Proxy Server Implementation Report

Introduction

This report aims to provide a detailed explanation of the important parts of a proxy server implementation without including actual code. The implementation is designed to handle HTTP and HTTPS requests, cache the files that are retrieved, and, if available, serve the cached files. The following sections will explain the main components and functions of the implementation in depth, focusing on key aspects of the code that contribute to the overall functionality.

Functionality

Imports and global variables:

The implementation imports a number of necessary modules to deal with various proxy server-related issues. They consist of:

socket: used to create and manage sockets that control connections between clients and distant servers.

sys: for obtaining command-line parameters and managing the script's execution.

threading: for establishing and maintaining threads that allow concurrent processing of numerous client connections.

ssl: for creating secure connections to HTTPS servers.

gzip: for processing compressed data in gzip format.

io: for handling file and memory buffers.

time: for timestamping log items.

os: to access the file system and perform operations like verifying and setting up cache folders.

Also defined in the script are global variables for timeout, buffer size, maximum file size, cache directory, and log file name settings. The proxy server's behavior is governed by these options.

Log and error functions:

Writing log messages to a log file is done by the log_message() function. This aids in monitoring the actions of the proxy server for the purpose of analysis and troubleshooting.

Whenever a problem arises while handling a request, the send_error() method is used to send a 500 Internal Server Error message to the client. This gives the client a uniform method of receiving error messages.

Validation and cache functions:

The validate_request() method determines if the client request that was received is valid. It checks to see if the HTTP version and request method are legitimate. If the request is invalid, an error message is returned. This aids in removing erroneous queries.

The check_cache() method looks for a certain file in the cache directory. It returns the file's path if the file is present. As a result, fewer queries to remote servers are necessary, and speed is enhanced since the proxy server may now provide cached files when they are accessible.

A file is saved to the cache directory using the save_to_cache() method. It guarantees that the information acquired from the distant server will be accessible for upcoming queries, which can speed up response times and use less bandwidth.

Request Handling function:

The primary method used by the proxy server is handle_request(). It takes the following actions to handle client requests:

- Validate the incoming client request using the validate_request() method.
- Use the check_cache() method to see if the requested file is present in the cache if the request is legitimate.
- Provide the file straight to the client if it has been cached.
- If the file is not cached, create a connection to the remote server, considering the URL scheme (HTTP or HTTPS) to utilize the appropriate connection technique.
- Deliver the request to the remote server and receive the answer.
- Parse the response headers to identify the status code, content length, and content encoding.
- If the status code is 200 (OK) and the file size is under the permitted limit, download the file, cache it using the save_to_cache() function, then transmit the response to the client.
- Provide suitable error messages to the client if any problems arise throughout the process, such as invalid requests, connection errors, or file size restrictions.

Main function:

The main() method configures the proxy server and monitors incoming connections from clients. It begins by confirming that the necessary command-line inputs, including the port number, are present. The server socket is then initialized, bound to the given port, and made ready to receive connections.

The handle_request() method is called by the main function for each incoming connection to handle the request. This improves performance and scalability by enabling the proxy server to manage numerous client connections concurrently.

Netcat results:

```
faruk.ulutas@Faruks-MacBook-Pro ~ % nc -l 8080
GET http://baidu.com/ HTTP/1.1
Host: baidu.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/201001
01 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
GET http://xinhuanet.com/ HTTP/1.1
Host: xinhuanet.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/201001
01 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

```
GET http://apache.org/ HTTP/1.1
Host: apache.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/201001
01 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Conclusion

Without using any real code, this article gives a thorough breakdown of the essential elements of a Python proxy server implementation. The solution is built to be reliable and efficient, processing different requests while caching files and providing them as they become available. Every function in the implementation has a specialized function, from threading and socket management to request validation and cache management. The proxy server solution offers a strong starting point for developing a scalable, high-performing proxy server that can handle a variety of use cases. Relevant netcat outputs requested in the first part of the assignment have also been added.