

CS2105

Tutorial 2

Question 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

Question 1

- What is the URL of the document requested by this browser?

www.example.org/index.html

- What version of HTTP is this browser running?

HTTP version 1.1

- 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’.

- 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 (to be learnt later).

Question 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

Question 2

- 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.

- What time did the server send the HTTP response message?

The HTTP response message was formed on Wed, 23 Jan 2019 13:50:31 Greenwich Mean Time.

- How many bytes are there in the document being returned?

There are 606 bytes in the document being returned.

- 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'.

Question 3: 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 (for example, www.mit.edu/research.html and www.mit.edu/students.html) can be sent over the same persistent connection.

True. They are on the same server.

Question 3: True or false?

- 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’.

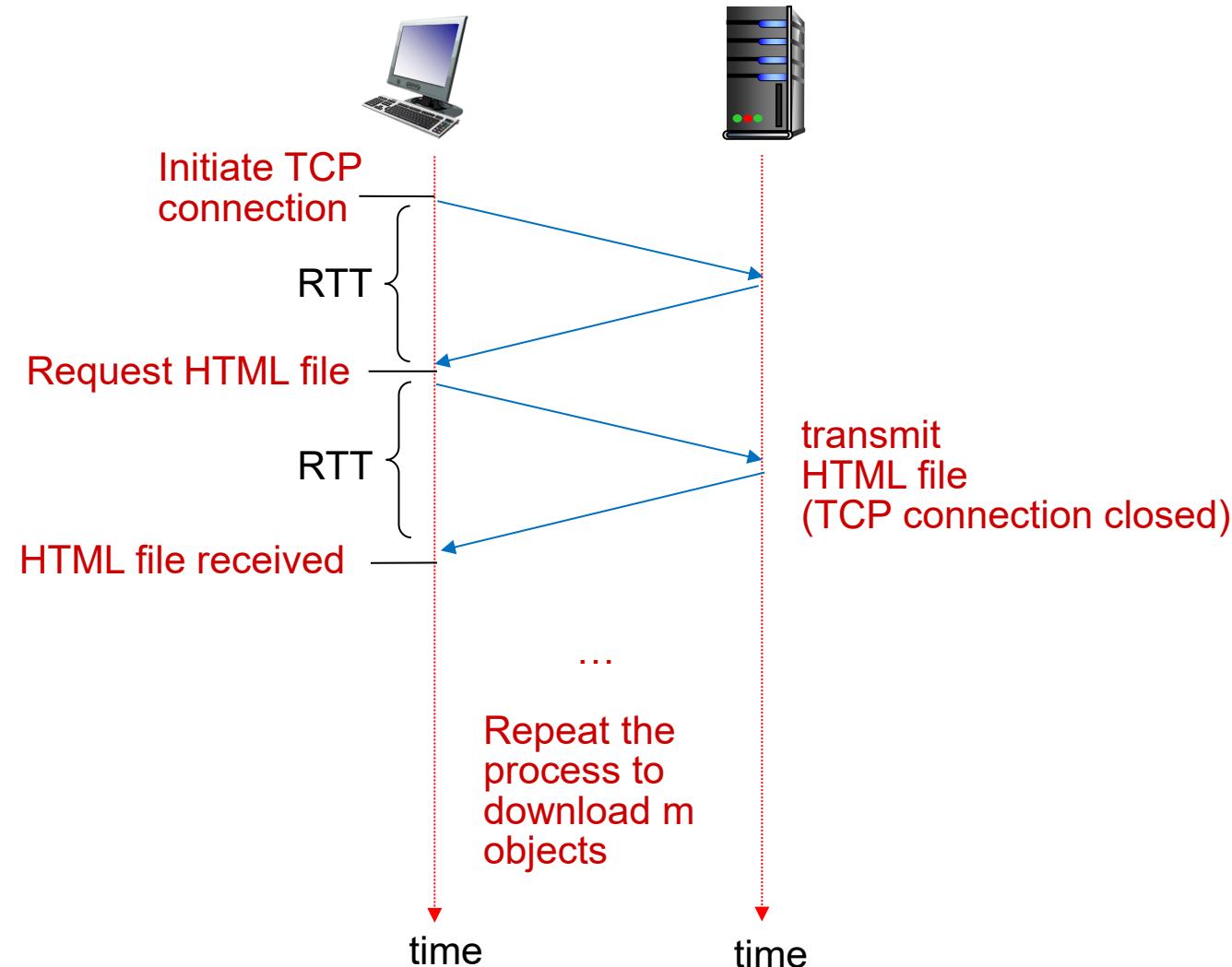
- d) HTTP response messages never have an empty message body.

False. E.g., conditional GET where browser’s cached copy is up-to-date.

Question 4

- 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?

Non-Persistent and Non-Parallel connection



Question 4 Workflow

1. Map Web server hostname to IP address (by visiting n DNS servers)
2. Establish TCP connection to the Web server
3. Obtain the HTML page
4. For each of the m objects -> establish connection + get object

Question 4 Calculation

- To map Web server hostname to IP address= $n \times D_{DNS}$
(note: DNS is over UDP, so no need to establish connection).
- To establish TCP connection and get the HTML page = $D_{Web} + D_{Web}$
(2 RTT)
- To establish m TCP connection and get all m objects = $m \times (D_{Web} + D_{Web}) = m \times 2 \times D_{Web}$
- Total time = $n \times D_{DNS} + 2 \times D_{Web} + m \times 2 \times D_{Web} = n \times D_{DNS} + (m + 1) \times 2 \times D_{Web}$

Question 5

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:

Question 5

a) Non-persistent HTTP with no parallel TCP connections?

$$3 \times D_{DNS} + (1 + 5) \times 2 \times D_{Web}$$

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

$$3 \times D_{DNS} + 2 \times D_{Web} + 2 \times D_{Web}$$

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

c) Persistent HTTP with pipelining?

$$3 \times D_{DNS} + 2 \times D_{Web} + D_{Web}$$

Need to fetch HTML file first ($2 \times D_{Web}$). The rest 5 objects can be fetched through the same TCP connection – no RTT needed to establish second TCP connection.

Question 6

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.

Question 6

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.
and have that page displayed in your browser.
 - <http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html>

Question 6

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.

Question 7

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.

Question 7

Now answer the following questions:

1. What is the status code returned from the server to your browser?

200

2. When was the HTML file that you are retrieving last modified at the server?

The value is denoted by the header field ‘Last - Modified’.

[Homework] Question 8

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

Question 6

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