COMP4021
Internet Computing

The HTTP Process

David Rossiter

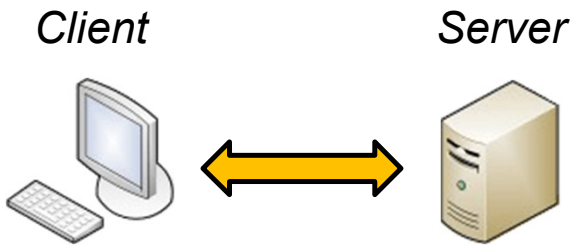
Getting a Web Page

- Here's the basic idea of getting a web page:
  1. You enter the URL into the browser
  2. The browser connects to the server
  3. The browser sends an HTTP request to the server
  4. The server returns an HTTP response, which includes the file you asked for
  5. The browser processes the file it has received and possibly might make more requests for files e.g. an image which is linked in the web page

1. Starting the Process

- Browsers use the HTTP protocol to talk to web servers
- For example, the browser asks for a web page using HTTP and the web page comes back using HTTP
- You can trigger the process by entering a URL in a browser:

2. Connecting to the Server



- The browser connects to the server typically through a *port*
- A port is like a door e.g. the French word for ‘door’ is ‘porte’
- A web site will typically use port 80
- Some other common ports are listed here:
  - HTTP and HTTPS are used for web page communication
  - HTTPS means ‘secure HTTP’, meaning that the communication can’t be read while it is being transmitted

Protocol	Port
FTP - file transfer	21
Telnet - typing commands on server	23
SSH - secure typing commands on server	22
HTTP	80
HTTPS	443

3. Sending an HTTP Request

- After successfully connecting to the server, the browser sends an HTTP request like this:

HTTP version being used in this communication

```
GET /index.html HTTP/1.1
```

1.1 is the current version

This request is using the HTTP GET method

Path on the server, and the file

A Complete Request

- For example, here is a real HTTP request sent by Chrome, showing everything that is sent to the server

```
GET /index.html HTTP/1.1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/80.0.3987.163 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3;q=0.9
```

this means the browser can accept anything.

Say to the server: these are the files I will accept.

Requesting The Root File on the Server

- If you enter something like *cars.com* (no file mentioned) the browser sends a request like this:

no file name here

```
GET / HTTP/1.1
```

  – This command asks for the root (/) file from the server, meaning the file at the top
  – Usually, the server replies with index.html (sometimes index.htm)

Request Methods

include the request parameters in the URL, which is different from POST

- GET is one of the possible requests
- There are others, for example:
  - POST — send some data to the server in a more 'invisible' way than the GET method
  - HEAD — request information about a file on the server **(but don't want the actual file)**

But GET is not very safe because the users can play around with the values in the URL.

like "tell me has a video been changed?",but don't give me the actual video file. In this way, if it is not changed, no data transmission occurs. This helps you save lots of bandwidth.

4. Returning an HTTP Response

- After the server gets the request it will send an appropriate response
- Each response has a response code
- For example:
  - `200 OK`
  - `400 Bad Request`
  - `404 Not Found`
  - `500 Internal Server Error`

The text as well as the number go back to the server.

The HTTP Response

- For example, here is an ‘OK’ response from the server:

The version of HTTP we are using

The response code

The response headers

```
HTTP/1.1 200 OK
Date: Thu, 16 Apr 2020 05:19:56 GMT
Server: Apache
Cache-Control: private
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

The content of the requested file

```
<!DOCTYPE html>
...
```

The type of the returned content

5. Processing the File

- If the file is an HTML file, the browser then converts the content into a memory structure (= the DOM)
- Depending on the file content, more requests will be sent to the server, e.g.:
  - If the page has a link to a CSS file, need to get it
  - If the page has a link to a JavaScript file, need to get it
  - If the page has a link to an image, need to get it
  - And so on…

Linking to Other Files

- Here is the HTML file returned from the CSE department website
- You can see the browser has to make lots of additional requests

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="author" content="Department of Comp
<!--meta name="description" content="Department
<!--meta name="keywords" content="CS, Computer
Engineering, HKUST"-->
<meta name="verify-v1" content="ErjT4aUT4dvi3EQ
<title>Department of Computer Science and Engir
<!--link rel="stylesheet" href="/styles/afix.cs
<link rel="stylesheet" media="print" href="/sty
<link rel="stylesheet" media="screen" href="/st
<link rel="stylesheet" media="screen" href="/st
<link rel="stylesheet" media="screen,print" hre

<script src="/scripts/script.js"></script>
<script src="/scripts/onload.js"></script>
<script src="/scripts/jquery/jquery-1.12.4.min.
<script src="/scripts/jquery-ui-1.12.1/jquery-u

...
```

Making Another Request

- For example, if the browser sees this link in the HTML file:

```
<script src="/scripts/script.js">
</script>
```

- Then it has to get that file, so it sends this request:

```
GET /scripts/script.js HTTP/1.1
```

Response From the Server

- Here is the response from the server:

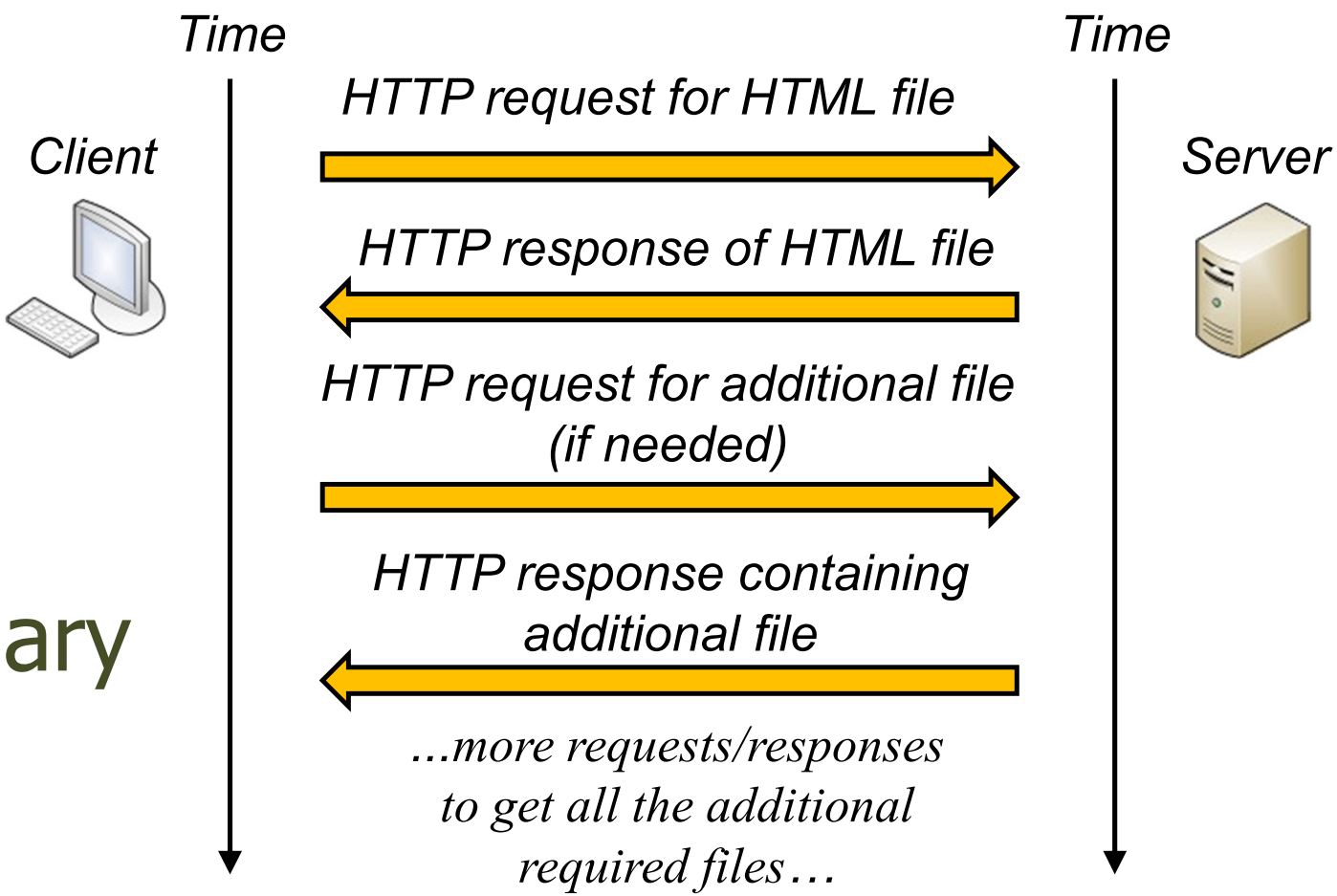
```
HTTP/1.1 200 OK
Date: Thu, 16 Apr 2020 05:19:57 GMT
Server: Apache
Last-Modified: Mon, 6 Apr 2020 03:00:00 GMT
ETag: "10df68bf-29f-4c4e9a0244c00"
Accept-Ranges: bytes
Content-Length: 671
Connection: close
Content-Type: application/x-javascript

function msend(user,place) {
...
```

This time the content is a JavaScript file

Here is the JavaScript file

Summary



websniffer.cc : view HTTP request and response headers.