Lesson 3:
Internet protocols overview and HTTP

Lesson goal

Before we approach the server-side programming, it is desirable to get some basic knowledge on the protocols that make browser to server communication possible. We are not going to delve into the protocol design details. Instead, we are interested here mainly in the following aspects:

- general understanding of the layered and modular design of Internet protocols;
- knowledge of advantages and limitations imposed by those protocols for the performance and functionality of Web applications;
- awareness of the lower level protocol originated details visible and useful for Web application developer.

Important details to investigate

- You need to get a general understanding of the OSI reference model for communication protocols, and be aware, how the actual set of Internet protocols shares its responsibilities among its layers (physical, network, transport, and application protocol layer)
- For the lower-layer protocols, we need to know the concept of encapsulation (i.e. how each protocol layer wraps and unwraps the payload delegated to it in order to fulfill its responsibilities)
- Regarding the abovementioned issues, make sure you are aware of the concepts like IP number (address), special addresses in IP (i.e loopback, private…) and port numbers
- What kinds of addressing are available for communication over the Internet? What is the role and underlying principles of operation of the DNS service
- What is the model of interaction imposed by HTTP and what are the main limitations of it (i.e. what are the main issues of HTTP 1.0 that version 1.1 addressed, and what performance issues motivated the recent introduction of HTTP2 ?).

External resources in English

To familiarize yourself with some details of the HTTP request structure and its semantics, you may want to check the source document of the HTTP 1.1 protocol specification (the RFC 2616 or newer) available e.g. at www.rfc-editor.org or www.w3.org .

Multiple sources explain the OSI reference model and how it maps into the TCP/IP protocol suite. See for example: https://www.youtube.com/watch?v=kCuyS7ihr_E

For a brief explanation of the overall context of the Internet protocol suite (the HTTP is used within) see e.g. http://www.thegeekstuff.com/2011/11/tcp-ip-fundamentals/

Hints on the assignment

This assignment deals with the level of abstraction lower than our usual work when developing a Web application, though we are still not lower than at the level 7 of the OSI model. We are going to get some insight into how HTTP requests and responses are constructed as a basically textual content divided into header and (in some cases) body parts. The actual shape of the HTTP request is normally composed by the browser and depends e.g. on the browser configuration settings, user actions, and on some client-side scripts working within the webpage currently opened.

If we want to investigate that traffic, the cleanest and most straightforward way is to depend on the features of the browser itself. For instance - consider the Developer Tools option in Chrome browser. In this of similar tool, the "Network" tab will show particular requests that are made by the browser to fulfil certain interaction with the server. The details like the code of the server's response and the request and response headers are shown. However, with standard tools at least, you are not able to modify and re-issue those requests. Hence the benefit of the tool like Fiddler Classic mentioned below: it includes dedicated options for showing a fully raw representation of request and response messages and allows to edit request and manually reissue them (see the "Unlock for editing" and (later) the "Replay > Reissue request" commands). On the other hand, the drawback of such tools is additional complexity as they constitute an intermediary separate from the browser (which requires entrusting its certificate in case the secure connections (https://) are to be utilized).

Assignment

1) Find and install a web debugger software of your choice, which allows to observe a complete content of HTTP 1.1 requests and responses intercepted from the browser communication, and makes it possible to send / re-send modified requests. That may be an integral part of your browser available as one of its “web developer” tools (e.g. Chrome offers a decent amount of such functionality out of the box), an optional plug-in, or a standalone tool (for the latter check e.g. the Fiddler Classic: https://www.telerik.com/fiddler/fiddler-classic).
2) Use that tool to investigate, how many requests are performed in the course of visiting a main page of a given website, and how those requests and respective responses are built.
3) Try to find a publicly available website and a document on it, for which a conditional GET request (i.e. using the “If-Modified-Since:” header) is supported (depends on a web server configuration, more frequently it occurs e.g. for graphical files being a part of the webpage).
4) Check the modification date of that document.
5) Invoke two variants of the request (using HTTP 1.1): one asking for that document unconditionally, and another containing a conditional request “if modified since” having some recent date. It is expected that you would receive another response code “Not modified” then.
6) For both the above requests provide the following:
    a. A complete code of the HTTP request, including all its headers (as short as possible – hence avoid any unnecessary header)
    b. A raw response from the server (you can skip the document body from it, but if it occurred – please indicate so and mark the place of its occurrence in the response)
    c. Describe the meaning of all the request and response headers.