CS536 Homework2(Spring 23)

Kazi Samin Mubasshir
Email: kmubassh@purdue.edu
PUID: 34674350

February 10, 2023

1 Answer to Q1

(a) In a non-persistent HTTP/1.1 connection without parallel TCP connections, the client sends a request for each object, waits for the response, and closes the connection after each response is received. This means that the RTT must be added to the time it takes to fetch each file for each request and response, so the total time it takes to fetch a file becomes:

$$T_k = \frac{2^{10-k} \cdot 8}{R} + 2 \cdot rtt$$

Note: 1 Byte = 8 bit

The total time it takes to fetch all 10 files is then given by:

$$T = T_{html} + T_1 + T_2 + \cdots + T_{10}$$

Where T_{html} is the time to fetch the base html file. $T_{html} = 2 * rtt$ as the size of the file is negligible.

$$T = 2 \cdot rtt + \frac{2^{10-1} \cdot 8}{R} + 2 \cdot rtt + \frac{2^{10-2} \cdot 8}{R} + 2 \cdot rtt + \cdots + \frac{2^{10-10} \cdot 8}{R} + 2 \cdot rtt$$

$$T = (2^0 + 2^1 + \cdots + 2^9) \cdot \frac{8}{R} + 22 \cdot rtt$$

$$T = \frac{2^{10} - 1}{2 - 1} \cdot \frac{8}{100} + 22 \cdot 1$$

$$T = 1023 \cdot \frac{8}{100} + 22$$

$$T = 81.84 + 22$$

$$T = 103.84 \ seconds$$

This is the total delay for a non-persistent HTTP/1.1 connection without parallel TCP connections, which adds up the time it takes to fetch each file and the RTT for each request and response.

(b) In a persistent HTTP/1.1 connection without pipelining, the client opens a single connection to the server, sends request for the objects, and waits for the responses in the order that they are received. So there will be as little as one rtt for all the objects.

Let's call the time to establish the connection t_c, t_{html} is the time to fetch the base html file and t_{req} is the time required for the client to send request for all the objects. Then the total time it takes to fetch all 10 files is given by:

$$T = t_c + t_{html} + t_{req} + T_1 + T_2 + ... + T_{10}$$

where T_k is the time it takes to fetch each file as calculated before:

$$T_k = \frac{2^{10-k} \cdot 8}{R}$$

The total time it takes to fetch all 10 files is:

$$T = rtt + rtt + rtt + \frac{2^{10-1} \cdot 8}{R} + \frac{2^{10-2} \cdot 8}{R} + \cdots + \frac{2^{10-10} \cdot 8}{R}$$

$$T = (2^0 + 2^1 + \cdots + 2^9) \cdot \frac{8}{R} + 3 \cdot rtt$$

$$T = \frac{2^{10} - 1}{2 - 1} \cdot \frac{8}{100} + 3 \cdot 1$$

$$T = 1023 \cdot \frac{8}{100} + 3$$

$$T = 81.84 + 3$$

$$T = 84.84 \; seconds$$

(c) When the client uses HTTP/2.0 to fetch the objects, the total time it takes to fetch all the objects is given by:

$$T = t_c + t_{html} + t_{req} + T_{object\_frames}$$

Where t_c is the time it takes to establish the connection, t_{html} is the time to fetch the base html file, t_{req} is the time to send requests for the objects and T_{object_frames} is the time it takes to receive all the frames of the objects. Here, number of frames per object are

$$frames_k = \frac{Object_{size}}{frame_{size}}$$

Here, total number of frames are

$$total\_frames = \frac{2^{10-1}}{1} + \frac{2^{10-2}}{1} + \cdots + \frac{2^{10-10}}{1}$$

$$total\_frames = 1023$$

Time to send frame, $t_{frame} = \frac{1 \cdot 8}{100}$

The time it takes to get all the objects:

$$T = t_c + t_{html} + t_{req} + T_{object\_frames}$$

$$T = rtt + rtt + rtt + total\_frames \cdot t_{frame}$$

$$T = 1 + 1 + 1 + 1023 \cdot \frac{1 \cdot 8}{100}$$

$$T = 3 + 81.84$$

$$T = 84.84 \ seconds$$

The time it takes to receive the first object: Num of frames of object 10 is 1. So it will be received first after the first round of the round robin. So total 10 frames needs to be received for the 10^{th} object to be receieved(1 frame of each object).

$$T = rtt + rtt + rtt + total\_frames \cdot t_{frame}$$

$$T = 1 + 1 + 1 + 10 \cdot \frac{1 * 8}{100}$$

$$T = 3 + 10 \cdot \frac{1 * 8}{100}$$

$$T = 3 + 0.8$$

$$T = 3.8 \ seconds$$

(d) To avoid Head-of-Line (HOL) blocking in HTTP, multiple parallel TCP connections can be used. By using multiple connections, multiple requests can be in-flight simultaneously, reducing the impact of HOL blocking. This will use multiple parallel connections to fetch different parts of a single resource or multiple resources in parallel, thus increasing the overall throughput.

One example of this technique is the HTTP/1.1 Pipelining feature, where multiple requests can be sent over a single connection without waiting for the corresponding responses. This can reduce the overhead of establishing and closing multiple connections, as well as reduce the impact of HOL blocking. However, HTTP/1.1 Pipelining is not widely used due to implementation and deployment issues, as well as the potential for misordering and performance degradation in certain network environments.

Another alternative can be using a Keep-Alive feature, which will allow multiple requests to be sent over a single connection, but with the added benefit of being able to detect and recover from errors more easily. The Keep-Alive feature can be an effective way to reduce the overhead of establishing and closing multiple connections while avoiding HOL blocking.

2 Answer to Q2

Here's a step-by-step description of how Alice's email reaches Bob:

1. Microsoft Outlook sends the email to the mail server of purdue.edu using the **SMTP** protocol.
2. Mail server of purdue.edu transfers the email to the mail server of gmail.com using the **SMTP** protocol.
3. The mail server of gmail.com stores the email in Bob's mailbox.
4. Bob uses a web browser to access his Gmail account.
5. The web browser retrieves the email from Bob's mailbox using the **HTTP** protocol.

So, in summary, the email is sent using the **SMTP** protocol and retrieved using the **HTTP** protocol.

3 Answer to Q3

(a) The IP address of the machine www.cs.purdue.edu is **128.10.19.120**

(b) If I enter the IP address **128.10.19.120** into a web browser, I can see the homepage of www.cs.purdue.edu. This is because it is a web server, so it returns a webpage which I can see in the browser.

(c) The IP address of the machine data.cs.purdue.edu is **128.10.2.13**. If I enter the IP address 128.10.2.13 into a web browser, I can not see a webpage data.cs.purdue.edu. This is because it is not a web server, so it does not return a webpage which I can see in the browser.

(d) Authoritative name server for cs.purdue.edu
Name: pendragon.cs.purdue.edu
IP Address: 128.10.2.5

(e) I can use the command line tool "nslookup" to find the mail server for alice@purdue.edu. Here's an example:

```
nslookup -type=MX purdue.edu
```

This will return the mail exchange (MX) record for the domain purdue.edu, which includes the hostname of the mail server for that domain.