THE UNIVERSITY OF SYDNEY

School of Computer Science

COMP5347: Web Application Development
Week 4 Tutorial: HTTP and Browser Performance

Learning Objectives

- Understand HTTP Protocol
- Understand browser behavior with respect to requesting resources and rendering process

Task 1: Examine the HTTP protocol using TELNET/SSH

Start PUTTY and set up a SSH session to `ucpu1.ug.cs.usyd.edu.au` (alternatively, you may use `ucpu3.ug.cs.usyd.edu.au`) using your unikey and password.

a) In the SSH window type the following command:

```
telnet soit-usrweb-pro-1.ucc.usyd.edu.au 80
GET /~cshe6391/week4.html HTTP/1.0
```

You need to type `Enter` key an extra time after the `GET` command to send out the request. This will send out an HTTP 1.0 request to the server `soit-usrweb-pro-1.ucc.usyd.edu.au` Examine the response message and identify the following fields from the status lines:

Protocol Version	HTTP1.1
Status Code	200
Status Message	OK

Also Identify the following fields from the header line:

Web Server	Apache/2.2.15 (Red Hat)
ETag	2c8003c-16c-5680ed05cab7d
Content Type	text/html; charset=UTF-8
TCP connection closed?	yes

b) Now type the following command to send out an HTTP/1.1 request

```
telnet soit-usrweb-pro-1.ucc.usyd.edu.au 80
GET /~cshe6391/week4.html HTTP/1.1
```

```
Host: soit-usrweb-pro-1.ucc.usyd.edu.au
```

Remember to type `Enter` key twice to send out the request. You may noticed that in the first request, the connection is closed immediately, while in the second one, the connection is still open and you are able to type in another request, that is, if your typing speed is really fast. The connection will be closed after some timeout period.

c) Now type the following command to send out a conditional request

```
GET /~cshe6391/week4.html HTTP/1.1
Host: soit-usrweb-pro-1.ucc.usyd.edu.au
If-None-Match: "2c8003c-16c-5680ed05cab7d"
```

Read the response message and identify the status code. You should expect an "HTTP/1.1 304 Not Modified" status code and message. The response should not contain any body.

Task 2: Inspect browser HTTP request/response details with DevTool

a) Start Google Chrome and open the developer tool same as you did in week 3 lab. In this week, we focus on the `Network` tab instead. Type the following URL in the address bar:

`http://soit-usrweb-pro-1.ucc.usyd.edu.au/~cshe6391/week4.html` and type enter to send out the request. You should get a screen similar to the Figure 1

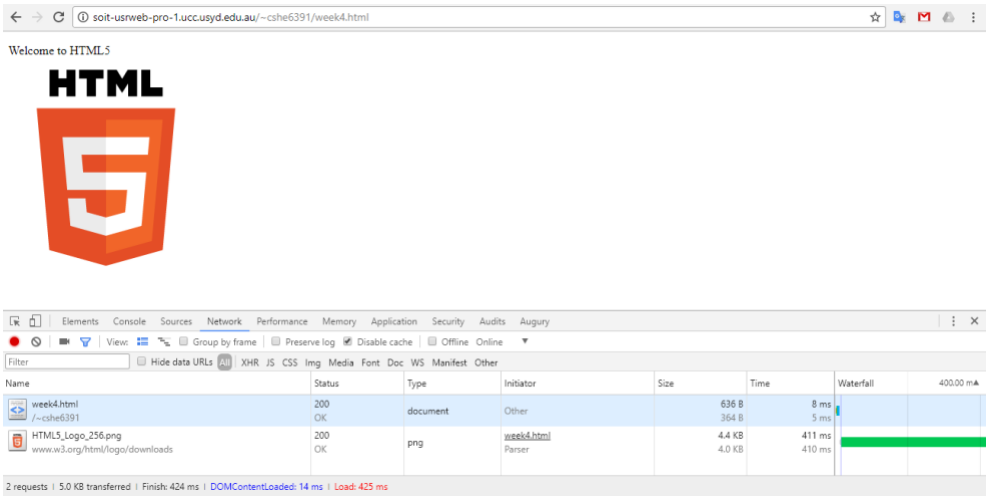

Figure 1: Screenshot of Network tab in DevTool

The network panel provides a number of views. Figure 1 does not show overview and use "small request rows" option. You may switch between different views by clicking the two option icons after the `View:` item.

You can inspect various details of each individual request by clicking the request and selecting what you want to view. For instance, Figure 2 shows the "timing" of request `week4.html`, Figure 3 shows the headers of request `HTML5 Logo 256.png`

Use the information provided by DevTool to answer the following question regarding HTTP messages:

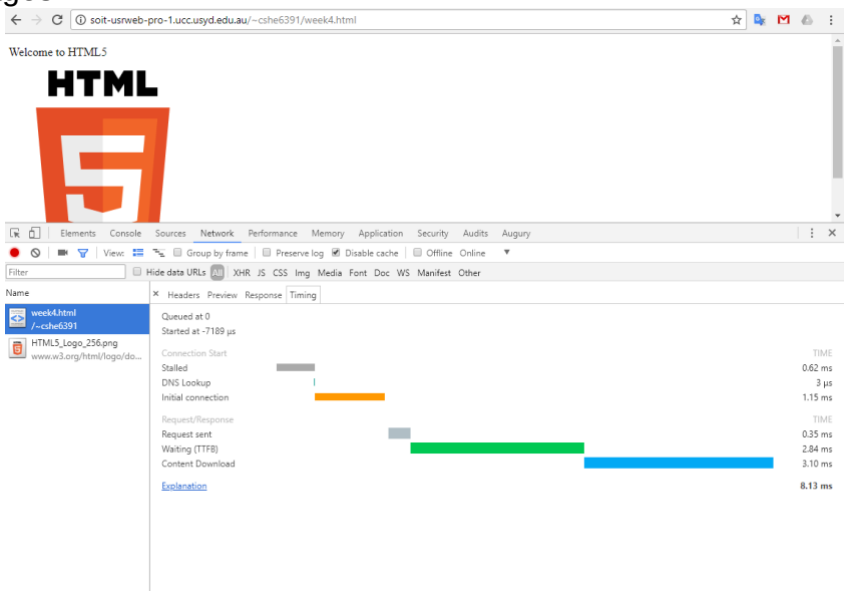

Figure 2: Timing details of request `week4.html`

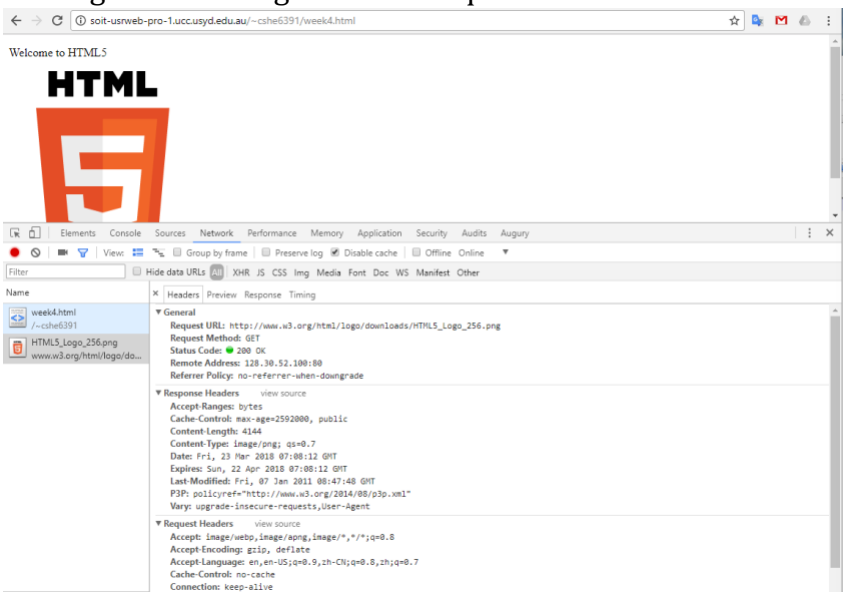

Figure 3: Headers of request `HTML5_Logo_256.png`

- How many requests are sent by the browser?
  Answer: two requests are sent by the browser
- Are the requests sent to the same server?
  Answer: No. The first one for `week4.html` is sent to `soit-usrweb-pro-1.ucc.usyd.edu.au` The second one for `HTML5_Logo_256.png` is sent to `www.w3.org`.
- What are the response status codes for all requests?
  Answer: Both are **200 OK** indicating the servers are able to find the resources and sent them back.

- Which server sends cache instructions and what are the instructions?
  Answer: The server `www.w3.org` sends cache instruction as part of response header. The two headers: `Cache-Control:max-age=2592000, public` and `Expires: Sun, 22 Apr 2018 07:08:12 GMT` carry the same message. It instructs the browser that the image `HTML5_Logo_256.png` is fresh for one month starting from the downloading date `Fri, 23 Mar 2018 07:08:12 GMT`

Use the information provided by DevTool to answer the following questions regarding browser rendering process:

- What are the DOMContentLoaded time and the Load time?
  Answer: The answer various but should have similar scale. Figure 1 shows that the `DOMContentLoaded` is 14 ms while `load` time is 425 ms
- What happens between these two events?
  Answer: The image file is downloading between these two events. The browser sends request to download the image when it encounters the `<img>` tag during HTML parsing. The image downloading usually takes longer time than the time used to construct DOM tree. The event `DOMContentLoaded` is fired after the DOM tree construction is complete. The `load` event is fired when the image finishes downloading. To be precise, the request for image starts slightly before the `DOMContentLoaded` because there are a few other elements after the `<img>` element.
- In Figure 1, there is a small gap between `week4.html` request time (8ms) and the `DOMContentLoaded` event time (14ms). What happens during the gap?
  Answer: During the gap, the browser is parsing the downloaded HTML file to build a DOM tree.
- In Figure 1, the request for `HTML5_Logo_256.png` took much 411ms while the request for `week4.html` took 8ms. You may have slightly different numbers in your DevTool. What is the main cause for delay in getting `HTML5 Logo 256.png`.
  Answer: Even though the image file is slightly larger than the text html file. The content download time do not differ a lot. The request for `HTML5 Logo 256.png` has much longer `TFFB` time than the request for `week4.html`. `TTFB` consists of network round-trip time (RTT) and server processing time. Since we are requesting static image/file, the server processing time should be quite small. `HTML5 Logo 256.png` has a longer RTT time because it is hosted on a remote server while `week4.html` is hosted on a local server.

b) Open a new chrome tab and navigate to `chrome://cache/`. This will bring up all content cached locally by Chrome. Search for the two objects: `week4.html` and `HTML5 Logo 256.png`. You should be able to find both. Clicking the link representing each object will display the actual cache entry. You will find the data captured are similar to the respective HTTP response header. For instance, both entries contain basic data such as `Last Modified` time. The cache entry of `HTML5 Logo 256.png` contain the `Cache-Control` header and `Expires` time. The cache entry of `week4.html` contains `ETag` data.

Theoretically, the browser will use cache validation mechanism for `week4.html` and cash expiration mechanism for `HTML5 Logo 256.png`. We will see if that is the case.

c) Now reload the page and inspect the content of Network panel (which would be similar to Figure 4 to answer the following questions:

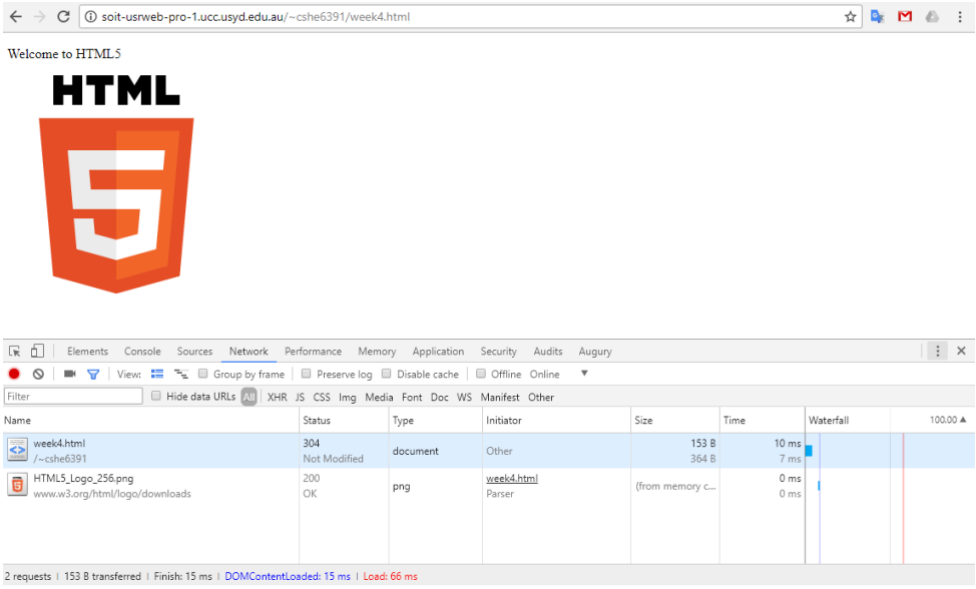

Figure 4: Reloading `week4.html`

- what are the latencies for both requests? Are there any difference between the latencies observed in the previous question (first time requesting `week4.html`)
  Answer: The latency for `week4.html` is similar in both cases: the requesting (8ms) and reloading(10ms). The latency for `HTML5 Logo 256.png` has a big difference: it was 411ms and changed to 0ms in the reloading case. The latency for `HTML5 Logo 256.png` is much shorter than the latency for `week4.html` in the reloading case.
- What are the `DOMContentLoaded` and `load` time? Are there any difference between the time recorded in the previous question (first time requesting `week4.html`)
  Answer: The `DOMContentLoaded` time is similar (15ms vs. 14ms). The `load` time (66ms vs. 411ms) is much shorter now.
- How does the browser obtain content of `week4.html` and `HTML5 Logo 256.png`?
  Answer: As expected, the browser uses cache validation mechanism for `week4.html` and cash expiration mechanism for `HTML5 Logo 256.png`. The cache entry of `week4.html` does not specify an expire time, so the browser sent a conditional request, with `ETag` and `Last Modified` time to server `http://soit-usrweb-pro-1.ucc.usyd.edu.au` The server returns code “304 Not Modified” to instruct the browser to use the cached copy. That explains the latency for getting `week4.html`, which still involves network latency. The `HTML5 Logo 256.png` cache entry has an expire time and we are still within the fresh frame, the browser does not send a request and uses the cached copy directly. That explains the very short latency

for getting `HTML5 Logo 256.png`, it is basically the time for reading the data from the local cache.

d) Open DevTool on another tab, point the browser to the following URL: `http://soit-usrweb-pro-1.ucc.usyd.edu.au/~cshe6391/week4withstyle.html` Your network panel may look like Figure 5:

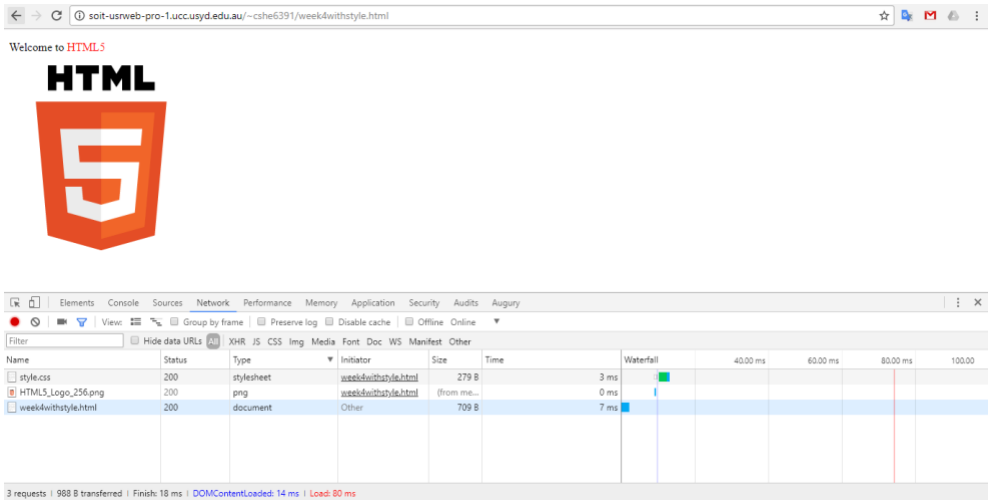

Figure 5: Loading `week4_ with style.html`

Try to interpret the rendering process of the browser based on the DevTool information. This page has two supporting resources: `style.css` and `HTML5 Logo 256.png`. In which order does the browser send request for each?

Answer: The browser first sets up a TCP connection to server `soit-usrweb-pro-1.ucc.usyd.edu.au` and obtained `week4 with style.html`. It starts to parse the HTML file to build a DOM tree. It first encounters the `<link>` tag pointing to the style sheet file. It sends a request to get this file. Because the style sheet is located in the same server (`soit-usrweb-pro-1.ucc.usyd.edu.au`), very likely, the browser will use the same TCP connection. At the same time, the browser keeps parsing the rest of the HTML file and encounters the `<img>` tag. It discovers that there is a cache copy for the file and loads it directly from the cache. That explains the short latency for getting the image file. The browser finishes parsing the DOM tree and fires `DOMContentLoaded` even at around 14ms. This happens before the style sheet file is downloaded. After the style sheet is downloaded and a CSSOM tree is built, the browser files the `load` event.