Lab 1: A Simple Web Proxy Server

50.012 Networks

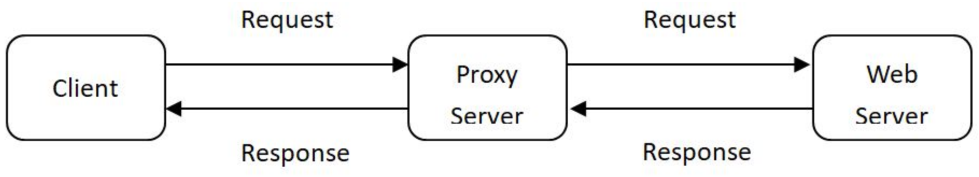
Hand-out: 2 Feb
Hand-in: 12 Feb 23:59

1 Objectives

In this lab, you will learn how web proxy servers work and one of their basic functionalities — caching. You will also familiarize yourself with socket programming and basic of HTTP protocol.

Your task is to develop a highly simplified web proxy server which is able to cache web pages. It is a very basic proxy that only understands HTTP GET requests. In particular, it does not need to support HTTPS and does not need to support other types of HTTP methods (e.g., PUT, POST...)

Generally, when a client makes an HTTP request, the request is sent to a web server. The web server then processes the request and sends back a response message to the requesting client. If we put a proxy server between the client and the web server, both the request message sent by the client and the response message delivered by the web server pass through the proxy server. In other words, the client requests the objects via the proxy server. The proxy server will forward the client's request to the web server. The web server will then generate a response message and deliver it to the proxy server, which in turn sends it to the client.



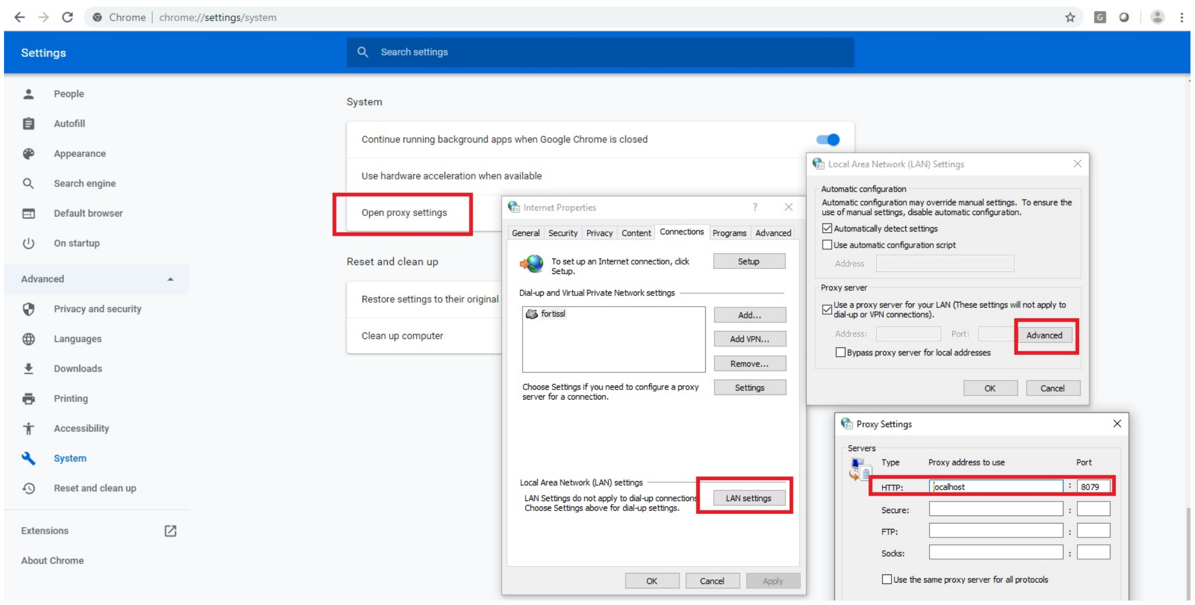
2 Code

You will be provided with a skeleton code (in Python 3) for the proxy. You are to complete the skeleton code. The places where you will fill in your code are marked with *#Fill in start* and *#Fill in end*. Each place may require one or more lines of code. Feel free to make additional change to the code (outside the marked regions) to make it work better. You can also start from scratch without using the provided skeleton, but please provide sufficient comments in your code and a brief README file to help us understand and test your code.

3 Running the Proxy Server

After you finish your coding, you can run the proxy. You can pass the port number you want your proxy to listen to as an optional argument. In the supplied code, it will listen to 8079 by default.

From a terminal, you can use *sudo lsof -i :8079* to checkout the process that is listening on that port.



Continue running background apps when Google Chrome is closed

Use hardware acceleration when available

Open proxy settings

To set up an Internet connection, click Setup.

Dial-up and Virtual Private Network settings

fortissl

Choose Settings if you need to configure a proxy server for a connection.

Local Area Network (LAN) settings

LAN Settings do not apply to dial-up connections. Choose Settings above for dial-up settings.

Automatic configuration

Automatic configuration may override manual settings. To ensure the use of manual settings, disable automatic configuration.

[✓] Automatically detect settings

[ ] Use automatic configuration script

Address:

Proxy server

[✓] Use a proxy server for your LAN (These settings will not apply to dial-up or VPN connections).

Address: Port:

[ ] Bypass proxy server for local addresses

Type	Proxy address to use	Port
HTTP:	localhost	8079
Secure:		
FTP:		
Socks:		

[ ] Use the same proxy server for all protocols

You can then configure your web browser to use your proxy. How to set this up depends on your browser. A snapshot of the configuration for chrome is given in the figure above (this may vary with your OS too). In general, you need to specify the IP address of the host that you run your proxy and its port number there. If you run your proxy and your web browser on the same computer, you can simply use 127.0.0.1 or localhost for the proxy's address. If you run them in different computers, make sure the two computers can reach each other, and you need to find out the IP address of the computer that runs the proxy server, and use that address to configure your browser's proxy setup. After you start your proxy server and configure your browser, to get a web page, you simply provide the URL of the page you want in the browser, e.g.,

- `http://gaia.cs.umass.edu/kurose_ross/interactive/`
- `http://www.neverssl.com/`
- `http://www.httpvshttps.com/`

As our simple proxy only supports HTTP sites, above list give you a few stable HTTP websites that you can test against.

Hint: after you visit a website, your browser likely also keeps its own local cache. To force your browser to ignore its local cache and always contact the server, you can use Chrome's DevTools (press F12 to launch it), and from there you can select Disable Cache (under the Network tab), and then you can use Ctrl + R to refresh.

4 Hand-in

You will hand in your completed proxy server code (submitted as a single file proxy.py).

Before you submit your proxy server code, please verify that it can help your browser load the web resources from a HTTP website. Please also verify that if you visit the same page for the second time, the proxy will be able to serve the content from its local cache, instead of contacting the original web server again. We will grade your submitted code, based on whether it supports these basic proxy functions, and based on your use of socket programming (so please do not write your code using higher level libraries, e.g., an HTTP library).

5 Further exploration (optional)

Note that after you put in the minimum code to make our simple proxy run, the proxy still lacks many important features, e.g., the understanding of various header fields received in a request from the client. Your are encouraged to read RFC7234 `https://tools.ietf.org/html/rfc7234` to learn more about some complexity involved in implementing a proxy.

Here are some questions you can think about when reading the RFC:

- How the proxy should use the If−None−Match and ETag headers in the HTTP Request and Response respectively to decide its response to the client?
- How to support HTTP Requests with the Range header?

You are encouraged to think about how to add these features into our simple proxy code.

In addition, you are encouraged to try out the following software. They all provide web proxy functionality as part of their feature offerings:

- Burp Suite
  - `https://portswigger.net/burp/communitydownload`
  - `https://portswigger.net/burp/documentation/desktop/tools/proxy/using`
- POSTMAN
  - `https://www.postman.com/downloads/`
  - `https://learning.postman.com/docs/sending-requests/capturing-request-data/capturing-http-requests/#built-in-proxy`
- NGINX
  - `https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/`

Some questions you can think about when trying the software: What are the main use cases of them, respectively? What are the benefits a web proxy can offer in the respective use cases? What are some different approaches that a proxy can take to handle Https connections?

If you explore some of the software listed above or implement any advanced features in your proxy, you are encouraged to summarize your findings in a separate write-up and submit that together with your proxy code. We will share selected submissions with your classmates.