National Institute of Technology, Raipur

Department of Computer Science & Engineering

Proxy Server

A Term Project on Network Programming

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

Submitted By:

Roll no: 14115012	Name: Antara Kale
Roll no: 14115039	Name: Hemanti Pegu
Roll no: 14115049	Name: Leesa Inhe
Roll no: 14115081	Name: Seeram Mounika

Abstract

Proxy Server are used to connect the private network or LAN to a public network such as the Internet, by acting as a gateway for internal client computers to the Internet. 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.

Table of Contents

Abstarct
1. Workflow

2. Introduction
   i. Problem
   ii. Project Outline

3. Solution to the problem
   i. HTTP Server
   ii. Proxy Server
   iii. Client
   iv. Caching of proxy server

4. Conclusion

5. References

Workflow

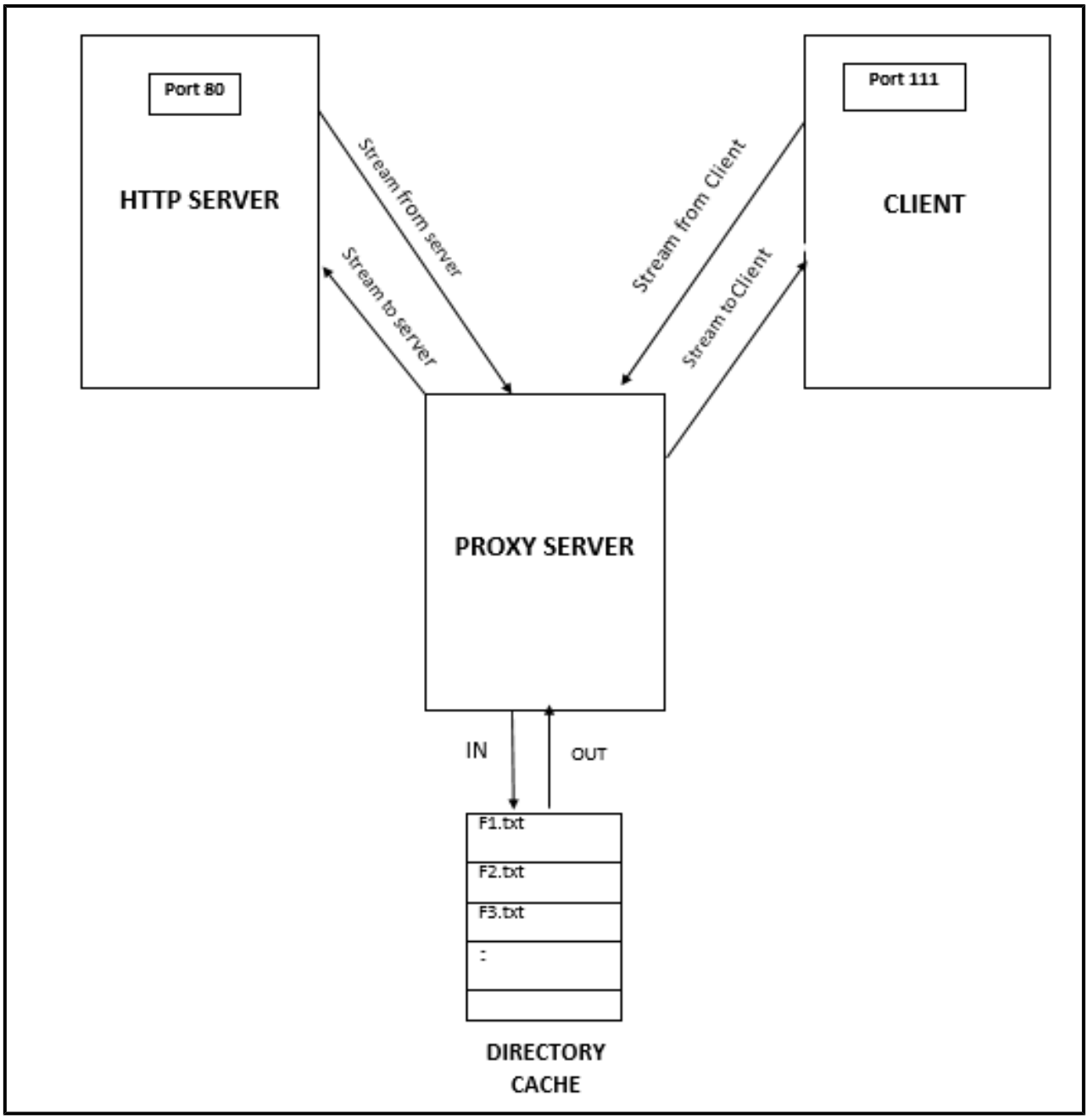

Fig. 1 Flow diagram of proxy server

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.

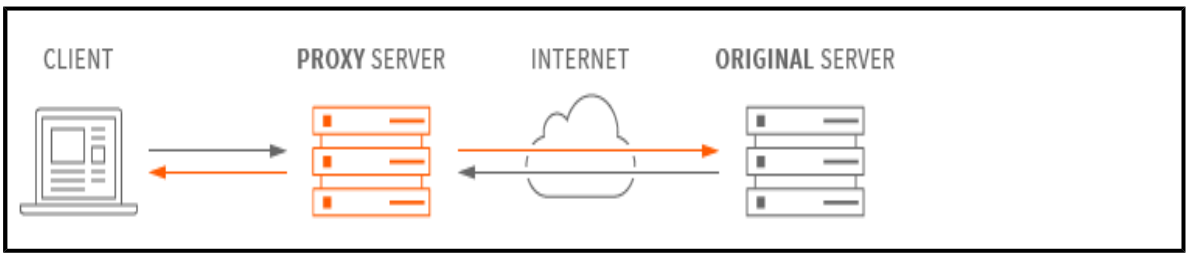

Fig 2: The proxy server

Problem: We need to implement a proxy server which will have following functionalities:

a) Cache in proxy server.
b) Can resolve misconfiguration of browser or client.
c) Can tackle down URL or webpage.
d) Can tackle connectivity or network problems.

Project Outline: The server process starts with setting up the HTTP Server which keeps looping to get connection from the client i.e. the proxy server here. The proxy server gets initiated and run in loops to take connection from client, which on 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. For that we have created two Servers:

1. HTTP Server
2. Proxy Server

HTTP Server: This HTTP Server is working on port 80. This server will create an HTTP connection with any Web Server whose web page or file we want to load.

Proxy Server: This Proxy Server is working on port number 111.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.

```
String str;
String reply;
while(true)
{
    str = stdin.readLine();
    if (str.equalsIgnoreCase("bye"))
        {
            System.out.println("Connection closed");
            break;
        }
    out.println(str);
    reply=in.readLine();
    System.out.println("Message from Server: Echo: "+reply);

}
```

Now when the client request for a particular webpage or a file, we sent this request to the proxy server through **StreamFromClient**. 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.

```
int bytesRead;
        FileOutputStream ifout=null;
        bytesRead = streamFromClient.read(request);
        streamToServer.write(request, 0, bytesRead);
        String str = new String(request);
        URL url=new URL(str);
        String file=url.getFile();
        int len=file.length();
        String filename=file.substring(1,len);
```

This request is sent to HTTP Server and at the same time being saved in a file named "**Filename.txt**". While sending the request to the HTTP Server, we are using **StreamToServe**r and we are reading the reply from **StreamFromServer.**

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 HTTP 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.

```
File f;
f=new File(naam);
if(f.createNewFile())
{
 int bytesR;
 int x;
            FileOutputStream fwriter=new FileOutputStream(naam);
            bytesR = streamFromServer.read(reply);
            streamToClient.write(reply, 0, bytesR);
            String strR = new String(reply);
            streamToClient.flush();
InputStream isnw=new ByteArrayInputStream(reply);
while((x=isnw.read())!=-1)
{

    fwriter.write(x);
}
}
```

In the second case, it will search the directory for the requested file or webpage. Using file handling it will read that file and send to the client on **StreamToClient** .

```
else
{
    int data;
    System.out.println("Reading from cache");
    FileInputStream fs=new FileInputStream(naam);
    DataInputStream iin =new DataInputStream(fs);
    BufferedReader br2=new BufferedReader(new InputStreamReader(iin));
    while ((data = br2.read()) != -1)
    {
       streamToClient.write(data);
    }

}
```

HTTP Server: Now this HTTP Server will connect with a webserver using HTTP connection class. We are checking whether the requested webpage or file exist or not using response code. If the response code is 200, it means that file is found and if it throws 404 response code it means that webpage is not found. If found it will read the content of that file and will write on **StreamFromServer.**

```
HttpURLConnection http = (HttpURLConnection)url.openConnection();
int statusCode = http.getResponseCode();
DataInputStream dinnw=new DataInputStream(http.getInputStream());
String s=dinnw.readLine();
ot.println(dinnw.readLine());
```

Conclusion

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

- client connection control
- caching of proxy server
- tackling down or unavailable URL/webpage

References

1. https://en.wikipedia.org/wiki/Proxy_server
2. https://www.iplocation.net/proxy-server
3. https://www.maxcdn.com/one/visual-glossary/proxy-caching/