Project #2a – Tiny HTTP Client and Server

Version 1.1

Purpose:

Distributed systems communicate between machines using a protocol implemented on top of sockets. Many different protocols have been defined and used, but in this project we will focus on the HTTP[1] protocol. That consists of messages composed of a message header with lines of text representing a command or response and optional header lines describing the message contents and appropriate processing via attributes. The message may also contain a body that may contain either text or binary data. The body length in bytes is given by a length attribute.

The intent of this project is to build HttpClient and HttpServer components, for both Windows and Linux, using sockets that enable the passing of HTTP messages between processes and machines. Note that the application could be a web server returning pages to be view in a browser, but will often be used in some other distributed application. One goal of this project is to build the smallest and simplest code possible so that these components can be included in other software to manage all of its communication activities.

As part of this project you will prepare and deliver three presentations. The first describes the technologies you are using and gives a brief description of the application you will implement. The second presents several probing projects that illustrate how the technology works. Finally, the third presentation shows your design, implementation, and demonstrates your project's capabilities.

Requirements:

For your HttpProcessing project you:

1. **shall** use standard C++ and the standard library, compile and link from the command line, using g++ within the NetBeans or Eclipse IDE and Visual C++ in the Visual Studio IDE.

2. **shall** develop an HttpServer, based on sockets, using the HTTP 1.0 protocol[2]. The server **shall** accept a function or functor that defines its response to the standard HTTP request message types: GET, POST, PUT, DELETE, and HEAD. That may be a message of type: 2xx success, 3xx redirection, 4xx not found, but may be something else, appropriate for a specific application. The message may have custom header lines that provide application specific details needed for server processing.

3. **shall** develop an HttpClient, also based on sockets, using the HTTP 1.0 protocol. This does not require you to build a browser. The HttpClient is simply responsible for sending an HTTP message to a specified URL. It should provide support for forming HTTP messages, perhaps via a lower level Message package.

4. Traditional HTTP clients always expect a response, usually consisting of a status message. Your client should normally do that, but **shall** also support a Message attribute of "OneWay" that indicates the HttpServer should not send a reply and the HttpClient will not wait for a reply.

5. **shall** develop a socket package that supports both ip4 and ip6[3]. This will be used by both the HttpClient and HttpServer.

6. **shall** provide one or more client and server applications and/or peer-to-peer applications that use the HttpClient and HttpServer for communications between processes and machines. The nature of these applications is up to you, but should be interesting and illustrate that you meet the requirements of this statement.

7. The socket packages **shall** be developed for both Linux and Windows, providing the same external interfaces in both environments. The HttpClient and HttpServer code should use either of the socket packages and otherwise be identical for both environments.

---

[1] http://www.jmarshal.com/easy/http
[2] http://en.wikipedia.org/wiki/HTTP, http://en.wikipedia.org/wiki/List_of_HTTP_status_codes
[3] http://beej.us/guide/bgnet/

You will find it helpful to look at the Man pages for System Calls on your Linux system. Those describe the semantics of each call and the header files you will need to include.