solution:

   The following preliminary steps are required:



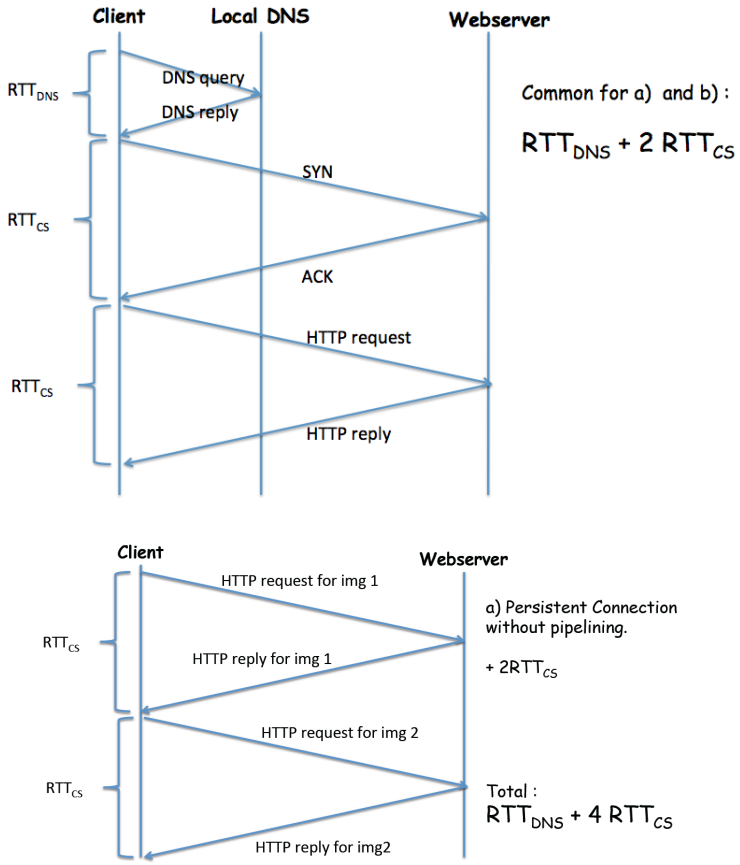
- First, we need to obtain the IP address of the webserver from the local DNS server. This takes RTT_{DNS} time.
- Second, we need to establish a TCP connection with the server. There are three messages in a TCP handshake: SYN, ACK and SYNACK. The first two take RTT_{CS} time. The third one is also used to send the HTTP request and is accounted for below.
- Third, we send the HTTP request for the HTML file and we get the request. This takes RTT_{CS} time.

Note: since all messages and objects are "small" we can neglect the transmission delays and focus on RTTs. Those of you who explicitly considered the transmission bandwidth are not penalized, though.

*Note:*Also, TCP disconnect is not considered as it is yet to be explained in the lecture. If the disconnect is mentioned in your answer that would be acceptable.

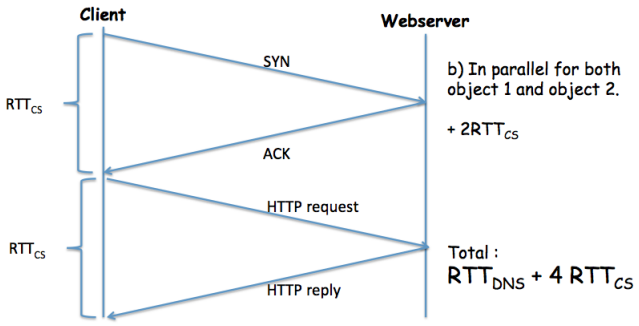
The part above are common to (a) and (b). The differentiation starts here.

(a) In the persistent connection, the existing TCP connection is used, no need to



open another. Since the requests are not pipelined, then getting the first and the second object should take $2 \cdot RTT_{CS}$ time, as shown in the figure,

(b) In the non-persistent connection, the connection needs to be re-established for each of the two objects (relevant part in your book is section 2.2.2 on non-persistent connections). The two connections are taking place *in parallel* and each of them looks as in figure (b) below. Therefore the time required to get both objects is $2 \cdot RTT_{CS}$.



4. *(10 Points)* **More DNS:** Problem 18

Solution:

Note: You are strongly advised to familiarize yourselves with the arguments and options of the commands `whois, nslookup` before you start this exercise.

(a) For a given input of domain name (such as cnn.com), IP address or network administrator name, the *whois* database can be used to locate the corresponding registrar, whois server, DNS server, and so on. You can use the `whois` or `nslookup` commands.

(b) The authoritative name servers for *uci.edu* and *google.com* stored in the corresponding TLDs for .edu (Educause) and .com (Verisign). They can be learnt by going to *whois.com* or by quering the whois databases (e.g., "`whois uci.edu`" and "`whois google.com`"). Or you can use the command line `nslookup` as shown next:

```
dhcp-v009-159:~ athina$ nslookup -type=NS uci.edu
Server: 68.105.28.11
Address: 68.105.28.11#53

Non-authoritative answer:
uci.edu nameserver = ns4.service.uci.edu.
uci.edu nameserver = ns5.service.uci.edu.
```

So, two authoritative name servers for uci.edu are ns4.service.uci.edu and ns5.service.uci.edu.

```
dhcp-v009-159:~ athina$ nslookup -type=NS google.com
Server: 68.105.28.11
Address: 68.105.28.11#53
```

```
Non-authoritative answer:
google.com nameserver = ns3.google.com.
google.com nameserver = ns2.google.com.
google.com nameserver = ns1.google.com.
google.com nameserver = ns4.google.com.
```

So, four authoritative name servers for google.com are {ns1, n2, ns3, ns4}.google.com.

(c) Now that we know the authoritative nameservers for the uci.edu and google.com domains we can send our DNS queries directly to them (by adding them at the end of the command line, see `man nslookup`). Let's pick `ns4.service.uci.edu` from part (b), send it type A,MX and NS queries, and see the DNS response.

Example of type A query sent to `ns4.service.uci.edu`:

```
dhcp-v009-159:~ athina$ nslookup -type=A www.uci.edu ns4.service.uci.edu
Server:         ns4.service.uci.edu
Address:        128.200.59.190#53

Name: www.uci.edu
Address: 128.195.188.232
```

Example of type NS query sent to `ns4.service.uci.edu`:

```
dhcp-v009-159:~ athina$ nslookup -type=NS uci.edu ns4.service.uci.edu
Server:         ns4.service.uci.edu
Address:         128.200.59.190#53

uci.edu nameserver = ns5.service.uci.edu.
uci.edu nameserver = ns4.service.uci.edu.
```

Example of type MX query sent to `ns4.service.uci.edu`:

```
dhcp-v009-159:~ athina$ nslookup -type=MX uci.edu ns4.service.uci.edu
Server:         ns4.service.uci.edu
Address:        128.200.59.190#53

uci.edu mail exchanger = 10 mta.service.uci.edu.
```

We omit requests to the google DNS servers, which can be sent similarly. Also, if you are on campus, your local DNS server is `ns4.service.uci.edu`, thus omitted; if you are working from home, your local DNS server is different from the UCI name servers.

(d) Unsurprisingly, `www.google.com` has multiple IP addresses:

```
dhcp-v009-159:~ athina$ nslookup -q=any www.google.com
Server: 68.105.28.11
Address:        68.105.28.11#53

Non-authoritative answer:
www.google.com has AAAA address 2607:f8b0:4002:c0c::6a
Name: www.google.com
Address: 74.125.138.147
Name: www.google.com
Address: 74.125.138.104
Name: www.google.com
Address: 74.125.138.103
Name: www.google.com
Address: 74.125.138.106
Name: www.google.com
Address: 74.125.138.105
Name: www.google.com
Address: 74.125.138.99
```

In contrast, UCI seems to have only one IP address. Based on the first query in part (c), you can see that `www.uci.edu` has one IP address: `128.195.188.232`. However, sometimes the DNS servers return only one address, even if they have stored multiple. Even if we repeat our query, or ask for more details, it seems that www.uci.edu has only one IP address:

```
dhcp-v009-159:~ athina$ nslookup -q=any www.uci.edu ns4.service.uci.edu
Server: ns4.service.uci.edu
Address:        128.200.59.190#53

Name: www.uci.edu
Address: 128.195.188.232
```

(e) You can go to the ARIN website `https://whois.arin.net/rest/org/UCI-1.html` or just type `whois -h whois.arin.net n 128.195.188.231`:

```
NetRange: 128.195.0.0 - 128.195.255.25
CIDR:       128.195.0.0/16
OriginAS:   AS299
NetName: UCI-NET
...
```

The IP addresses assigned to UCI are **NetRange: 128.195.0.0 - 128.195.255.25**.