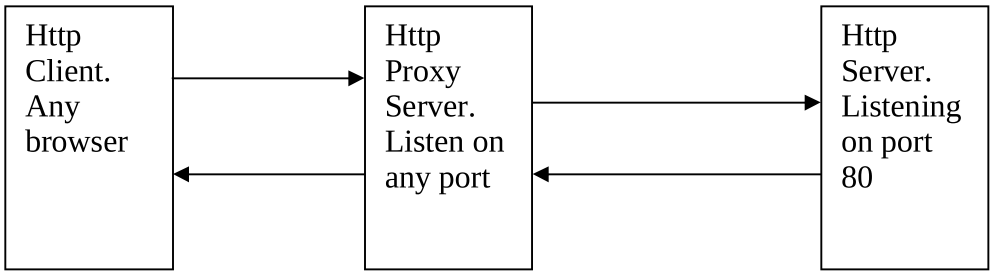
CSci 5131: Advanced Internet Programming, Spring 2009
Assignment 1: Building a HTTP Proxy server in Java
Due Date: Feb 8, 2009
This assignment is to be done either individually OR in a group of three.

Objective:
The objective of this assignment is to learn HTTP protocols (1.0 and 1.1) and build a HTTP proxy server. In this assignment you would learn how to program using Java and TCP sockets the functionalities of an HTTP client and server.

You will need go through RFC 1945, which describes HTTP 1.0 protocol. You may also need to refer to RFC 2616 for HTTP 1.1 protocol (Section 14.9 of this document specifically discusses cache management.)

Introduction:
When a web client (such as Internet Explorer) connects to a web server (such as www.google.com) the interaction between them happens through Hyper Text Transfer Protocol (HTTP). A proxy server is an intermediary between HTTP client and HTTP server. In this assignment we will build an http proxy server. The functioning of the http proxy server is as shown in the following figure and explained below:


Http Client. Any browser
Http Proxy Server. Listen on any port
Http Server. Listening on port 80

Working of the HTTP Proxy server in brief:
1. The proxy server acts like a HTTP server to the HTTP client and as HTTP client for a HTTP server.
2. An HTTP client connects to the proxy server instead of directly connecting to the HTTP server. Most of the web browsers can be set to connect to the Internet through the proxy server.
3. In order that the client can connect to the proxy server, the proxy server has to keep on listening on a particular port that is known to the client.
4. Once the proxy server gets a request from the client, it opens a connection to the HTTP server mentioned in the request on port 80. The proxy server sends the HTTP request to the HTTP server on this connection.
5. The proxy server waits for the HTTP server to respond. When the proxy server receives the HTTP response, it forwards that response to the HTTP client.
6. Advantages of using a proxy server can be listed as follows:
   a. Access to certain websites can be blocked
   b. Access to certain type of content can be blocked (for example jpeg files).
   c. Files can be cached locally and the client requests can be satisfied from the local cache.

Problem Statement:
Write a proxy server in Java. The required functionality of the proxy server is as follows:

1. It should block access to certain websites. The names of the websites will be specified in a text file. The text file will also contain comments describing the filter. These comments and blank lines will be ignored by the proxy server.
   1.1 If the client request accesses one of the blocked websites then the proxy shall inform the client that the access is denied. The response from the proxy shall be in the form of a HTTP response.
2. It should block certain type of content. The type will be specified in a text file.
   2.1 If the response from the server contains one of the disallowed types then the proxy shall not forward that content to the client.
3. It should cache the content of the non-blocked websites. A client request shall be satisfied from the cached content if the requested data is present in the cache.
   3.1 If the requested data is not in cache then the proxy shall bring the data from the server and cache the data locally.
   3.2 There is no limit on the cache size of the proxy.

Example:
Suppose that a file named config.txt contains the names of the domains and the names of the content-type that is blocked by the proxy in the following format:

```
# Filename: config.txt
# Description: This is the config file for proxy server
# Author: <your name>
# Date: Jan 22, 2009

# block complete access to the site
www.badsite.com  *

# block complete access to the site (missing * defaults to 'block complete access' as above)
www.badsite.com

# block all gif files from www.microsoft.com
www.microsoft.com     image/gif

# block all images i.e. gif, jpeg, png, etc.
www.sco.com           image/*
```

The complete list of currently used MIME content type is available at
http://www.iangraham.org/books/html4ed/appb/mimetype.html
Suppose you are working on one of the itlabs machine: watermelon.itlabs.umn.edu.
You will run your proxy from the command line as follows:

```
% java Proxy config.txt 50000
```

where, config.txt is the config file, and 50000 is the port where this server will run.

Setting up Browsers to Use Proxy:

1. Mozilla:

   Tools->Options->Advanced->Network->Settings. Once you reach this window you will find three radio buttons. Click on "Manual proxy configuration". Then in the "HTTP Proxy" field give the complete machine name of the machine on which you are running your proxy, in our case it will be "watermelon.itlabs.umn.edu". In the "Port" field enter the port number on which you are running your proxy.

2. Internet Explorer (IE):
For the IE browser the settings are as follows:
Tools->Internet Options->Connections->Lan Settings->Proxy Server

To test your program, you will be given one or more sample configuration files. During grading, we will test your program with these as well some additional test files.

Note on using port numbers:
Port numbers are divided into 3 ranges:

1. Well-known ports: 0 through 1023.
2. registered ports: 1024 through 49151
3. dynamic or private ports: 49152 through 65535

Use any port from the 3rd range.

Once the above setup is complete, you can try connecting to any website through the web browser. Suppose we connect to www.gnu.org. Then you should be able to see that site without any problems. On the other hand if you connect to www.badsite.com then you should not be able to visit this site. Instead, you should get a message: "You are trying to visit a blocked site!"

Assignments Details:

1. You will have to read the following parts from the HTTP 1.1 (RFC 2068). The link for the RFC can be found on the class web page.
   a. HTTP Request format
   b. HTTP Response format

2. Schematic design of your Proxy server can be as shown below. You can use this design as a starting point, or you may develop your own design.

```
public class Proxy {

        // Data Members:
        int proxyPort; // Proxy's port
        ServerSocket proxySock; // The Proxy Server will listen on this socket

        // clientSock represents the Proxy's connection with the Client
        Socket clientSock;

        // serverSock represents the Proxy's connection with a HTTP server
        Socket serverSock;

        // Constructor
        public Proxy (String configfilePath) {
          // Read the config file, and populate appropriate data structures.
          // Create Server socket on the proxyPort and wait for the clients to
          // connect.

          proxySock = new ServerSocket(proxyPort);
          while (true) {
                clientSock = proxySock.accept();
                handleRequest(clientSock);
          } // end of while
        }

        /* This method is used to handle client requests */
        private void handleRequest(Socket clientSocket) {
                // read the request from the client
                // check if the request is for one of the disallowed domains.
                // If the request is for a disallowed domain then inform the
                // client in a HTTP response.
                // satisfy the request from the local cache if possible
                // if the request cannot be satisfied from the local cache
                // then form a valid HTTP request and send it to the server.
                // read the response from the server.
                // check the Content-Type: header field of the response.
                // if the type is not allowed then inform the client in a
                // HTTP response.
                // Send the response to the client
                // Cache the content locally for future use
        }
```

```
            // main method
            public static void main(String args[]) {
              // Read the config file name and the Proxy Port.
             // Do error checking. If no config file is specified, or if no Port is      specified then exit.
            }
    }
```

The disallowed domains and the disallowed types will be specified in a config file, similar to the one described above. The file will be specified on the command line when the proxy server is run.

3. The HTTP client (eg: mozilla) will send a complete HTTP request to the proxy server. You can pass the same request to the HTTP server after you have extracted any information necessary for your purpose. An example of the client requesting a web page is shown below:

   When client requests www.gnu.org, your proxy will see the following request:

```
GET http://www.gnu.org/ HTTP/1.1
Host: www.gnu.org
User-Agent: Mozilla/5.0 (X11; U; SunOS sun4u; en-US; rv:1.1) Gecko/20020827
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-
mng,image/png,image/jpeg,image/gif;q=0.2,text/css,*/*;q=0.1
Accept-Language: en-us, en;q=0.50
Accept-Encoding: gzip, deflate, compress;q=0.9
Accept-Charset: ISO-8859-1, utf-8;q=0.66, *;q=0.66
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://www.gnu.org/
```

   Note the first two lines (shown in bold). These two lines are necessary in HTTP 1.1.

   - The first line is the HTTP GET request. The format of the GET request is
     GET <complete-path-of-resource> HTTP/1.1
   - The second line is the HTTP header
     Host: www.gnu.org

   A control feed and a newline character terminates each line (CRLF).

   When the browser connects to the proxy server it will send all these lines. You can use a single read statement to read this from the socket.

4. Proxy server will have to extract the hostname of the HTTP server to establish a connection with it. It can extract the hostname from the *Host: header* field in client's request. Once the proxy server opens a socket connection to the HTTP server, it can pass the same request that it received from the client.

The HTTP server will respond to proxy server's request with a HTTP response.

The format of the response is as shown:

```
HTTP/1.0 200 OK
Date: Mon, 08 Sep 2003 15:37:44 GMT
P3P: policyref="http://p3p.yahoo.com/w3c/p3p.xml", CP="CAO DSP COR CUR ADM DEV TAI PSA PSD IVAi IVDi CONi TELo
OTPi OUR DELi SAMi OTRi UNRi PUBi IND PHY ONL UNI PUR FIN COM NAV INT DEM CNT STA POL HEA PRE GOV"
Cache-Control: private
Pragma: no-cache
Expires: Thu, 05 Jan 1995 22:00:00 GMT
Connection: close
Content-Type: text/html

<data>
```

5. Note that you will not know the size of the requested web page. Hence you will have to read the response from the HTTP server in a while loop. A schematic is as follows:

```
InputStream serverInputStream = serverSock.getInputStream();
OutputStream clientOutputStream = clientSock.getOutputStream();
byte buffer[] = new byte[1024];
while((read = serverInputStream.read(buffer)) >= 0 ) {
        ostream.write(buffer, 0, read);
}
```

   - The read call returns the number of bytes read or –1. It will return –1 if EOF is reached or if the HTTP server closes the connection. The HTTP protocol is a stateless protocol. This means that the HTTP server will close the connection when it has sent all the data. Thus when the HTTP server closes the connection the read system call will return with a –1 and you will then come out of the while loop.
   - serverSock is the socket which the proxy server has opened with the HTTP server.
   - clientSock is the socket which the proxy server has opened with the client.
   - In the above while loop you read from the server and write to the client

6. Record the information about the requests in a Logfile. If the request is denied because of a disallowed domain or disallowed Content-type then the logfile should contain entries like this:

   www.cnn.com::blocked
   www.gnu.org/tux.gif::File type not allowed

   If the requested file is allowed then record the name of the cache file in the Logfile along with its size. For example:
   www.google.com::ProxyServerCache/<fname> 12450

7. Format of the local proxy cache:
   a. Make a directory called ProxyServerCache and save the requested files in this directory.
   b. You can use any format to store the files inside the directory. For example if the request is GET http://www.gnu.org HTTP/1.1 then you can create a directory inside the ProxyServerCache directory and store all the files for this site inside that directory. Or you can create a lookup table and store the file with some different name convenient to you. Basically you should be able to satisfy the request from the local cache if it is available locally.

Useful Java Classes:
Here is a list of Java classes that may be useful for this Assignment.

- Socket
- ServerSocket

- Check the following site http://java.sun.com/j2se/1.4.2/docs/api/ for additional information on these classes.
- Lecture Notes #3 are also useful. Specifically, check the following:
  a. HTTPget program
  b. EchoServer program
- The above programs are also available on the Examples page of the Class web-page.

Things to submit:
Submit a tar file with the name: assignment1.tar

The tar file (assignment1.tar) should contain:
1. Proxy.java
2. Any other Java files that you might be using

Submit your solution using the Homework Submission link on the class web page before 12 midnight on the due date.

Please include the following the information in your Proxy.java file:
1. Names
2. Student IDs
3. README. This file should contain any specific instructions for running your proxy.

Note on Comments:
Please comment your code sufficiently. We reserve the rights to deduct up to 10% of the allocated points if there are NO proper comments in the code.