CSE1500: Web - Lecture 1 - Intro & HTTP

Lecture Summary

The WDT Team

Ujwal Gadiraju, Sagar Chethan Kumar, Ciprian Ionescu

Contents

1 Intro & HTTP **2**
- 1.1 Web Clients and Servers . . . 2
- 1.2 HTTP Protocol & Messages . . . 2
- 1.3 Uniform Resource Locators (URLs) . . . 5
- 1.4 Authentication . . . 6

2 Sample Question Types **7**

3 Extra Exercises **8**

4 References and Further Reading **8**

⚠ These summaries are not extensive overviews of lecture content and **DO NOT** cover all possible exam material. Rather, these serve as a refresher and capture **main points** for a given lecture.

Learning Goals

You should be able to do the following after the **lecture**:

- **Explain the basic architecture of the Internet and the Web. (LO1)**
- **Describe** the interaction of Web clients and servers.
- **Describe** the different URL components.
- **Understand** the difference between HTTP and HTTPS.
- **Identify** the common HTTP message codes.

1 Intro & HTTP

1.1 Web Clients and Servers

The *World Wide Web* (WWW) is a global system of interconnected hypertext documents via the Internet. The *internet* is interconnected computer networks that span the globe communicating through a common standard (TCP/IP).

Simplified, the interaction between a Web server and a client can be described as a sequence of (1) **HTTP requests** and (2) **HTTP responses**. For example, if you open a web browser and type in the URL of your email provider, e.g. https://gmail.com, your web browser is acting as the client sending an HTTP request. Your email provider is acting as the server, sending an HTTP response.

The general architecture of network communication enables these interactions. In our course[1], however, we specifically need to be aware of the following network protocols: Internet Protocol **(IP)**, Transmission Control Protocol **(TCP)**, Hypertext Transfer Protocol **(HTTP)**.

1.2 HTTP Protocol & Messages

HTTP uses reliable data-transmission protocols (inherited from TCP). In layman terms, it is a protocol that enables users to communicate across the World Wide Web. A HTTP request and response can look like the following:

```
// Start Line
GET / HTTP/1.1

// Header Fields
Host: www.tudelft.nl
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:31.0)
   Gecko/20100101 Firefox/31.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1

// Body (Optional)
Cookie: __utma=1.20923577936111.16111.19805.2;utmcmd=(none);
```

Figure 1: HTTP Request Message

[1] Fun Fact! In CSE1405 Computer Networks (Y1Q4) you will learn much more about computer networks, including networks protocols which we cover briefly.

```
// Start Line
HTTP/1.1 200 OK

// Header Fields
Date: Fri, 01 Aug 2014 13:35:55 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 5994
Connection: keep-alive
Set-Cookie: fe_typo_user=d5e20a55a4a92e0; path=/; domain=tudelft.nl
[...]
Server: TU Delft Web Server

// Body (Optional)
...
...
```

Figure 2: HTTP Response Message

Entity bodies contain raw data. Hence, headers are needed to be able to interpret the data. In reality there are many header fields, but for our course specifically, here are some important ones: [2]:

Header Field	**Description**
Content-Type	Entity type
Content-Length	Length/size of the message
Content-Language	Language of the entity sent
Content-Encoding	Data transformations applied to the entity
Content-Location	Alternative location of the entity
Content-Range	Range defines the pieces sent for partial entities
Content-MD5	Checksum of the content
Last-Modified	Date this entity was created/modified
Expires	Date the entity will become stale
Allow	Lists the legal request methods
Connection	Connection control options (e.g., Keep-Alive)
Upgrade	Upgrade Protocol (e.g., HTTP/2)

Table 1: Well-Known HTTP Header Fields

[2] You are expected to understand the primary purpose of these headers. A good starting point can be found here.

In addition to headers, HTTP Status Codes are used through HTTP responses and can represent various things. Here is an overview of some common ones:

Status Code	Category	Example(s)
1xx	Informational	**100 Continue** - Request is still ongoing and not rejected by the server.
2xx	Success	**200 OK** - Request was successful, and the response contains the requested web resource.
3xx	Redirected	**301 Moved Permanently** - The requested resource has been definitively moved to the URL given by the Location headers.
4xx	Client Error	**404 Not Found** - The requested web resource or entity does not exist on the server.
5xx	Server Error	**502 Bad Gateway** - Indicates errors on the server side, typically a problem with a gateway or proxy server.

Table 2: Common HTTP Status Codes

You can use Telnet (a TCP connection to a Web server) to discover how these HTTP requests and responses look like in practice with your favorite websites![3] Naturally, there are also common HTTP methods used in practice that enable you to interact with these websites.

Method	Description
GET	Get a document from the Web server.
HEAD	Get the header of a document from the Web server.
POST	Send data from the client to the server for processing.
PUT	Save the body of the request on the server.
TRACE	Trace the message through proxy servers to the server.
OPTIONS	Determine what methods can operate on a server.
DELETE	Remove a document from a Web server.

Table 3: Common HTTP Methods

[3] `See the Reference and Further Extra Exercises Sections.`

1.3 Uniform Resource Locators (URLs)

Uniform Resource Locators (URLs) provide a standardized method for referencing any resource on the Internet. While not limited to the "http" scheme, the syntax may slightly differ from scheme to scheme. In general, a URL follows the format:

`<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<fragment>.`

You are expected to know what each part of the URL contributes to and relevant constraints. In addition there are also various schemes (more than just http) like the following:

- `http://<host>:<port>/<path>?<query>#<frag>`
- `https://<host>:<port>/<path>?<query>#<frag>`
- `mailto:<valid-email-address>`
- `file://<host>/<path>`
- `file:///Users/my_home_dir/tmp.html`
- `ftp://<user>:<passwd>@<host>:<port>/<path>;<params>`

This also introduces the notion of relativity within URLs and how references can change when using absolute or relative referencing. Consider the following case:

Base URL:

`http://www.example.com/subfolder/`

Absolute URL:

`http://www.example.com/subfolder/index.html`

Links

- Relative URL (../page1.html): Resolves to

  `http://www.example.com/subfolder/page1.html`

- Relative URL (subfolder2/page2.html): Resolves to

  `http://www.example.com/subfolder/subfolder2/page2.html`

1.4 Authentication

Authentication can also be present in HTTP. So far: HTTP serves as an anonymous, stateless request/response protocol. It treats the same request, regardless of the client sending it, in precisely the same manner.

HTTP basic authentication is a method where the server explicitly requests user authentication, requiring a username and password. HTTP has an inherent mechanism to support username and password-based authentication using the **WWW-Authenticate and Authorization** headers. In future HTTP requests to the site, the browser automatically issues the stored username/password.

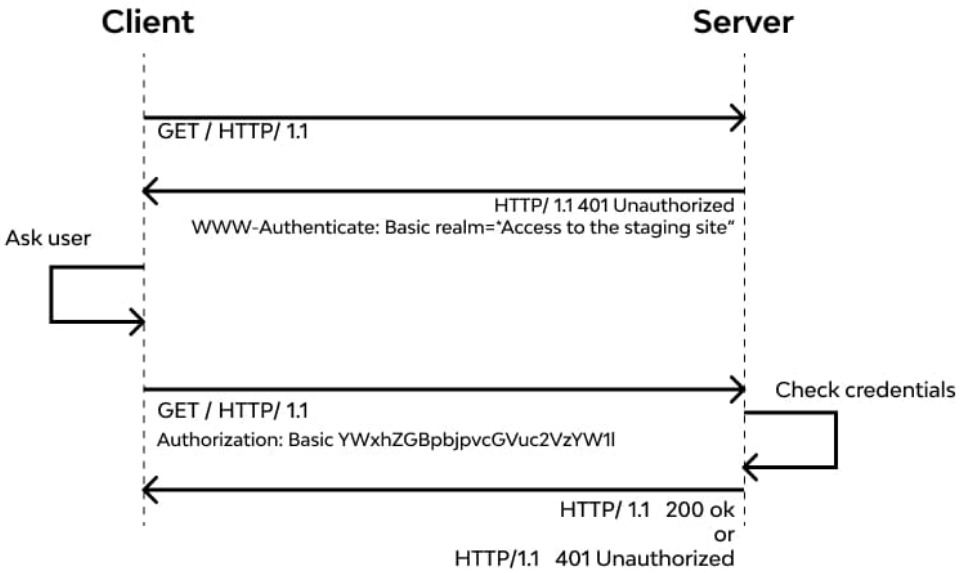

Figure 3: Example of Basic Authentication

However, these username and password can be decoded trivially (as they are sent over the network "in the clear"). To avoid this? **HTTPS** to the rescue.

HTTPS ensures Request and response data are encrypted before being sent across the network (SSL: Secure Socket Layer). **HTTPS** uses TLS (SSL) to encrypt normal HTTP requests and responses, and to digitally sign those requests and responses avoiding malicious users assuming an unintended role in the network.

2 Sample Question Types

(1) Which of the following is NOT an HTTP method?

a. GET

b. PUT

c. POST

d. REMOVE

(2) Consider the HTTP request message below. Which of the following statements about it is TRUE?

```
GET HTTP/1.1 /
Host: www.tudelft.nl
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9;
rv:31.0)
Accept: text/html,application/xhtml+xml
,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
DNT: 1
Cookie: __utma=1.20923577936111.16111.19805.2
;utmcmd=(none);
```

a. This HTTP request is invalid, as the client has to employ the Set-Cookie header to send its cookies to the server.

b. This HTTP request is invalid, as the syntax of the first line is incorrect. The correct syntax is the following: GET / HTTP/1.1

c. This HTTP request is invalid, as the User-Agent header field can only be set by the server.

d. This HTTP request is valid.

(3) Which of the following statements about URLs is FALSE?

a. The last part of a URL (#<frag>) names a piece of a resource and is only used by the client.

b. The <host> part of a URL can be a domain name or a numeric IP address.

c. The <path> is the local path to the requested resource.

d. If no <port> is provided in a URL, the default port number 0 is added to it.

3 Extra Exercises

If you are feeling confident on the topic of **An Introduction to Web Technologies and HTTP** check out the following tasks[4]:

- Write a sample HTTP request message for a GET request to retrieve an image file hosted on a remote server.
- Investigate the status codes in the 4xx range (client errors) and provide examples of situations in which they might be returned by a web server.
- Find a real-world website and use browser developer tools to inspect the HTTP requests and responses that occur when you visit the site. Document the key information you find.

4 References and Further Reading

Here are some useful resources for learning more about HTTP:

- Mozilla Developer Network - HTTP Guide
- W3Schools - What is HTTP?
- HTTP Status Codes - HTTPstatuses.com
- Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content - RFC 7231
- Basic Authentication Example

[4] `Note: None of these exercises are mandatory nor provide a bonus.`