


This exercise is meant to give you a limited experience of socket programming and some intimacy with the HTTP protocol. If you feel that you are already well-versed with these topics consider following Assignment 2 (mail client), 2a (simpler mail client), or assignment 4 (web proxy server) from the book website.

Tasks marked  need to be documented in your course log, handed in for the exam evaluation in the end of the semester.

In addition you should send intermediate results by email to [mikkelbu@itu.dk](mailto:mikkelbu@itu.dk) as specified below.

## Introweek: Warm-Up

Read Section 2.2 in Kurose & Ross, which describes the HTTP Protocol. The specification of the protocol is contained in RFC 2616:

<http://www.w3.org/Protocols/rfc2616/rfc2616.html>

(also included in the zip file with resources for this exercise: `rfc.zip`).


*Remark:* You do not need to study this RFC in detail!

Read Section 2.7 in Kurose & Ross, which describes socket programming with TCP. The part on `TCPServer.java` is particularly important for the exercise.

Solve exercises P2 and P3 pp. 206–207 in Kurose & Ross.


You should also experiment with `Ethereal` (see p. 102 in Kurose & Ross) and with `traceroute` (p. 66 loc. cit.). Both are useful for network debugging, and used in practice a lot.

## Week 2: Listening for requests

 Implement Assignment 1: *Building a multi-threaded web server* from the book's website, part A.

Constraints: Your program should not have more than 200 lines of Java code (mine had two classes, in total 110 lines including blanks).

When your server is ready connect to it with your browser (or several different browsers), and with a telnet session containing at least one GET request. Document the output from the server.

 Additionally solve problem P15 p. 210 in Kurose & Ross.

*Hand-in:* a single PDF file containing source code, the output from the server (max 1 page), and a solution to Problem P15 (max 1 page).

*Deadline:* before the third lecture in the course.

## Week 3: Serving files

- ✎ Implement Assignment 1: *Building a multi-threaded web server*, Part B.

File `rfc.zip` contains the HTTP 1.1 specification. Such a small website without figures (just hyper linked text in HTML) is useful to test your web server. If you are able serve images, then use any HTML source you have access to — for example your own website.

Constraints: Your program should not have more than 400 lines of Java code (mine had 5 classes, 220 lines in total).

Briefly test your server using several browsers and a telnet session containing both a successful and an unsuccessful GET request.

It is best to stop reading now.  
Continue after your implementation is ready.

After you are done, place some file, say `secret.html`, in a parent directory to the one in which your server is running. Will your server prevent an attacker from reaching this secret file? Try to get the file with your browser by following this link (assuming that the web server runs on port 6789):

<http://localhost:6789/../secret.html>

- ✎ Does it work? Discuss with your group why it does, or why it does not. Briefly summarize the conclusion in writing.
- ✎ Design and describe a simple attack to get `secret.html` served by your server using `telnet localhost 6789` instead of a standard browser.

*Deadline:* before the fourth lecture in the course.

*Hand-in:* a single PDF file containing source code, a transcript of a telnet session accessing the server (with a page being successfully served) (max 2 pages), and a discussion on security as requested above (1–2 pages).