

Computer Networks

- A Simple HTTP proxy -

Objectives

The intent of this assignment is to help you gain a thorough understanding of:

- The interaction between browsers and web servers
- The basics of the HTTP protocol
- The role of proxies in the web infrastructure

Description

HTTP (HyperText Transfer Protocol) is a simple client-server transfer protocol. The client is the Web browser that requests Web objects. The Web server sends objects in response to requests.

Suppose that you type the URL <http://www.csc.villanova.edu/index.html> into your Web browser. The Web browser will open a TCP connection to www.csc.villanova.edu on port **80** and send something like this:

```
GET /index.html HTTP/1.0\r\n\r\n
```

The `\r\n` sequence represents the ASCII character 13 (`^r`) followed by ASCII character 10 (`^n`). In response, the web server will locate the file called `/index.html` and transmit its contents to the web client in the form:

```
// some header here
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head language="JavaScript">
<title>Welcome to the Villanova University Department of Computing
Sciences ...
```

Here only the first few lines of the HTML file that is delivered are displayed.

Try it out

1. Telnet to your favorite Web server (for instance, www.csc.villanova.edu):

```
telnet www.csc.villanova.edu 80
```

The telnet application opens a TCP connection to port 80 (default HTTP port) on www.csc.villanova.edu. Anything that you type in next will be sent to port 80 at www.csc.villanova.edu

2. Type in an HTTP GET request:

```
GET /index.html HTTP/1.0
```

and hit Enter **TWICE**. In this way you send this minimal (but complete) GET request to the HTTP server. It is possible that you won't see your GET request on the screen as you type it in. If you make a typing error, you will get a "**Bad Request**" response from the server – in this case, start over from Step 1.

3. Look at the response send by the HTTP server! It should be identical to the text you see when selecting "**View/Source**" on the top bar menu of your browser.

The HTTP GET request

The full HTTP specifications are given in RFC 2616 (Request For Comments 2616), available from the Web Consortium at www.w3.org/Protocols, but it happens to be about 150 pages long. Here is all you need to know about **GET** requests (there are other requests as well, such as HEAD, POST, but you need not worry about them for this assignment).

A GET request consists of several lines of text separated by `\r\n`. The first line is of the form: **GET URL VERSION**. The following lines are headers. The end of the headers is signified by a blank line. For instance, a GET request might look like

```
GET /index.html HTTP/1.0\r\n
Accept: /* */\r\n
Accept-Language: en-us\r\n
If-Modified-Since: Mon, 16 Jan 2009 13:55:31 GMT\r\n
If-None-Match: "1763b-235a-3d85e2d3"\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n
Host: www.csc.villanova.edu\r\n
Proxy-Connection: Keep-Alive\r\n
\r\n
```

Do not worry about the details of these headers. You will only need to process the first line and simply append the rest of headers to your HTTP request.

What is an HTTP proxy server?

An HTTP *proxy server* is a program that acts as an intermediary between the client and the server in order to monitor, modify and/or prohibit transactions. Proxies are commonly used in firewalls. All Web browsers may be configured to use a proxy server. This means that when you try to load a page such as

<http://www.csc.villanova.edu/index.html>

the browser instead contacts the designated proxy server and asks it for the page as follows:

```
GET http://www.csc.villanova.edu/index.html HTTP/1.0 \r\n
Accept: /*/* \r\n
Accept-Language: en-us \r\n
If-Modified-Since: Mon, 16 Jan 2009 13:55:31 GMT \r\n
If-None-Match: "1763b-235a-3d85e2d3" \r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n
Host: www.csc.villanova.edu \r\n
Proxy-Connection: Keep-Alive \r\n
\r\n
```

Notice the difference here: instead of asking just for the resource [/index.html](#), the web browser asks for the full resource <http://www.villanova.edu/index.html>. The proxy then has enough information to fetch the file [/index.html](#) from www.csc.villanova.edu on behalf of the web browser and deliver it as a result, or do whatever it wants to do.

To fetch the file [/index.html](#) from www.csc.villanova.edu, the proxy extracts the file name from the first GET line and forwards the request

```
GET /index.html HTTP/1.0 \r\n
Accept: /*/* \r\n
Accept-Language: en-us \r\n
If-Modified-Since: Mon, 16 Jan 2009 13:55:31 GMT \r\n
If-None-Match: "1763b-235a-3d85e2d3" \r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n
Host: www.csc.villanova.edu \r\n
Proxy-Connection: Keep-Alive \r\n
\r\n
```

as if it came from the HTTP client. The HTTP server sends the requested file back to the proxy server, which forwards it to the HTTP client (or filters it out). The proxy server may also decide to cache a copy of the file and serve the local copy in response to future requests rather than going to the server.

What you need to do:

Implement a multi-threaded HTTP proxy that accept requests through a known TCP port, forwards each request to the real server and sends back any reply to the client. So your program must be able to act as a server (to receive HTTP requests) and as a client (to send the HTTP requests to the real server).

To get familiar with threads, study the multi-threaded EchoServer.java available on the class page (in the Java Material section). Then you'll be ready to start implementing your own HTTP proxy. Write code that does something like this:

1. Accept a TCP connection from a real HTTP client (running in a Web browser)
2. Spawn a new thread to handle the connection
3. Go to Step 1

The spawned thread should do the following:

1. Read a request from the client and parse the first line. If it is not a GET request you can ignore it (close the connection). Here is an example of a GET request (first line only):

```
GET http://www.villanova.edu:80/index.html HTTP/1.0\r\n
```

You need to parse this line to determine the hostname (www.villanova.edu), the port number (**80** by default, if none specified) and the pathname of the HTML object you want (</index.html>). This tells you that the HTTP server you should contact is running on the host **www.villanova.edu** on port **80**

2. Establish a TCP connection to port 80 on the host (www.villanova.edu) determined in Step 1
3. Forward the GET request and any following HTTP header lines (using *only the path* instead of the full URL) to the real HTTP server – for our example the request you should forward is

```
GET /index.html HTTP/1.0 \r\n
Accept: /*/* \r\n
Accept-Language: en-us \r\n
If-Modified-Since: Mon, 16 Jan 2009 13:55:31 GMT \r\n
If-None-Match: "1763b-235a-3d85e2d3" \r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\r\n
Host: www.csc.villanova.edu \r\n
Proxy-Connection: Keep-Alive \r\n
\r\n
```

Make sure you are sending an extra blank line after all the headers, to signify the end of the request.

4. Read everything sent back by the HTTP server and send it back to the client. Since binary files (such as images) may get sent to the client, you cannot use **readLine**, but you may use a loop similar to the one below:

```
InputStream netIn;          // to read from the Web server
OutputStream clientOut;    // to write to the Web client

// Initialize netIn, clientOut

byte [] buffer = new byte[4096];
int bytes_read;
while ((bytes_read = netIn.read(buffer)) != -1)
{
    clientOut.write(buffer, 0, bytes_read);
    clientOut.flush();
}
```

5. Close the connections to the server and client and exit

These steps make up a minimal proxy HTTP server. Proxies are used for many other purposes (to cache objects, to keep a log of all requests in a disk file, etc.)

Important notes:

1. Your program must be able to act as a server (to receive HTTP requests) and as a client (to make the HTTP requests from the real server). I strongly suggest that you begin by writing the client side first, and getting the client to work with an existing Web server. You can start by sending the request

```
GET /index.html HTTP/1.0\r\n\r\n
```

to the Web server running on www.villanova.edu, port 80 (hardwired in your code) and printing out the reply as text only:

```
// some header here
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head language="JavaScript">
<title>Welcome to the Villanova University Department of Computing
Sciences ...
```

Once you've got this part working, write the server that accepts requests from real HTTP clients and processes them before forwarding them to the real server.

2. Remember, each request ends with an empty line “\r\n”
3. Your proxy should work with Internet Explorer. To set up an Internet Explorer proxy, look under [Tools/Internet Options](#), click on [Connections](#), then [LANSettings](#) and select “[Use a proxy server](#)”. Type in the address of the host running the proxy (**localhost** if the local machine) and the port number on which your proxy is listening for connections.
4. Each student's proxy must listen on a different port number to avoid port number collisions in case you happen to run your programs on the same machine. Pick a random number – the likelihood that two students working on the same machine will pick the same number is small.
5. Remember to close every socket that you use in your program.

Submission guidelines

[Schedule a demo time with me.](#) Hand in a short README file and a hardcopy of your source files (one per team). The README file should explain the contribution of each member to the project, and list any bugs and/or deviations in your program from the assignment specification. **Have fun!**