Lab #1 - Web and Time Server

Karl-Johan Grinnemo
karl-johan.grinnemo@kau.se

Karlstad University — January 12, 2023

Introduction

This lab aims at introducing students to network programming and sockets. However, the students will also deepen their understanding of the Internet and the Web.

Examination

The lab is graded as *pass* or *failed*. To pass, the student or students should hand in their answers to the theoretical questions provided in the separate document, `Lab_1_Exercises.pdf`. They should also demonstrate the lab for one of the lab assistants in the course and submit their source code according to instructions on Canvas.

Preparations

- Review Chapter 1, *Computer Networks and the Internet*, in the textbook.[1]
- Solve the theoretical questions in the separate document, `uppgifter_lab1.pdf`.
- Read Section 2.2, *The Web and HTTP*, in Chapter 2, *Application Layer*, in the textbook.[1]
- Read Section 6.1.3, *Berkeley Sockets*, and Section 6.1.4, *An Example of Socket Programming: An Internet File Server*, in the document, `kompendium1.pdf`[2]

Description

This lab comprises two parts. In the first part, you are supposed to implement a miniature web server and, in so doing, get an opportunity to understand how HTTP and HTML interwork over the Internet fully. In the second part, your job is to implement a time server and a complementing time client that runs a time protocol over the UDP transport protocol.

[1] J. F. Kurose och K. W. Ross. *Computer Networking - A Top-Down Approach*, 7th edition.
[2] Excerpt from the textbook: A. Tanenbaum and D. J. Wetherall. *Computer Networks*, 5th edition.

Part I - Web Server

Implement a basic web server in C that can service a request of the default homepage, that is, `index.html`, from a sample website. The sample website is found on Canvas in a compressed file: `sample_website.zip`. Uncompress `sample_website.zip`, and write your webserver to serve the files from the folder `sample_website` when the browser requests them. This means that you should be able to add files to the `sample_website` folder without needing to recompile your webserver to serve them. You should be able to access your web server from a web browser, for example, Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari, by entering the URL: `http://localhost:<port>/index.html`. You decide which port your web server should listen to; however, a suggestion is to let it listen on port 8080. A good starting point for your web server is the tiny Internet file server described in the document, `kompendium1.pdf` (see Section "Preparations" above). The format of HTTP requests and responses is provided below: `<length>` is a placeholder for the length of the message body in bytes, `<type>` is the MIME type, for example, `text/html` for HTML content and `image/jpeg` for image data in JPEG format, and `<body>` is a placeholder for the actual message body.

Client or Browser Request:

```
GET␣/index.html␣HTTP/1.0\r\n
Host:␣localhost:8080\r\n\r\n
```

Server Response:

```
HTTP/1.0␣200␣OK\r\n
Server:␣Demo␣Web␣Server\r\n
Content-Length:␣<length>\r\n
Content-Type:␣<type>\r\n
\r\n
<body>
```

Apart from being able to handle successful responses (status code 200), your web server should be able to handle non-found resources (status code 404). You may omit the remaining possible responses.

Part II - Time Server

Implement a time server and client in C that works over UDP according to the standard RFC 868, *Time Protocol*[3]. The client should print out the time returned from the server in a readable format, for example, the format produced by the C standard library function, `ctime` (`time.h`). Remember that IP addresses and port numbers should be sent over a network in network byte order.

End of Lab

[3]The RFC is available for download on Canvas.