National University of Singapore
School of Computing
CS2105: Introduction to Computer Networks
Semester 1, 2018/2019

Tutorial 2
Application Layer

These questions will be discussed during the next week's discussion group meetings. Please be prepared to answer these questions during the session in class. Some of the questions are taken from the textbook, so please bring it along for reference.

1. [**Modified from KR, Chapter 2, P4**] 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 `<cr><lf>` line-endings are omitted and replaced with an actual newline for readability.

```
GET /~cs2105/demo.html HTTP/1.1
Host: www.comp.nus.edu.sg
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/39.0.2171.99 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
```

   (a) What is the URL of the document requested by this browser?

   Ans: `www.comp.nus.edu.sg/~cs2105/demo.html`

   (b) What version of HTTP is this browser running?

   Ans: HTTP version 1.1

   (c) Does the browser request a non-persistent or a persistent connection?

   Ans: The browser requests a persistent connection, as indicated by the header field `Connection: keep-alive`.

   (d) What is the IP address of the host on which the browser is running?

   Ans: [Tricky question] IP address is not shown in HTTP message. One would be able to get such information from socket.

   (e) What type of browser initiates this message? Why is the browser type useful in an HTTP request message?

   Ans: A Mozilla-compatible browser. Unfortunately, all browsers today send a user-agent string that includes Mozilla, Chrome, Webkit, and Safari. So it is not possible to distinguish which.

   Trivia: Due to historical reasons of the browser wars, today every browser advertise as Mozilla! (`http://webaim.org/blog/user-agent-string-history/`)

2. [**Modified from KR, Chapter 2, P5**] The text below shows the header of the response message sent from the server in reply to the HTTP GET message in the question above. Answer the following questions.

```
HTTP/1.1 200 OK
Date: Tue, 20 Jan 2015 10:08:12 GMT
Server: Apache/2.4.6 (Unix) OpenSSL/1.0.1h
Accept-Ranges: bytes
Content-Length: 73
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

(a) Was the server able to successfully find the document or not?
Ans: Yes. The status code 200 and the phrase OK indicate that the server was able to locate the document successfully.

(b) What time did the server send the HTTP response message?
Ans: On Tuesday, 20 Jan 2105 10:08:12 GMT, as indicated in the `Date` field.

(c) How many bytes are there in the document being returned?
Ans: 73 bytes, as indicated in the `Content-Length` field.

(d) Did the server agree to a persistent connection?
Ans: Yes. This is indicated in the `Connection: Keep-Alive` field.

3. **[KR, Chapter 2, P1]** True or false?

(a) A user requests a Web page that consists of some text and three images. For this page, the client will send one request message and receive four response messages.
Ans: False. Each web object has to be requested individually.

(b) Two distinct Web pages (for example, `www.mit.edu/research.html` and `www.mit.edu/students.html`) can be sent over the same persistent connection.
Ans: True. HTTP 1.1 can be used to send multiple web objects from the same server.

(c) With non-persistent connections between browser and origin server, it is possible for a single TCP segment to carry two distinct HTTP request messages.
Ans: False. Non-persistent connection will terminate once the web object has been sent.

(d) The `Date:` header in the HTTP response message indicates when the object in the response was last modified.
Ans: False. The header `Date` is the date/time of the server's response. The last modified time of the object is denoted by another header field `Last-modified`.

(e) HTTP response messages never have an empty message body.
Ans: False. An example might be a conditional GET when the browser's cache is up-to-date.

4. **[Modified from KR, Chapter 2, P7]** Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address.

Suppose that $\boldsymbol{n}$ DNS servers are visited before your host receives the IP address from DNS—visiting them incurs an RTT of $\boldsymbol{D}_{dns}$ per DNS server.

Further suppose that the Web page associated with the link contains $\boldsymbol{m}$ very small objects. Suppose the HTTP running is non-persistent and non-parallel. Let $\boldsymbol{D}_{web}$ denote the RTT between the local host and the server for each object.

Assuming zero transmission time of the object, how much time elapses from when the client clicks on the link until the client receives the object?

Ans: Time to get the IP address from the DNS $= n \times D_{dns}$. Note that DNS is over UDP so there is no connection setup time needed.

To establish the TCP connection and retrieve the HTML page $= D_{web} + D_{web}$.

To establish $\boldsymbol{m}$ TCP connections and get all $\boldsymbol{m}$ objects $= m \times (D_{web} + D_{web})$.

Total time $= nD_{dns} + 2(m+1)D_{web}$

5. [**Modified from KR, Chapter 2, P8**] Referring to the previous question, suppose that three DNS servers are visited. Further, the HTML file references five very small objects on the same server. Neglecting transmission delay, how much time elapses with:

   (a) Non-persistent HTTP with no parallel TCP connections?
   Ans: $3D_{dns} + 2(5+1)D_{web}$

   (b) Non-persistent HTTP with the browser configured for five parallel connections?
   Ans: $3D_{dns} + 2D_{web} + 2D_{web}$
   The first $2D_{web}$ is to fetch the HTML file. Subsequently the 5 objects can be fetched in parallel with each using a TCP connection ($2D_{web}$).

   (c) Persistent HTTP with pipelining?
   Ans: $3D_{dns} + 2D_{web} + Dweb$
   Like before, the HTML file needs to be fetched first. The subsequent 5 objects can be fetched through the same TCP connection in parallel. As the objects are small, the transmission time is negligible, and no additional RTT for TCP handshake is needed.

6. What is DNS cache poisoning? Search the web for a real-life occurrence.

   Ans: DNS cache poisoning (a kind of DNS spoofing) is a computer hacking attack, whereby rogue DNS records are introduced into a DNS resolver's cache, causing the name server to return an incorrect IP address, diverting traffic to the attacker's computer (or any other computer). For example, DDoS (Distributed Denial of Service Attack) on a particular machine can be achieved via DNS cache poisoning.

   Examples:

   (a) DNS Poisoning in China: `http://www.howtogeek.com/161808/htg-explains-what-is-dns-cache-poisoning/`

   (b) Angry Bird Website Defaced: `https://arstechnica.com/security/2014/01/angry-birds-website-defaced-following-reports-it-enables-government-spying/`

7. **Introduction to Wireshark**

   Wireshark is a tool for observing the messages exchanged between executing protocol entities. It observes messages being sent and received by applications and protocols running on your computer.

   Download and install the Wireshark software: Go to `http://www.wireshark.org/download.html` to download and install the Wireshark binary for your computer.

8. Let us test our Wireshark installation and begin our exploration of HTTP by downloading a very simple HTML file, and contains no embedded objects.

   (a) Start up your web browser.

   (b) Start up the Wireshark software.

   (c) To begin packet capture, select the **Capture** pull down menu and select Interfaces.

(d) Click on **Start** for the interface on which you want to begin packet capture.

(e) While Wireshark is running, enter the URL: `http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html` and have that page display in your browser.

(f) After your browser has displayed the `INTRO-wireshark-file1.html` page, stop the Wireshark packet capture by selecting **stop** in the Wireshark capture window. You now have live packet data that contains all protocol messages exchanged between your computer and other network entities!

(g) Type in "`http`" (without the quotes, and in lower case—all protocol names are in lower case in Wireshark) into the display filter specification window at the top of the main Wireshark window. Then select **Apply**. This will cause only HTTP message to be displayed in the packet-listing window.

Now answer the following questions:

(a) What is the status code returned from the server to your browser?
Ans: 200

(b) When was the HTML file that you are retrieving last modified at the server?
Ans: the value is denoted by the header field "Last-Modified".