Assignment 2

Computer Science 441

Due: Friday March 1, 2019 at 23:55

Instructor: Majid Ghaderi

1 Objective

The objective of this assignment is to practice server side network programming with TCP. Specifically, you will implement a multi-threaded proxy web server from scratch to serve web objects to HTTP clients over the Internet. The server can act as a web server as well as a proxy server, depending on the client requests.

2 Specification

2.1 Overview

The server functions in two modes, namely, as a web server or a proxy server. In the web serve mode, the server processes client requests and serves them locally. In the proxy server mode, the server does not process client requests, instead it simply forwards them to the appropriate remote web server, and then relays back the remote web server response to the client. The server determines in which mode to operate based on the client request. Specifically, if the `Host` header filed in the client request specifies a host that is different from the host on which the web server is running, then the request should be handled in the proxy mode.

Your web server should be able to handle more than one client connection simultaneously. For *simplicity*, the server is required to only support non-persistent HTTP requests. This means that once an HTTP request is served, in either mode, the server closes the associated TCP connection.

2.2 Proxy Server Mode

In the proxy mode, the HTTP request has a `Host` header line, which specifies a host that is different from the one your web server is running on. The format of the `Host` header is as follows:

```
Host: remote-web-server-host-name[:port]
```

where the optional `[:port]` specifies the port number of the remote web server. If a port number is not specified, then the remote web server is listening on the default port number 80. Here is a sample request with a `Host` header line:

```
GET /index.html HTTP/1.1
Host: things.cs.ucalgary.ca
Connection: close
```

In this mode, your web server establishes a TCP connection to the remote host and forwards the client HTTP request to the remote web server using non-persistent HTTP. That is, when forwarding the request, attach the following header line if it is not already included in the request:

```
Connection: close
```

Once the request is forwarded, your server should read the response from the remote web server and simply relay it back to the client with no modification (*i.e.*, byte for byte).

2.3 Web Server Mode

In the web server mode, either there is no `Host` header line in the client request, or the specified host in the request is the same as the one your web server is running on. In this mode, your server is required to handle HEAD and GET requests. For GET requests, it should be able to handle both regular as well as *Range* requests. Once a request is received, the server locates the requested object in the file system and sends an appropriate response to the client. Make sure to include the connection close header in your HTTP response:

```
Connection: close
```

As this is a simplified web server, your program needs to return responses with the following status codes only:

- `200 OK` – request was processed successfully
- `400 Bad Request` – there was a problem with format or semantic of the request
- `404 Not Found` – object was not found on the server

3 Implementation

3.1 High Level Structure

To handle multiple connections, the server has to be multi-threaded. In the main thread, the server listens on a fixed port. When it receives a TCP connection request, it sets up a TCP connection through another socket and services the request in a separate worker thread. That is, once the server accepts a connection, it will spawn a new thread to parse the incoming HTTP request and process it (locally or remotely).

The working directory of the web server is its root directory. That is, if the requested object path is `/object-path` then the file containing the object is located on relative path `object-path`

in the working directory of the web server.

Programs 1 and 2 provide a high-level description of the web server's main and worker thread functionality.

Program 1 Main Thread

```
1: while not shutdown do
2:     Listen for connection requests
3:     Accept a new connection request from a client
4:     Spawn a new worker thread to handle the new connection
5: end while
6: Wait for worker threads to finish
7: Close the server socket and clean up
```

Program 2 Worker Thread

```
1: Parse the HTTP request
2: if proxy mode then
3:     Add "Connection: close" if not included
4:     Open a TCP connection to the remote server
5:     Send the request to the remote server
6:     Read from the remote server and forward to client
7: else /*web server mode*/
8:     Ensure well-formed request (return error otherwise)
9:     Determine if requested object exists (return error otherwise)
10:    if HEAD request then
11:        Send response header lines only
12:    else /*Get request*/
13:        Send response header lines
14:        Send appropriate range of the object content
15:    end if
16: end if
17: Close the socket and clean up
```

3.2 Header Lines in Web Server Mode

To be compliant with what most web browsers expect from a well-behaving web server, include the following header lines in your server response whenever the response is `200 OK`:

- `Date` – current date on the server

- `Server` – name of your web server (your choice!)
- `Last-Modified` – get it from the file system
- `Accept-Ranges: bytes` – the server accepts range requests
- `Content-Length` – length of the file
- `Content-Type` – type of the file
- `Connection: close`

For response codes 400 and 404, include the following header lines:

- `Date`
- `Server`
- `Connection: close`

For Range requests indicate the actual range of bytes that is returned in the response by including the header line:

```
Content-Range: bytes X-Y/Z
```

where integer numbers X and Y are the first and last bytes of the range, and Z is the length of the file in bytes.

4 Software Interfaces

4.1 Server Interface

The abstract class `BasicWebServer` is provided to you. You are being asked to develop a Java class called `WebServer` that extends `BasicWebServer`. There are two abstract methods in this class, namely `run` and `shutdown`, that you need to implement in `WebServer`. A description of these methods follows:

- `void run()`
  Starts the web server in listening mode and implements Program 1.
- `void shutdown()`
  Informs the web server to shut down, clean up and exit.

4.2 Package Structure

Notice the statement

```
package cpsc441.a2;
```

at the top of `BasicWebServer`. This statement indicates that your code should be organized as a Java package with name `cpsc441.a2`.

4.3 Exception Handling

Your implementation should include exception handling code to deal with all checked exceptions in your program. Print exception messages to standard system output.

4.4 Other Files

The following files are also provided to you:

- `ServerDriver.java`: A simple driver class so that you can see how we are going to run your server.
- `WebServer.java`: A stub class to allow the driver class to compile and run. You have to write your own `WebServer` class as described in the assignment.
- `Utils.java`: Several helper methods are defined in this class. Feel free to use them in your implementation (your choice). In particular, the method `isLocalHost()` can be used to check if a given host name refers to the local host on which your web server is running. All methods have javadoc documentation.
- `README`: Includes instructions on how to compile and run your code using command line.

4.5 Testing The Server

You should be able to use Telnet, your `QuickUrl` tester from Assignment 1, or even a web browser to test your web server implementation.

Restrictions

- You are not allowed to modify class `BasicWebServer`. Your changes will be overwritten when marking your submission. However, you can (and should) implement additional methods and classes as needed in your implementation.
- You are not allowed to use class `URL` or `URLConnection` or their subclasses for this assignment. Ask the instructor if you are in doubt about any specific Java classes that you want to use in your program.