SYSC 5201 2019 Fall Assignment 2 Solution

1. (Textbook problem P6 in page 174) Obtain the HTTP/1.1 specification (RFC 2616). Answer the following questions:
    a. Explain the mechanism used for signaling between the client and server to indicate that a persistent connection is being closed. Can the client, the server, or both signal the close of a connection?
    b. What encryption services are provided by HTTP?
    c. Can a client open three or more simultaneous connections with a given server?
    d. Either a server or a client may close a transport connection between them if either one detects the connection has been idle for some time. Is it possible that one side starts closing a connection while the other side is transmitting data via this connection? Explain.

    Solution:

    a) Persistent connections are discussed in section 8 of RFC 2616 (the real goal of this question was to get you to retrieve and read an RFC). Sections 8.1.2 and 8.1.2.1 of the RFC indicate that either the client or the server can indicate to the other that it is going to close the persistent connection. It does so by including the connection-token "close" in the Connection-header field of the http request/reply.
    b) HTTP does not provide any encryption services.
    c) (From RFC 2616) “Clients that use persistent connections should limit the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD NOT maintain more than 2 connections with any server or proxy.”
    d) Yes. (From RFC 2616) “A client might have started to send a new request at the same time that the server has decided to close the "idle" connection. From the server's point of view, the connection is being closed while it was idle, but from the client's point of view, a request is in progress.”
2. (Textbook problem P4 in page 173) Consider the following string of ASCII characters that were captured by Wireshark when the browser sent an HTTP GET message (i.e., this is the actual content of an HTTP GET message). The characters *<cr><lf>* are carriage return and line-feed characters (that is, the italized character string *<cr>* in the text below represents the single carriage-return character that was contained at that point in the HTTP header). Answer the following questions, indicating where in the HTTP GET message below you find the answer.

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (
Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec
ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex
t/xml, application/xml, application/xhtml+xml, text
/html;q=0.9, text/plain;q=0.8, image/png,*/*;q=0.5
<cr><lf>Accept-Language: en-us,
en;q=0.5<cr><lf>Accept-
Encoding: zip, deflate<cr><lf>Accept-Charset: ISO
-8859-1, utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive:
300<cr>
<lf>Connection:keep-alive<cr><lf><cr><lf>
```

a. What is the URL of the document requested by the browser?
b. What version of HTTP is the browser running?
c. Does the browser request a non-persistent or a persistent connection?
d. What is the IP address of the host on which the browser is running?
e. What type of browser initiates this message?
f. Why is the browser type needed in an HTTP request message?

Solution:

a) The document request was http://gaia.cs.umass.edu/cs453/index.html. The Host : field indicates the server's name and /cs453/index.html indicates the file name.
b) The browser is running HTTP version 1.1, as indicated just before the first <cr><lf> pair.
c) The browser is requesting a persistent connection, as indicated by the Connection: keep-alive.
d) This is a trick question. This information is not contained in an HTTP message anywhere. So there is no way to tell this from looking at the exchange of HTTP messages alone. One would need information from the IP datagrams (that carried the TCP segment that carried the HTTP GET request) to answer this question.
e) Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax)
f) The browser type information is needed by the server to send different versions of the same object to different types of browsers.

3. Consider a short, 10-meter link, over which a sender can transmit at a rate of 150 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or handshaking) are 200 bits long. Assume that N parallel connections each get 1/N of the link bandwidth. Now consider the HTTP protocol, and suppose that each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer.

Solution:
Note that each downloaded object can be completely put into one data packet. Let Tp denote the one-way propagation delay between the client and the server.

First consider parallel downloads using non-persistent connections. Parallel downloads would allow 10 connections to share the 150 bits/sec bandwidth, giving each just 15 bits/sec. Thus, the total time needed to receive all objects is given by:

(200/150+*T*p + 200/150 +*T*p + 200/150+*T*p + 100,000/150+ *T*p)
+ (200/(150/10)+*T*p + 200/(150/10) +*T*p + 200/(150/10)+*T*p + 100,000/(150/10)+ *T*p)
= 7377 + 8**T*p (seconds)

Now consider a persistent HTTP connection. Assuming pipelining approach. The total time needed is given by:

(200/150+*T*p + 200/150 +*T*p + 200/150+*T*p + 100,000/150+ *T*p)
+ (200/(150/10)+*T*p + 100,000/(150/10)+ *T*p)
=7351 + 6**T*p (seconds)

Assuming the speed of light is $300*10^6$ m/sec, then Tp=10/($300*10^6$)=0.03 microsec. Tp is therefore negligible compared with transmission delay.

Thus, we see that persistent HTTP is not significantly faster (less than 1 percent) than the non-persistent case with parallel download.

4. (Textbook problem P18 in page 178)
   a. What is a *whois* database?
   b. Use various whois databases on the Internet to obtain the names of two DNS servers. Indicate which whois database you used.
   c. Use nslookup on your local host to send DNS queries to three DNS servers: your local DNS server and the two DNS servers you found in (b). Try query for type A, NS, and MX reports. Summarize your findings.
   d. Use nslookup to find a Web server that has multiple IP addresses. Does the web server of Carleton University have multiple IP addresses?
   e. Use the ARIN whois database to determine the IP address range used by Carleton University
   f. Use the CIRA whois to determine the DNS servers used by Carleton Unievrsity
   g. Describe how an attacker can use whois databse and the nslookup tool to perform reconnaissance on an institution before launching an attack.
   h. Discuss why whois databases should be publicly available.

Solution: Results will vary from student to student.

a) For a given input of domain name (such as ccn.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.

b) NS4.YAHOO.COM from www.register.com; NS1.MSFT.NET from ww.register.com

c) *Local Domain: www.mindspring.com*

Web servers : www.mindspring.com
207.69.189.21, 207.69.189.22,
207.69.189.23, 207.69.189.24,
207.69.189.25, 207.69.189.26, 207.69.189.27,
207.69.189.28

Mail Servers : mx1.mindspring.com (207.69.189.217)
mx2.mindspring.com (207.69.189.218)
mx3.mindspring.com (207.69.189.219)
mx4.mindspring.com (207.69.189.220)

Name Servers: itchy.earthlink.net (207.69.188.196)
scratchy.earthlink.net (207.69.188.197)

www.yahoo.com

Web Servers: www.yahoo.com (216.109.112.135, 66.94.234.13)

Mail Servers: a.mx.mail.yahoo.com (209.191.118.103)
b.mx.mail.yahoo.com (66.196.97.250)
c.mx.mail.yahoo.com (68.142.237.182, 216.39.53.3)
d.mx.mail.yahoo.com (216.39.53.2)
e.mx.mail.yahoo.com (216.39.53.1)
f.mx.mail.yahoo.com (209.191.88.247, 68.142.202.247)
g.mx.mail.yahoo.com (209.191.88.239, 206.190.53.191)

Name Servers: ns1.yahoo.com (66.218.71.63)
ns2.yahoo.com (68.142.255.16)
ns3.yahoo.com (217.12.4.104)
ns4.yahoo.com (68.142.196.63)
ns5.yahoo.com (216.109.116.17)
ns8.yahoo.com (202.165.104.22)
ns9.yahoo.com (202.160.176.146)

www.hotmail.com

Web Servers: www.hotmail.com (64.4.33.7, 64.4.32.7)

Mail Servers: mx1.hotmail.com (65.54.245.8, 65.54.244.8, 65.54.244.136)
mx2.hotmail.com (65.54.244.40, 65.54.244.168, 65.54.245.40)
mx3.hotmail.com (65.54.244.72, 65.54.244.200, 65.54.245.72)
mx4.hotmail.com (65.54.244.232, 65.54.245.104, 65.54.244.104)

Name Servers: ns1.msft.net (207.68.160.190)
ns2.msft.net (65.54.240.126)
ns3.msft.net (213.199.161.77)
ns4.msft.net (207.46.66.126)
ns5.msft.net (65.55.238.126)

d) The yahoo web server has multiple IP addresses

www.yahoo.com (206.190.36.45, 206.190.36.105)
Carleton web server has a single IP address
www.carleton.ca (134.117.6.162)

e) The address range for Carleton University: 134.117.0.0 – 134.117.255.255

f) The DNS servers of Carleton University:
ns1.carleton.ca
ns2.carleton.ca
ns1.d-zone.ca
ns2.d-zone.ca

g) An attacker can use the *whois* database and nslookup tool to determine the IP address ranges, DNS server addresses, etc., for the target institution.

h) By analyzing the source address of attack packets, the victim can use whois to obtain information about domain from which the attack is coming and possibly inform the administrators of the origin domain.

5. Suppose that your department has a local DNS server for all computers in the department. You are an ordinary user (i.e., not a network/system administrator). Can you determine if an external Web site was likely accessed froma computer in your department a coupkle seconds ago? Explain.
Solution:
Yes, we can use dig (in Unix and Linux) to query that Web site in the local DNS server. For example, "dig cnn.com" will return the query time for finding cnn.com. If cnn.com was just accessed a couple of seconds ago, an entry for cnn.com is cached in the local DNS cache, so the query time is 0 msec. Otherwise, the query time is large.

6. Consider distributing a file of *F* bits to *N* peers using a P2P architecture. Assume a fluid model. For simplicity assume that peer download bandwidth is never a bottleneck. Suppose the uploading bandwidth of the server is u_s and the uploading bandwidth of peer *i* is u_i.
   a. Suppose that $u_s \leq (u_s + u_1 + \cdots + u_N)/N$. Specify a distribution scheme that has a distribution time of F/u_s .
   b. Suppose that $u_s \geq (u_s + u_1 + \cdots + u_N)/N$. Specify a distribution scheme that has a distribution time of $NF/(u_s + u_1 + \cdots + u_N)$.
   c. Conclude that the minimum distribution time is in general given by $max\{F/u_s, NF/(u_s + u_1 + \cdots + u_N)\}$

Solution:

a) Define u = u1 + u2 + ….. + uN. By assumption

$u_s <= (u_s + u)/N$ Equation 1

Divide the file into N parts, with the i^{th} part having size $(u_i/u)F$. The server transmits the i^{th} part to peer i at rate $r_i = (u_i/u)u_s$. Note that $r_1 + r_2 + ... + r_N = u_s$, so that the aggregate server rate does not exceed the link rate of the server. Also have each peer i forward the bits it receives to each of the *N-1* peers at rate r_i. The aggregate forwarding rate by peer i is $(N-1)r_i$. We have

$(N-1)r_i = (N-1)(u_s u_i)/u <= u_i$,

where the last inequality follows from Equation 1. Thus the aggregate forwarding rate of peer i is less than its link rate u_i.

In this distribution scheme, peer i receives bits at an aggregate rate of

$$r_i + \sum_{j<>i} r_j = u_s$$

Thus each peer receives the file in F/u_s.

b) Again define $u = u_1 + u_2 + ... + u_N$. By assumption

$u_s >= (u_s + u)/N$ Equation 2

Let $r_i = u_i/(N-1)$ and
$r_{N+1} = (u_s - u/(N-1))/N$

In this distribution scheme, the file is broken into *N+1* parts. The server sends bits from the i^{th} part to the i^{th} peer (i = *1, …., N*) at rate r_i. Each peer i forwards the bits arriving at rate r_i to each of the other N-1 peers. Additionally, the server sends bits from the $(N+1)^{st}$ part at rate r_{N+1} to each of the *N* peers. The peers do not forward the bits from the $(N+1)^{st}$ part.

The aggregate send rate of the server is

$r_1 + ... + r_N + N r_{N+1} = u/(N-1) + u_s - u/(N-1) = u_s$

Thus, the server's send rate does not exceed its link rate. The aggregate send rate of peer i is

$(N-1)r_i = u_i$

Thus, each peer's send rate does not exceed its link rate.
In this distribution scheme, peer i receives bits at an aggregate rate of

$$r_i + r_{N+1} + \sum_{j<>i} rj = u/(N-1) + (u_s - u/(N-1))/N = (u_s + u)/N$$

Thus each peer receives the file in $NF/(u_s+u)$.
(For simplicity, we neglected to specify the size of the file part for i = 1, …., N+1. We now provide that here. Let $\Delta = (u_s+u)/N$ be the distribution time. For i = 1, …, N, the i^{th} file part is F_i

= r_i Δ bits. The (N+1)st file part is $F_{N+1} = r_{N+1}$ Δ bits. It is straightforward to show that F_1+ ….. + F_{N+1} = F.)

c) The solution to this part is similar to that of 17 (c). We know from section 2.6 that

$$D_{P2P} >= max\{F/u_s,\ NF/(u_s + u)\}$$

Combining this with a) and b) gives the desired result.