656M-P2

Introduction

In this project you will extend the proxy you built (in Project 1) to perform protocol mediation, that is, accept HTTP requests on one side and perform the actions of another protocol (FTP) on the other side. Your proxy will continue to listen for HTTP requests on one port and it will continue to handle HTTP requests as before. However, when a FTP request is received, your proxy will 'learn' how to handle this request, perform the FTP transaction and return the requested file over HTTP.

Requirements

Your proxy will speak HTTP/1.1 to the browser and plain FTP (RFC 959) on the other side. When a browser FTP request is received (e.g. GET ftp://ftpsite.com[:port]/path/to/file.dat) it will parse the request and perform the FTP transaction and obtain the file from the server and return the file as a HTTP response. Your proxy should satisfy these requirements.

- All types of files (up to a size of 1 GB) should be handled correctly. You can assume that we will test your proxy for files of .ISO, various images (JPG/GIF), compressed formats (GZ, Z, zip) as well as plain text and HTML files.
- You may assume we will specify the full path name in the URL, so directory browsing is not needed.
- As before, you cannot use fork(), threads, predefined libraries or any other parsers.
- You should handle all requests, including HTTP requests interspersed with FTP requests. Your proxy should not crash under any circumstances. Malformed requests should be ignored.

Running and testing

We will run your proxy (only *one* port) on AFS exactly like the previous project. We will use Firefox/IE on Windows. Bear in mind that your proxy will return the requested file or object on the same HTTP connection.

Grading

We will test your proxy with a combination of HTTP and FTP requests. We may use a malformed FTP request (which should be ignored) as well as FTP servers/ports specified as IP addresses. You may assume that the largest requested file will be 1GB in size. However, this project will focus more on the protocol operations rather than negative testing.

FAQ

Clarifications will be posted to the FAQ (same link as Project 1) which will remain the final arbiter of requirements and grades. The University Academic Integrity policy remains in full effect.

Note - You can implement project 2 in C or Java (both full credit). Java projects will have to follow the specifications outlined earlier - here. Remember that your Java proxy (if you decide to use Java) will also have to handle HTTP requests (e.g. Project 1).