Assignment 1

Computer Science 441

Due: 23:55, Friday February 8, 2019

Instructor: Majid Ghaderi

1 Objective

The objective of this assignment is to practice network programming and learn about application layer protocols. Specifically, you will implement an HTTP client program to download a web object specified by its URL using parallel HTTP requests.

2 Specification

2.1 Overview

In this assignment, you will implement a multi-threaded HTTP client called `QuickUrl`. You have to write code to establish multiple TCP connections to a given server and send and receive HTTP messages over different connections in parallel. Your program should be able to download both binary and text objects. All downloaded objects are stored locally in the same directory where your program is running.

To download an object using parallel HTTP requests, your program should implement *HTTP Range* requests. With Range requests, a client can ask for a specific part of an object on the server. This is achieved by using the *Range* header to specify the part of the object that is being requested. Each part of the object is specified by a *byte range* indicating the index of the first and last byte of that part of the object. The first byte of the object has index 0.

2.2 Range Request

Consider an object referenced by the URL:

```
people.ucalgary.ca/~mghaderi/cpsc441/dots.txt
```

Assume the size of the the object is 200 bytes. The following HTTP request can be used to get the first half of the object:

```
GET /~mghaderi/cpsc441/dots.txt HTTP/1.1
Host: people.ucalgary.ca
Range: bytes=0-99
```

Similarly, to ge the second half of the object, the following HTTP request can be used:

```
GET /~mghaderi/cpsc441/dots.txt HTTP/1.1
Host: people.ucalgary.ca
Range: bytes=100-199
```

2.3 Server Support

To use Range requests, you need to find out if the server hosting the object supports Range requests. This can be achieved by sending a HEAD request to the server. A HEAD request has the same syntax as a GET request, but specifies the command HEAD instead of GET in the request line. The server's response to a HEAD request has all header lines of a normal GET response, but has no body. The following is an example of a HEAD request:

```
HEAD /~mghaderi/cpsc441/dots.txt HTTP/1.1
Host: people.ucalgary.ca
```

with the following response from the server:

```
HTTP/1.1 200 OK
...
Accept-Ranges: bytes
Content-Length: 200
```

In this response, the header line `Accept-Ranges:bytes` indicates that bytes can be used as unit to define a range. The `Content-Length` header is also useful as it indicates the size of the object on the server. If there is no `Accept-Ranges` header in the response, or its value is set to `none`, then the server does not support Range requests. In such cases, the entire object should be downloaded using a single GET request.

2.4 Implementation

A high-level description of the internal operation of `QuickUrl` is presented in Program 1.

The program `QuickUrl` takes as input the URL of the object and the number of parallel requests for downloading the object. The input URL is a properly formatted URL with the following format:

```
hostname[:port]/pathname
```

where, `[:port]` is an optional part which specifies the server port. If no server port is specified, use the default port 80.

To send an HTTP request, you should establish a TCP connection to the server on the specified port number. Then create a request using the `pathname` and send the request to the server.

Program 1 `QuickUrl`

Input Parameters: `url`, `conn`

```
1. Send a HEAD request for url. Parse the response to retrieve:
   (a) Accept-Ranges
   (b) Content-Length
2. If Range requests are not supported, then set conn = 1
3. Start conn threads:
   (a) Each thread opens a TCP connection
   (b) Sends a Range request for one part of the object
   (c) Each part has Content-Length/conn bytes
4. Wait for all threads to complete
5. Merge all parts to rebuild the full object
6. Clean up (e.g., close sockets, streams and remove temp files)
```

Once the request is submitted, start reading the HTTP response from the socket. The local file name of the downloaded object should match the file name specified by the URL.

Your implementation should include appropriate exception handling code to deal with various exceptions that could be thrown in the program.

3 Software Interfaces

The abstract class `ConcurrentHttp` is provided to you. You are being asked to develop a Java class called `QuickUrl` that extends `ConcurrentHttp`. There is only one abstract method, namely `getObject`, in class `ConcurrentHttp` that needs to be implemented in `QuickUrl`. A description of this method follows:

- `void getObject(String url)`
  This is the main method for downloading objects. The parameter `url` specified a properly formatted URL that specifies the object to be downloaded.

  As explained earlier, if the server supports Range requests, you should divide the specified object into `conn` parts, where all parts are download concurrently using `conn` threads. There is a set method in class `ConcurrentHttp` to set the value of field `conn`, which is defined as a private member variable.

Notice the statement

```
package cpsc441.a1;
```

at the top of `ConcurrentHttp`. This statement indicates that your code should be organized as a Java package with name `cpsc441.a1`. A simple driver class `Driver` is also provided to you, which imports the class `cpsc441.a1.QuickUrl`.

To implement class `QuickUrl`, you probably need to implement several additional classes, *e.g.*, to deal with HTTP headers, requests, and responses. Putting all your code in one class is often a good indication of poor design and coding skills.

Restrictions

- You are not allowed to modify class `ConcurrentHttp`. 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 the class `URL` or `URLConnection` for this assignment. Ask the instructor if you are in doubt about any specific Java classes that you want to use in your program.