National Institute of Technology,Raipur

Department of Computer Science & Engineering

Proxy Server

A Term Project on Network Programming

GitHub Project Link: https://github.com/kishoretaggi007/Proxy-Server.git

Submitted By:

Roll no: 14115002	Name: Addala Srujica
Roll no: 14115010	Name: Annabathula Brahmani Sai
Roll no: 14115044	Name: Kasthala Bhanoday Sai
Roll no: 14115047	Name: Komatireddy Sowmya Lahari
Roll no: 14115089	Name: Taggi Kishore

Abstract

A proxy server is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource available from a different server and the proxy server evaluates the request as a way to simplify and control its complexity. Proxy Server is a secure gateway which you can use to provide Internet connectivity for IP and IPX based networks. A gateway is a computer that makes it possible for two networks to communicate. Proxy Server services should run on one computer to which both the private network and the public network is connected to. The computer running the Proxy Server services should have two network interfaces:

- A network interface that points to the public network.
- A network interface that points to the private network

The operations of Proxy Server are transparent to client computers. This means that users are not aware that the Proxy Server is actually requesting content on the Internet on their behalf. Users only becomes aware of the presence of the Proxy Server when they request content that the Proxy Server has been configured to disallow. The Web server servicing the request for content processes these requests as if they originated from the actual users.

Proxy Server can locally cache Internet sites and files which are frequently requested. These requests are then serviced from the local cache. This leads to an increase in Internet performance. Proxy Server can provide network address translation to support private IP addressing. Proxy Server includes a number of services which administrators can utilize to manage and control connections to the Internet. You can limit the Web sites that users can access. You can also prevent unauthorized Internet users from accessing the private network. When Proxy Server is used as a gateway to the Internet, unauthorized Internet users are basically prevented from accessing the private network. This is due to Proxy Server being the barrier between the private network and public network requests for content on the Internet is allowed, and unauthorized access from the Internet is blocked. You can however use the reverse proxy feature to provide Internet users with the ability to access Web sites on the network via the Proxy Server.

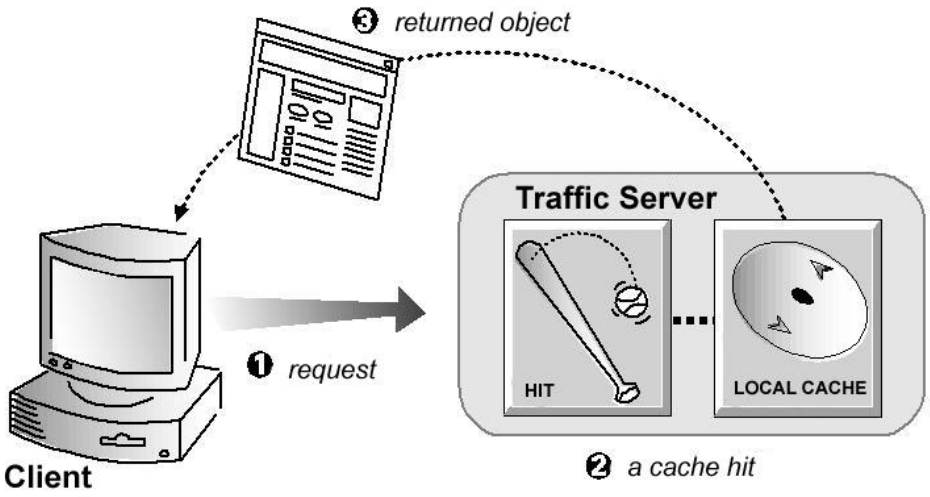
Need of caching in proxy server:

- The browser or client is misconfigured.
- The URL or Web site is down.
- Connectivity or network problems exist.
- An actual proxy server problem exists.

How Proxy Caching Works:

1. A user opens a web page containing static content such as media, JavaScript or CSS.
2. The user's browser automatically connects to a proxy server. This could be a CDN which caches content in various locations around the world.
3. The browser requests resources from the proxy server. The proxy server checks to see if it not only has the resource, but if resource is recent. If the resource is old or missing, the proxy fetches a new copy from the source.
4. The proxy delivers the resource to the browser. If the proxy had to fetch a new copy, it caches the copy for future use.

List of Figures



A Cache Hit

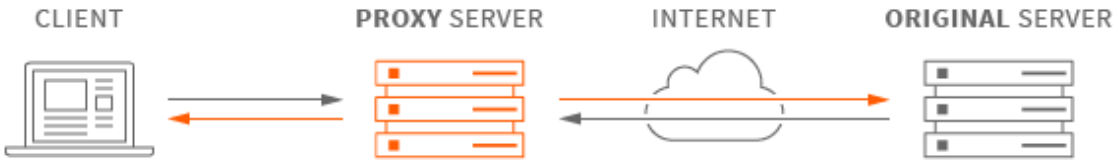
```
cybertron@trojan:~$ telnet localhost 2112
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
www.google.co.in

Reached the other error code
Connection closed by foreign host.
cybertron@trojan:~$
```

Table of Contents

Introduction

A proxy server is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource, available from a different server. The proxy server evaluates the request according to its filtering rules. If the request is validated by the filter, the proxy provides the resource by connecting to the relevant server and requesting the service on behalf of the client. It also caches responses from the remote server, and returns subsequent requests for the same content directly. We have designed an HTTP proxy-server that supports efficient caching of resources. The most prominent feature of our proxy server is its ability to cache the different objects.



Problem:

Implementation of a Proxy server which will:
1) Pass the requests from the user after filtering based on the set of filtering rules
2) Cache web pages so that subsequent access would be faster
3) Can tackle down URL or webpage
4) Can tackle connectivity or network problems

Project Outline:

The server starts on localhost using port 2112 (default port). The proxy server gets initiated and getting connection checks the URL/webpage in its cache. If the requirement is present in the proxy server's cache then it writes it to the client, else, it requests the web server. The web server on getting request from the proxy server, writes the data into its output stream. The proxy server, after getting the data, write it to both, its cache and to the client's output stream. In case the requested URL/webpage does not exists or any problem in connection establishment, the proxy server throws and catch respective exception.

Solution to the problem:

We have used Java platform for our solution. We implemented it in the following way.

i) Server.java
ii) Request.java
iii) ErrorCode.java
iv) ClientSocket.java

We created server using proxy 'localhost; and port no '2112'. This server will create an connection with any Web Server whose web page or file we want to load.

Proxy Server:

This Proxy Server is working on port number 2112. It acts as both server and client
- Server for user (client)
- Client for the Web Server

Client: The User requesting File/Webpage

According to our problem statement we have been asked to build a proxy server which will pass the request from client to server and it should have caching capabilities. So, for these features we have used Java Socket Programming and Java File Handling. As we can see in the workflow given, when the client is created we are creating input and output stream for communicating with the proxy server
- Stream from client.
- Stream to Client.

```
{
    try
    {
        // Accept new clients by starting a thread for each incoming connection
        mSocket = mProxy.accept();
        mCurrentConnections++;

        // Send message to client that the server is at it's limit for concurrent connections.
        // TODO: Maybe have a more 'official' error message
        if(mCurrentConnections > CLIENTS_LIMIT)
        {
            PrintStream outMessage = new PrintStream(mSocket.getOutputStream());
            outMessage.println("The server cannot accept anymore clients, please try again later.");
            outMessage.close();
            mSocket.close();
        }
        // Start a separate thread for each incoming client
        else
        {
            ClientSocket client = new ClientSocket(this, mSocket, UUID.randomUUID());
            Thread threadedClient = new Thread((Runnable) client);
            threadedClient.start();
            System.out.println(mCurrentConnections + " client(s) currently connected.");
        }
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

Now when the client request for a particular webpage or a file, we sent this request to the proxy server through ClientSocket. When the proxy server responds we read it through stream to client. The proxy server will read the request from the client which will be an URL. In order to process this URL we are using the by default URL class and its functionalities.

```
private void buildRequest(List<String> headers)
{
    if(headers.size() > 0)
    {
        String[] initialHeader = headers.get(0).split(" ");
        if(initialHeader.length > 2)
        {
            try
            {
                mFlag = initialHeader[2];
                if(mFlag.equals("HTTP/1.1") || !mFlag.equals("HTTP/1.0"))
                {
                    setError(" ");
                    return;
                }

                URI uri = new URI(initialHeader[1]);
                mHostName = uri.getHost();
                mPath = uri.getPath();
            }
            catch (URISyntaxException e)
            {
                try
                {
                    String[] hostHeader = headers.get(1).split(" ");
                    URI uri = new URI(hostHeader[1]);
                    mPath = initialHeader[1];
                    mHostName = uri.getHost();
```

Adding to the cache data

```
public void addPageToCache(String domain, String page)
{
    if(!mCache.containsKey(domain))
    {
        FileOutputStream outputStream = null;
        try
        {

            String path = "Files/file"+ domain.hashCode();
            File file = new File(path);
            outputStream = new FileOutputStream(file);
            byte[] content = page.getBytes(StandardCharsets.UTF_8);

            outputStream.write(content);
            //System.out.println("Hello 2");
            outputStream.flush();

            mCache.put(domain, path);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
```

```
private void sendGetRequest(String hostname, String path, String flag)
{
    try
    {
        //TODO: Port 80 needs to not be hard coded
        InetAddress address = InetAddress.getByName(hostname);
        Socket getRequest = new Socket(address, 80);

        //Send header information
        BufferedWriter writer = new BufferedWriter(
                new OutputStreamWriter(getRequest.getOutputStream(), "UTF8"));
        writer.write("GET " + path + " " + flag + CRLF);
        writer.write("Content-Type: application/x-www-form-urlencoded" + CRLF);
        writer.write(CRLF + CRLF);

        writer.flush();

        //Get Request
        BufferedReader response = new BufferedReader(new InputStreamReader(getRequest.getInputStream()));
        String line;

        // Print response to the client
        while ((line = response.readLine()) != null)
        {
            mPrintStream.println(line);
        }

        writer.close();
        response.close();
```

Caching in proxy server:

For caching two cases arises:

- The request made is for first time.
- This request has been made earlier.

In first case, when the server will send the requested data to the proxy server then the proxy server will create a new file and save that data in that file.

```
public String getCachedPage(String domain)
{
    if(mCache.containsKey(domain))
    {
        try
        {
            byte[] bytesEncoded = Files.readAllBytes(Paths.get(mCache.get(domain)));
            return new String(bytesEncoded, StandardCharsets.UTF_8);
        }
        catch (Exception e)
        {
            e.printStackTrace();
            return null;
        }
    }

    return null;
}
```

Conclusion

In this project we implemented a web proxy server with the following features:

- Multi-Client connection control
- Caching of proxy server

References

1. https://www.maxcdn.com/one/visual-glossary/proxy-caching/
2. https://docs.trafficserver.apache.org/en/4.2.x/admin/http-proxy-caching.en.html
3. https://en.wikipedia.org/wiki/Proxy_server
4. https://www.iplocation.net/proxy-server