

CS447-Network and Data Communication

Project II, Summer 2026

Due 11:00 a.m. on **July 28**, 2026

1. Introduction

We will implement “a multi-threaded application-layer security proxy” using socket interface. Proxy is a middleman between clients and a server for any client-server based network applications. Proxies are used for various purposes. For example, many proxies are used to hide a server host computer from clients to filter out unwanted incoming requests as well as to prevent denial-of-service (DoS) attacks. The concept of proxies is visualized in the following figure (Fig. 1).

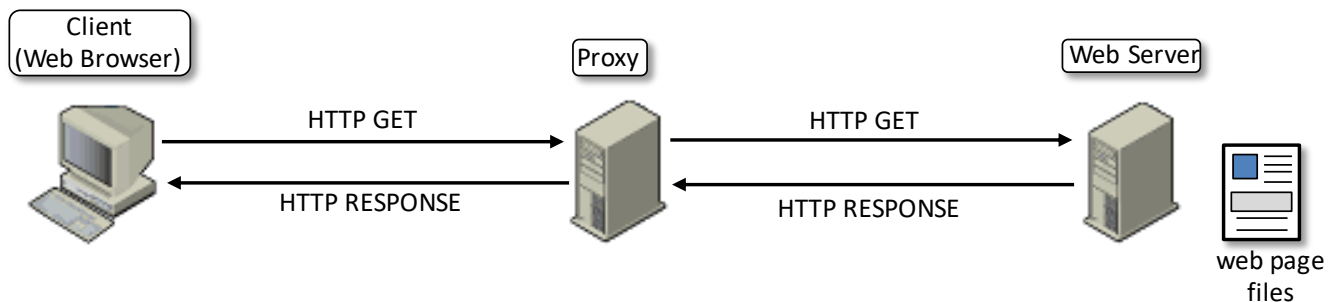


Fig. 1 – Concept of proxies

2. Requirements

The proxy we implement should satisfy the following requirements:

- The proxy accepts HTTP requests from the existing commercial web browsers (MS IE, FireFox, Netscape Navigator, and etc.) at port 9080.
- The IP address (and the port number) of a real web server can be hard coded in the proxy (should be defined as a constant using `#define` pre-compiler directives).
- As soon as the proxy accepts a new HTTP request (i.e., HTTP GET) from a remote client (a web browser), it establishes a TCP/IP connection to a web server at a remote host computer, which should be listening to port 4080 (should be defined as a constant using `#define` pre-compiler directives).
- When the remote web server responds to HTTP GET from a client, the response(s) must be forwarded to the client.
- Most of the commercial web browsers generate multiple HTTP requests in parallel. Your proxy should be able to handle such concurrent multiple HTTP requests to a remote web server (and the responses from the remote web server should be correctly processed) using multiple threads.
- No data should be dropped at the proxy.
- The proxy should work with any existing commercial web browsers and web servers.
- All the responses to a client's HTTP request must come from a web server (i.e., the proxy can not synthesize any web contents).
- The proxy should be MS Windows platform using C++ (Visual C++ and W32 environment)
- Your web proxy should work the given web server (test_server.cpp).

3. Expectations when a question is asked:

- (1) Identify where (in your source code) the problem exists
- (2) Describe the symptom(s) of the problem
- (3) Describe how the problem happens (always happen, sometime happen, the condition(s) for the problem to happen, etc.)
- (4) Describe what you tried (to understand and/or solve the problem)
- (5) Stop by Dr. Fujinoki's office (no question through e-mail).
- (6) Bring your source code (either hard copy or soft copy)

4. Required proxy organizations

The following figure shows the required proxy structure for web accesses from non-prohibited source (web users) IPs (Fig. 2).

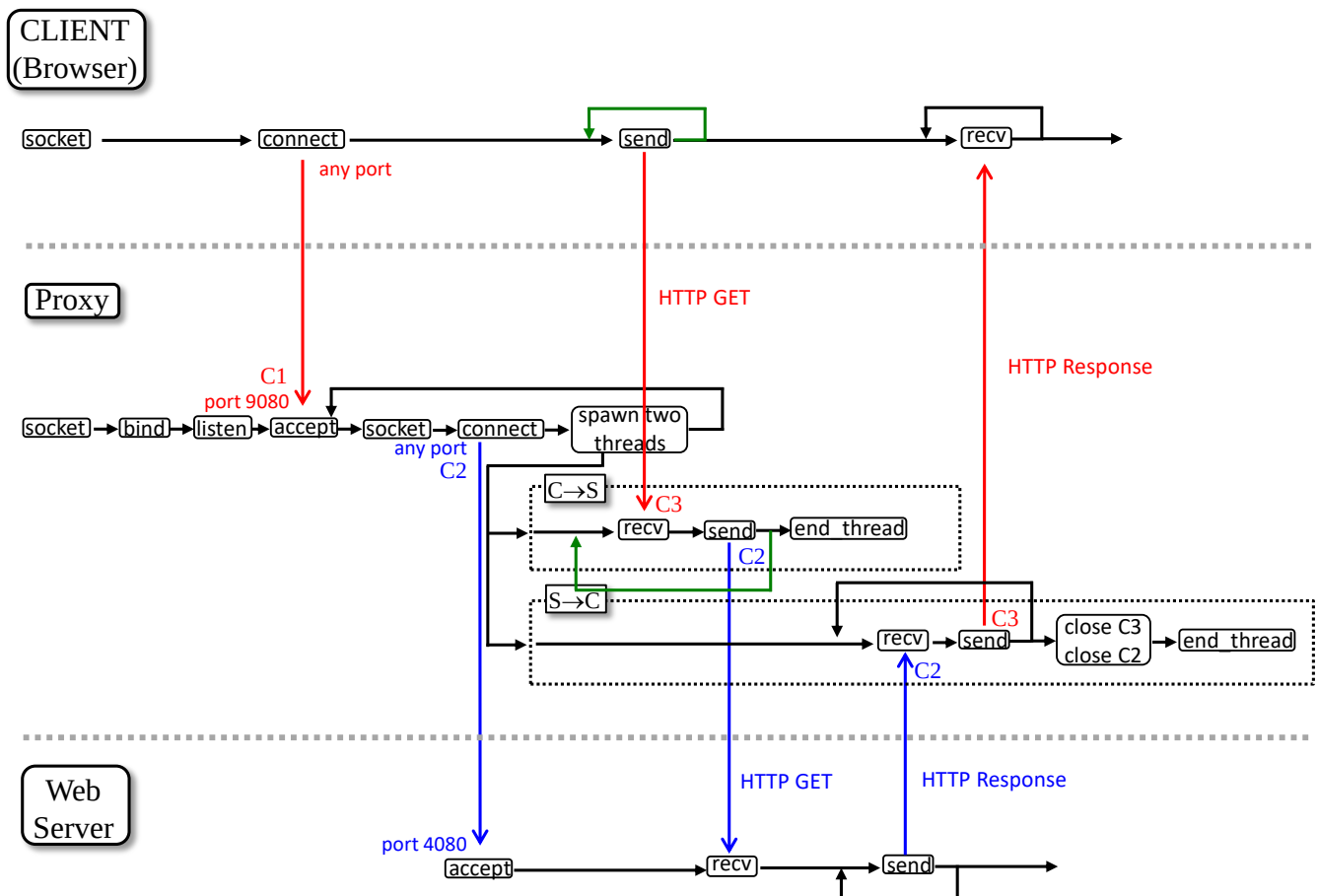


Fig. 2 – the required proxy organization for non-prohibited source (web users) IPs

The next figure (next page) shows the required proxy structure for web accesses from prohibited source (web users) IPs (Fig. 3).

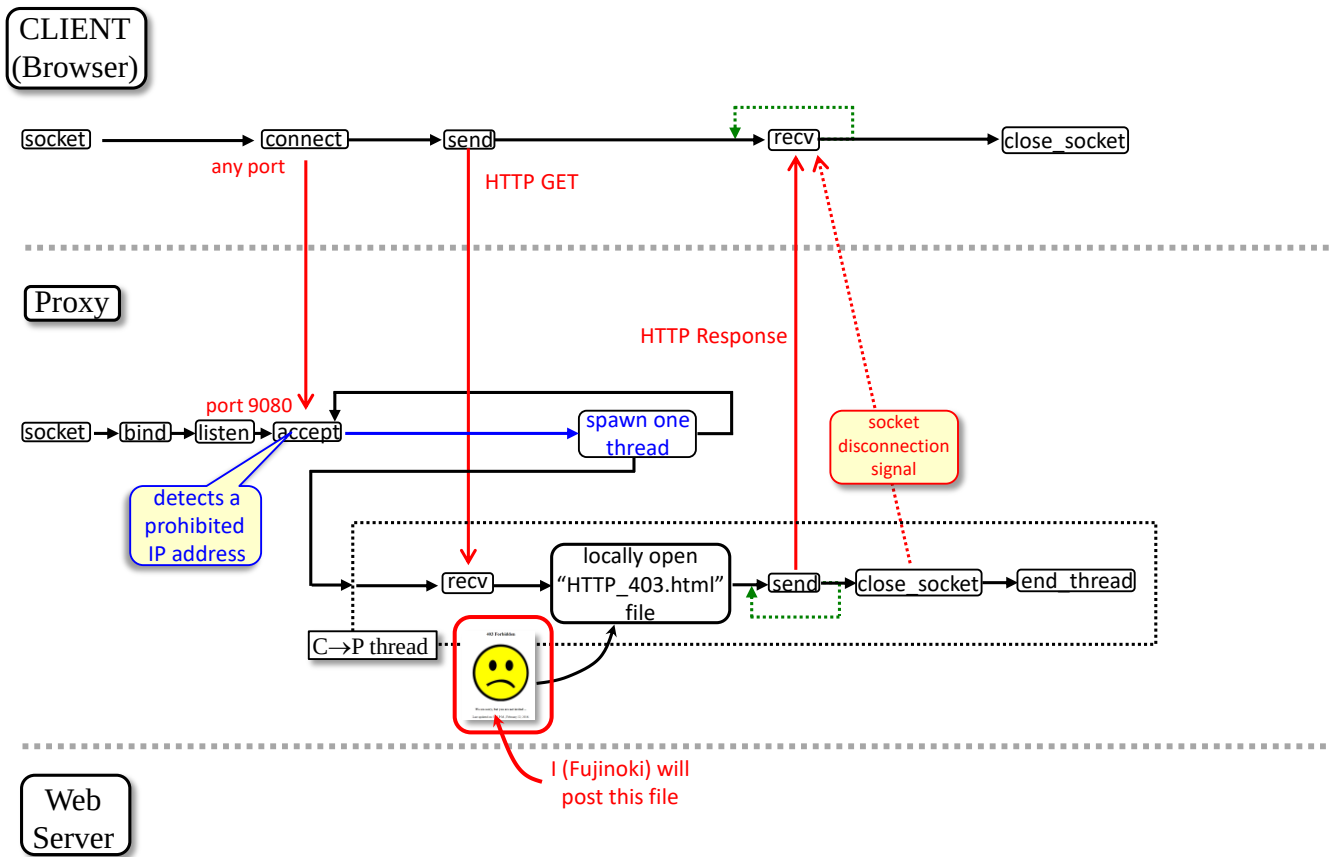


Fig. 3 – the required proxy organization for prohibited source (web users) IPs

When the proxy detects connection requests from a prohibited IP address, it responds to HTTP GET requests from those prohibited users by “HTTP 403 Forbidden” (as shown in Figure 3). The contents of the HTTP 403 response must come from the specified file, HTTP403.html”.

5. Suggested Project Milestones

The following milestones are suggested for your successful completion of this project:

- (1) **Milestone #1:** test (download the source code file, compile the source code file, and run it at your PC) the required web server (test_server.cpp) using your web browser.
- (2) **Milestone #2:** implement the required web proxy
- (3) **Milestone #3:** test your using a “monolithic web page” (a html file that does not have any link to other files)
- (4) **Milestone #4:** the security proxy for a “composite web page” (a html file that have multiple links link to other files)

- (5) **Milestone #5:** Test your web proxy for prohibited IP address (for “HTTP 403” for prohibited IP addresses (shown in Fig. 3))

*: “Monolithic web pages” are those that do not contain any other files or links.

6. Rules to avoid academic dishonesty

- (a) Sharing source code file(s) is considered academic dishonesty
- (b) Exchanging ideas through oral communications are acceptable
- (c) Regarding samples you find anywhere except in the references (the ones Dr. Fujinoki introduced in the first programming assignment), “syntax-level reference” (which should not be more than three C++ statements) is acceptable. For any case that exceeds three statements, please consult to Dr. Fujinoki.

Note: If you have anything you are not sure, please do not make your own assumptions. If you are making your own assumptions for anything you are not sure, please be responsible for your own assumptions (Dr. Fujinoki will not be responsible for your assumptions). Remember: asking a question is free (no matter what is your question) and once you got an answer from Dr. Fujinoki, Dr. Fujinoki will be responsible for whatever answer he gave to you.

7. Other issues

- For those who have technical problems to finish this project, please consult to Dr. Fujinoki as soon as possible.