Assignment Two

A HTTP Proxy Server

Set: *2nd of May 2011*
Due: *13th of May 2011 @ 23:55 CEST*

Synopsis:
Implement a HTTP Proxy server with a local cache and examine its effect on network characteristics

Introduction

This is the second of four assignments in the *Datanet* course. The four assignments are practical in nature, and you will gradually build a distributed anonymising web proxy. Your proxy will be part of a real distributed system running on the Internet, with peers contributed by your fellow students. In this second assignment you will need to gain an understanding of how an HTTP proxy server works, provide an implementation of a caching proxy server, and explore how it can change network characteristics, such as latency and bandwidth.

Implementation

In this assignment you must implement a caching HTTP proxy server. You are encouraged to use the web server from the previous assignment and extend it to also work as a proxy server. This is not a requirement however, and you can start over and implement the caching proxy server from scratch. In this, and in the following assignments, you may use HTTP aware libraries but do beware that some HTTP aware libraries may prevent you from emitting correct headers or cause other issues.

In order to make your implementation easier we suggest you split it into to two stages. In the first your implementation should be able to parse a HTTP proxy request, forward it to the correct destination and return the response, *without* caching the content. Your implementation should modify as little as possible, similar to what is called a transparent proxy in RFC 2616. You should then extend your proxy to cache cacheable responses.

The final caching proxy implementation *must* maintain a cache of all cacheable responses which means that you should examine the `Cache-Control`, `Last-Modified`, `If-Modified-Since` headers and possibly also the `ETag` header. You *should* support the `HEAD` method, as the browser will likely use it once it detects a cache. Implementing a complete caching proxy is tricky and you are not supposed to support all cases possible according to the RFC. A simple working design is preferred over a complicated partially working design. You should explain and justify the types of caching that your caching proxy supports.

As in Assignment One, You may choose to implement the web server in any reasonably readable language (e.g. Python, Java, C#, SML, Perl, ...). Your caching proxy *should* be able to serve multiple requests at the same time and you could use either multithreading or asyncrhonous IO to achieve this. Be aware however that special attention is needed as the cache is shared between the simultaneous requests. You will need to choose a strategy to deal with this problem. You may, for example, want to delay a request if the file is already being fetched; or you could choose to make multiple requests for the same file and then figure out what to do once the results arrive.

Experiment

To complete the assignment, you must also perform a set of experiments using your caching proxy. These experiments will explore the effect that the cache has on page loading. For testing, you should pick a website that consists of many small files, such as: `http://slashdot.org/`, `http://gizmodo.com/`, or `http://dk.msn.com/`. With your chosen site(s), perform the following:

- Clear your browser cache
- Disable proxies
- Load the selected site and record how long it takes
- Re-load the site while, again recording how long it takes

This experiment will give you an idea of how the page loading performs in the browser, both with and without a cache. When timing the loading of a page, you can either use a stop-watch, or you can use FireBug for FireFox or the Web Inspector in Safari or Chrome. These tools all have a *network* panes which will show you a detailed breakdown of the duration of a page load.

Now, repeat the experiment, this time ensuring that the web browser uses your proxy[1] but ensuring that it does not cache any data. This experiment will show you how much latency your proxy adds to the overall procedure.

Finally, you should then enable the cache in your proxy server and repeat the experiment. Once with a clean proxy cache, and then again with a filled proxy cache. This will show you the effects of caching the files locally.

The experiments detailed above should enable you to produce a table with the following values:

- Time without proxy and without any cache
- Time without proxy but with browser cache
- Time with proxy but without any cache
- Time with proxy and browser cache

[1] If you are using FireFox you can set the proxy settings in the FireFox preferences. If you are using Safari, Chrome, or Internet Explorer you will need to change your system wide proxy settings, which is what the browsers use.

- Time with proxy while filling proxy and browser cache
- Time with proxy with both browser and proxy cache filled
- Time with proxy and proxy cache but without browser cache

To illustrate that latency and bandwidth are not necessarily related, you should also try to load a large file, with and without your proxy. You can try to download an Ubuntu CD from their website: `http://ubuntu.virginmedia.com/releases/10` If you have a slow Internet connection you may want to find a smaller file for your experiment.

You must download the same file at least three times, and report the following:

- Time with no proxy and no cache
- Time with proxy but no cache
- Time with proxy and with cache

As a final experiment, if you are able to change your server from serving multiple concurrent requests to being able to serve only a single request at a time, then you can explore latency hiding. Perform the first experiment, where you load a webpage consisting of many small files, first: with your proxy in *concurrent requests* mode and then: with your proxy *single request* mode. Note down the times and discuss how the browser hides the latency of loading the many items that make up a webpage.

Report

Your report *must* be written in ACM format. An ACM template for LaTeX and Microsoft Word is available for download via Absalon. Your reports should contain:

- An abstract describing the contents of your report
- A description of your proxy server
- A description of your cache strategy, including strategy for simultaneous requests
- A description of what headers you examine and what you use them for
- A description of your experiments
- The timings found for each of the ten required experiments
- The optional latency hiding experiment

Deliverables for This Assignment

You are encouraged to work in informal groups for this assignment, for the purpose of discussing implementation details and limitations. We strongly encouraged you to come to the exercise, where we will use time to discuss the design, implementation etc. The implementation and report that you hand in must however be **your own individual work**. You should submit the following items:

- A single PDF file, A4 size, no more than 3 pages, in ACM format, describing each item from report section above
- A single ZIP/tbz2 file with all code relevant to the implementation

Handing In Your Assignment

You will be handing this assignment in using Absalon. Try not to hand in your files at the very last minute, in case your caching proxy server eats your homework. **Do not email us your assignments unless we expressly ask you to do so**.

Assessment

You will get written qualitative feedback, but no grade for the assignment. Your assignments will be evaluated together with your exam and produce a final grade for the course.