National University of Singapore
School of Computing

CS2105 **Tutorial 2** Answer paper

To students:

Due to time constraint, not all the questions will be discussed in class. Your tutor has the discretion to choose the questions to discuss (or you may request your tutor to discuss certain questions). Please do go through the rest questions after class.

1. Consider the following HTTP request message sent by a browser.

   GET /index.html HTTP/1.1

   Host: www.example.org

   Connection: keep-alive

   User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36

   Accept-Encoding: gzip, deflate

   …

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

      www.example.org/index.html

   b) What version of HTTP is this browser running?

      HTTP version 1.1

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

      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?

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

2. The text below shows the header of the response message sent from the server in reply to the HTTP GET message in Q1 above. Answer the following questions.

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Wed, 23 Jan 2019 13:50:31 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Connection: Keep-Alive
Content-Length: 606
…
```

   a) Was the server able to successfully find the document or not?

      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?

      The HTTP response message was formed on Tuesday, 20 Jan 2015 10:08:12 Greenwich Mean Time.

   c) How many bytes are there in the document being returned?

      There are 73 bytes in the document being returned.

   d) Did the server agree to a persistent connection?

      The server agreed to a persistent connection, as indicated by the header field '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.

      False. Download one object per request.

   b) Two distinct Web pages on the same Web server (for example, www.mit.edu/research.html and www.mit.edu/students.html) can be sent over the same persistent connection.

      True because they are on the same server.

   c) The **Date:** header in the HTTP response message indicates when the object in the response was last modified.

False. Header field 'Date' indicates the server response time. The time the object is last modified is denoted by another header field 'Last-Modified'.

d) HTTP response messages never have an empty message body.

False. E.g. conditional GET whereby browser's cached copy is up-to-date.

4. **[Modified from KR, Chapter 2, P10]** 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 ***n*** DNS servers are visited before your host receives the IP address from DNS; visiting them incurs an RTT of D_{DNS} per DNS server.
Further suppose that the Web page associated with the link contains ***m*** very small objects (in addition to the HTML page). Suppose the HTTP running is non-persistent and non-parallel. Let D_{Web} denote the RTT between the local host and the server of each object.
Assuming zero transmission time of each object, how much time elapses from when the client clicks on the link until the client receives all the objects?

To map from hostname to IP address: n x D_{DNS} (note: DNS is over UDP, so no need to establish connection).

To establish TCP connection and get the HTML page = $D_{Web} + D_{Web}$

To establish *m* TCP connection and get all *m* objects = m x ($D_{Web} + D_{Web}$)

Total time = n x D_{DNS} + (m + 1) x 2 x 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?

3 x D_{DNS} + (5 + 1) x 2 x D_{Web}

b) Non-persistent HTTP with the browser configured for five parallel connections?

3 x D_{DNS} + 2 x D_{Web} + 2 x D_{Web}

Need to fetch HTML file first (2 x D_{Web}). Subsequently the rest 5 objects can be fetched in parallel each using a TCP connection (2 x D_{Web}).

c) Persistent HTTP with pipelining?

3 x D_{DNS} + 2 x D_{Web} + D_{Web}

Need to fetch HTML file first (2 x D_{Web}). The rest 5 objects can be fetched through the same TCP connection in parallel – no RTT for TCP handshake is needed.

6. Do you know what is DNS cache poisoning? Search online for a real example.

   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:

   1. **DNS Poisoning in China: http://www.howtogeek.com/161808/htg-explains-what-is-dns-cache-poisoning/**
   2. **Angry Bird Website Defaced: https://arstechnica.com/security/2014/01/angry-birds-website-defaced-following-reports-it-enables-government-spying/**

7. Wireshark Introduction
   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 and download and install the Wireshark binary for your computer.

   Taking Wireshark for a test run:
   1. Start up your web browser.
   2. Start up the Wireshark software.
   3. To begin packet capture, select the Capture pull down menu and select Interfaces.
   4. Click on Start for the interface on which you want to begin packet capture.
   5. While Wireshark is running, enter the URL: http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html and have that page displayed in your browser.
   6. After your browser has displayed the INTRO-wireshark-file1.html page, stop 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!

7. 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.

Congratulations! You’ve now completed the Wireshark introduction.

8. Wireshark: HTTP GET/response interaction

Let’s begin our exploration of HTTP by downloading a very simple HTML file, and contains no embedded objects. Do the following:

1. Start up your web browser.
2. Start up the Wireshark packet sniffer. Enter “http” in the display-filter-specification window and begin Wireshark packet capture.
3. Enter the following to your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html.
4. Stop Wireshark packet capture.

Now answer the following questions:

1. What is the status code returned from the server to your browser?
   Ans: 200
2. 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’.