****

**课程实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **课程编号** |  | **课程名称** | **计算机网络** |
| **实验学分** |  | **任课教师** | **叶松涛** |
| **学生学号** | **190320020** | **行政班级** | **信科1901** |
| **学生姓名** | **田章** | **联系方式** | **17680314099** |

**20 21 -2022学年度2学期**

**教务处制**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验** | | | | |
| **实验编号** | **实验4** | **实验名称** | **多线程Web代理服务器** | |
| **指导教师** | **叶松涛** | **实验日期** | **2022.06.08** | |
| **实验同组人** | **未分组** |  |  | |
| 1. **实验目的和要求**   **实验目的**：开发一个能够缓存网页的小型Web代理服务器。这是一个很简单的代理服务器，它只能理解简单的GET请求，但能够处理各种对象 —— 不仅仅是HTML页面，还包括图片。  **实验要求**：通常，当客户端发出一个请求时，请求将被直接发送到Web服务器。然后Web服务器处理该请求并将响应消息发送客户端。为了提高性能，我们在客户端和Web服务器之间建立一个代理服务器。现在，客户端发送的请求消息和Web服务器返回的响应消息都要经过代理服务器。换句话说，客户端通过代理服务器请求对象。代理服务器将客户端的请求转发到Web服务器。然后，Web服务器将生成响应消息并将其传递给代理服务器，代理服务器又将其发送给客户端。 | | | | |
| 1. **实验设备、软件**   实验软件：编辑器VSCode、虚拟机VMware® Workstation 16 Pro、火狐浏览器、火狐浏览器SmartProxy插件  Python环境：3.10.1 | | | | |
| 1. **实验内容（步骤、计划、方案）**   **第一步：服务器准备**  VMware中打开CentOS7.9系统，安装好Tomcat服务器，在Tomcat安装目录下的conf找到server.xml，将端口号修改为80。在Tomcat安装目录下的webapp目录下新建一个文件夹，即项目名，取名为test，在其他项目下复制一个WEB-INF到test目录下，将web.xml文件中web-app标签体内容清空，然后在test下编写两个html文件和放置一张图片以搭建一个简单的服务器。    进入Tomcat安装目录下的bin下开启Tomcat服务器    **第二步：将虚拟机的ip在windows上配置一个域名**    在windows中，进入资源管理器修改"C:\Windows\System32\drivers\etc\hosts"文件，在文件末尾添加一行  192.168.24.129 centos-server  后面我们可以使用域名centos-server来代替CentOS的IP地址  **第三步:代理服务器的设置**  打开火狐浏览器，在插件库中安装SmartProxy插件，进入选项设置  **设置代理服务器：**    **设置代理规则**：    **第四步：编写代码，下面是一些关键代码的描述**  获取URL和文件名  我们将端口号设置为终端输入，即运行代码时使用终端命令python 文件名 8888，这里是因为我们上述设置的代理服务器为本机的8888端口。获取连接后，我们需要从连接中解析出URL，而文件名我们就使用主机名+URI，其中/就创建一个目录。其中这里的消息不是请求的消息（请求消息格式是请求行+请求头+空白行+请求体，其中请求行是由请求方式+URI+HTTP协议版本号），这里的请求消息前面是请求方式+URL，我们通过split方法按空白字符分割来返回一个list，URL即为该集合的第二部分，而文件名，我们需要的是主机名+URI，即去掉http://，因此这里我们使用partition(“//”)[2]获取第一次出现//后面的部分，也即我们需要的文件名。这里简单说一下URL和URI，比如我们访问<http://centos-server/test/hello.html>  URL: <http://centos-server/test/hello.html>  URI: [/test/hello.html](http://centos-server/test/hello.html)  而我们的文件名选择的是centos-server/test/hello.html，即主机名+URI，这里/功能和windows文件路径中的\一样，即功能一样，由于\是一个转义字符，我们需要使用\\才能在字符串中识别为\，因此/更通用。    如果文件存在，即我们不需要通过服务器来获取，直接从代理服务器获取就行了，返回状态行并设置显示的文本类型（浏览器上设置为html就行），然后将文件通过二进制的方式发送，这里r表示可读，b表示二进制.    如果缓存中不存在文件，open方法会报异常，因此在except块中编写向服务器发送请求以及向客户端发送请求成功的响应行就行 | | | | |
| 1. **实验原始记录、结果分析（可以附页）**   一切准备就绪后，我们在终端输入python .\ProxyServer.py 8888，然后在浏览器中输入URL，即<http://centos-server/test/hello.html>    我们在浏览器中再发一次请求 | | | | |
| 1. **实验思考题及其解答**   **练习1 目前代理服务器不能处理错误。这可能会导致一些问题，当客户端请求一个不可用的对象时，由于“404 Not Found”响应通常没有响应正文，而代理服务器会假设有正文并尝试读取它。**  **思考：**这里比较简单，由于上面不论结果如何，我们都给客户端返回一个200 OK的状态码，这里我们只需要在代理服务器向服务器连接时，将服务器返回的相应状态码获取即可，由于相应格式是状态行+响应头+空白行+响应体，而状态行是请求方式+响应状态码，通过空格分隔开来，因此我们对相应的消息使用split方法按空格分割，返回list集合，集合中第二个元素就是状态码的数字形式了，不过这里是二进制格式，需要加一个b，因此在实验4的基础上添加以下代码就行了    **运行结果截图**  由上面搭建服务器的过程可知，test项目下只有hello.html、success.html和一张图片，并且web.xml中也没有对资源得路径进行配置，因此这里我们运行后访问一个不存在的jiwang请求，运行时我们按F12打开火狐的开发者工具进入网络选项是否变成404了呢。    **练习2当前代理服务器只支持HTTP GET方法。通过添加请求体来增加对POST的支持。**  **思考：**有了上面的基础，这题也不难，我们只需要在返回的message中获取到请求方式然后进行判断即可，也就是继续使用split方法按照空白字符来分割，获取返回集合的第一个元素就行，这里由于前端只能够发送GET请求或者POST请求，因此我们不需要两个if，直接if-else即可。另外说明一下，前端发送的请求，除了表单能够设置POST请求以外，其他所有的请求（直接输入地址、超链接、表单中指定为GET等等）都是GET请求。  **添加的代码如下**：    由于我们在服务器中的hello.html中设置了一个发送POST请求的表单，因此只需要点击表单的提交按钮即可  **运行结果截图**：  运行程序并在火狐浏览器中输入http://centos-server/test/hello.html，显示页面后我们F12打开火狐的开发者工具并进入网络选项，点击提交即可发送POST请求，并进入success.html页面 | | | |
| 1. **实验评分**   **教师签名：**    **年 月 日** | | | |

**附录：**

**由于练习1是在实验4的基础上添加代码，练习2是在实验1的基础上添加代码。因此我们附上练习2的代码即可。**

**ProxyServerPost.py**

from socket import \*

import sys

import os

if len(sys.argv) <= 1:

    print(

        'Usage : "python ProxyServer.py server\_ip"\n[server\_ip : It is the IP Address Of Proxy Server')

    sys.exit(2)

# Create a server socket, bind it to a port and start listening

tcpSerSock = socket(AF\_INET, SOCK\_STREAM)

tcpSerPort = int(sys.argv[1])

tcpSerSock.bind(("", tcpSerPort))

print(tcpSerPort)

tcpSerSock.listen(10)

while 1:

    # Strat receiving data from the client

    print('Ready to serve...')

    tcpCliSock, addr = tcpSerSock.accept()

    print('Received a connection from:', addr)

    message = tcpCliSock.recv(1024).decode()

    if(message == ''):

        continue

    # Extract the filename from the given message

    print("URL:", message.split()[1])

    filename = message.split()[1].partition("//")[2]

    print("filename:", filename)

    fileExist = False

    try:

        # Check wether the file exist in the cache

        f = open(filename, "rb")

        outputdata = f.read()

        f.close()

        fileExist = True

        # ProxyServer finds a cache hit and generates a response message

        tcpCliSock.send("HTTP/1.1 200 OK\r\n".encode())

        tcpCliSock.send("Content-Type:text/html\r\n\r\n".encode())

        tcpCliSock.send(outputdata)

        print('Read from cache')

    # Error handling for file not found in cache

    except IOError:

        if not fileExist:

            # Create a socket on the proxyserver

            c = socket(AF\_INET, SOCK\_STREAM)

            try:

                # Connect to the socket to port 80

                serverName = message.split()[1].partition(

                    "//")[2].partition("/")[0]

                serverPort = 80

                print((serverName, serverPort))

                c.connect((serverName, serverPort))

                URI = ''.join(filename.partition('/')[1:])

                print("URI:", URI)

                # Create a temporary file on this socket and ask port 80

                # for the file requested by the client

                fileobj = c.makefile('rwb', 0)

                if message.split()[0] == 'GET':

                    fileobj.write(

                        f"GET {URI} HTTP/1.1\r\nHost: {serverName}:{serverPort}\r\n\r\n".encode())

                else:  # POST

                    fileobj.write(

                        f"POST {URI} HTTP/1.1\r\nHost: {serverName}:{serverPort}\r\n\r\n".encode())

                # Read the response into buffer

                serverResponse = fileobj.read()

                if serverResponse.split()[1] == b'404':

                    print('404')

                    tcpCliSock.send("HTTP/1.1 404 Not Found\r\n\r\n".encode())

                    tcpCliSock.close()

                    continue

                # Create a new file in the cache for the requested file.

                # Also send the response in the buffer to client socket and the corresponding file in the cache

                filesplit = filename.split('/')

                for i in range(0, len(filesplit) - 1):

                    if not os.path.exists("/".join(filesplit[0:i+1])):

                        os.makedirs("/".join(filesplit[0:i+1]))

                tmpFile = open(filename, "wb")

                serverResponse = serverResponse.split(b'\r\n\r\n')[1]

                tmpFile.write(serverResponse)

                tmpFile.close()

                tcpCliSock.send("HTTP/1.1 200 OK\r\n".encode())

                tcpCliSock.send("Content-Type:text/html\r\n\r\n".encode())

                tcpCliSock.send(serverResponse)

            except:

                print("Illegal request")

            c.close()

        else:

            # HTTP response message for file not found

            print("NET ERROR")

    # Close the client and the server sockets

    tcpCliSock.close()

tcpSerSock.close()