哈尔滨工业大学

<<计算机网络>>

实验报告

(2018 年度春季学期)

姓名:	许家乐
学号:	1150310329
学院:	计算机学院
教师:	李全龙

实验一 HTTP 代理服务器的设计与实现

一、实验目的

熟悉并掌握 Socket 网络编程的过程与技术；深入理解 HTTP 协议，掌握 HTTP 代理服务器的基本工作原理；掌握 HTTP 代理服务器设计与编程实现的基本技能。

二、实验内容

(1) 设计并实现一个基本 HTTP 代理服务器。要求在指定端口（例如 8080）接收来自客户的 HTTP 请求并且根据其中的 URL 地址访问该地址所指向的 HTTP 服务器（原服务器），接收 HTTP 服务器的响应报文，并将响应报文转发给对应的客户进行浏览。
(2) 设计并实现一个支持 Cache 功能的 HTTP 代理服务器。要求能缓存原服务器响应的对象，并能够通过修改请求报文（添加 if-modified-since 头行），向原服务器确认缓存对象是否是最新版本。（选作内容，加分项目，可以当堂完成或课下完成）
(3) 扩展 HTTP 代理服务器，支持如下功能：（选作内容，加分项目，可以当堂完成或课下完成）
a) 网站过滤：允许/不允许访问某些网站；
b) 用户过滤：支持/不支持某些用户访问外部网站；
c) 网站引导：将用户对某个网站的访问引导至一个模拟网站（钓鱼）。

三、实验过程及结果

1. Socket 编程的客户端和服务器端主要步骤（以 TCP 连接为例进行使用的函数说明）

（1）客户端

①加载套接字库
WSAStartup()（但是实验中使用 python，不需要这一步，以下省略）。
②创建套接字
socket.socket(family,type[,protocal])
③向服务器发出连接请求
socket 类中的 connect(address)方法，一般 address 的格式为元组（hostname,port），如果连接出错，返回 socket.error 错误。
④和服务器端进行通信
发送数据：
socket 类中的 send(string[,flag])方法或者 sendall(string[,flag])方法，send 将 string 中的数据发送到连接的套接字。返回值是要发送的字节数量，该数量可能小于 string 的字节大小；sendall 将 string 中的数据发送到连接的套接字，但在返回之前会尝试发送所有数据。成功返回 None，失败则抛出异常。
接收数据：

socket 类中的 recv（bufsize[,flag]）方法，数据以字符串形式返回，bufsize 指定要接收的最大数据量。flag 提供有关消息的其他信息，通常可以忽略。

⑤关闭套接字

socket 类中的 close()方法。

⑥关闭加载的套接字库

WSACleanup()（python 中不需要，以下省略）

（2）服务器端

①创建套接字

使用的函数及用法与客户端的相同。

②绑定套接字到相应的 IP 地址和一个端口上

客户端不需要这一步因为操作系统会完成，使用 socket 类中的 bind(address)方法，以元组（host,port）的形式表示地址

③将套接字设置为监听模式等待连接请求

使用 socket 类中的 listen(backlog)方法，backlog 指定在拒绝连接之前，操作系统可以挂起的最大连接数量。该值至少为 1，大部分应用程序设为 5 即可。

④请求到来后，接受连接请求，得到新的对应于此次连接的套接字

使用 socket 类中的 accept()方法，并返回（conn,address）,其中 conn 是新的套接字对象，可以用来接收和发送数据。address 是连接客户端的地址。

⑤用返回的套接字和客户端进行通信

使用的函数及用法与客户端的相同。

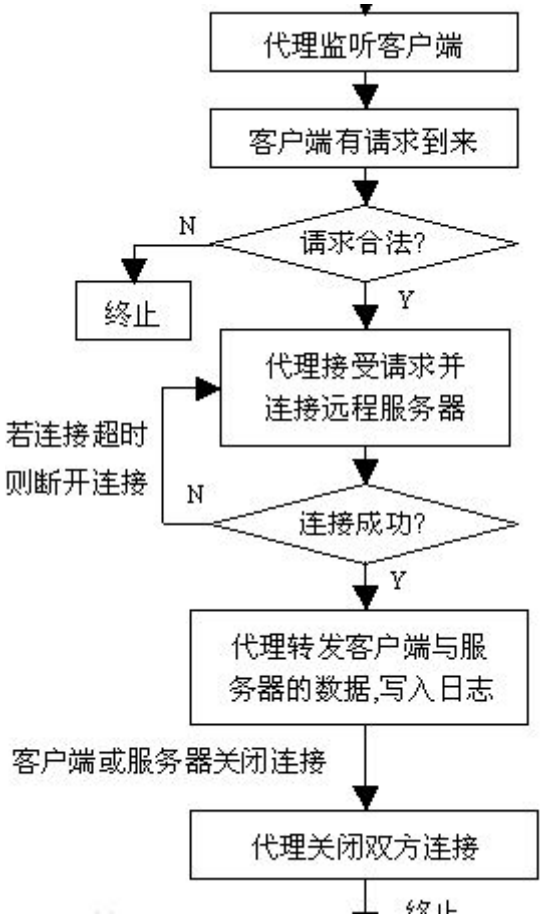
⑥关闭套接字

使用的函数及用法与客户端的相同。

2. HTTP 代理服务器的基本原理

HTTP 代理服务器是网络的中间实体,既是 Web 服务器又是 Web 客户端。代理位于 Web 客户端和 Web 服务器之间，扮演“中间人”的角色。对于客户端来说，代理扮演的是服务器的角色，接收 request 并返回 response；对于服务器来说，代理扮演的是客户端的角色，发送 request，接收 response。其通过接受客户端的请求并根据客户端的请求连接远程服务器,并将请求发送给服务器,然后再将接收到的服务器的响应发送回客户端完成其基本功能。而代理服务器的扩展功能可以通过对 已经接受 / 要发送 的来自 客户端 / 服务器端 的报文进行一定的处理来实现。

3. HTTP 代理服务器的程序流程图



4. HTTP 代理服务器实验验证过程以及实验结果

准备工作：设置 chrome 浏览器代理

（1）验证 http 代理服务器基本功能

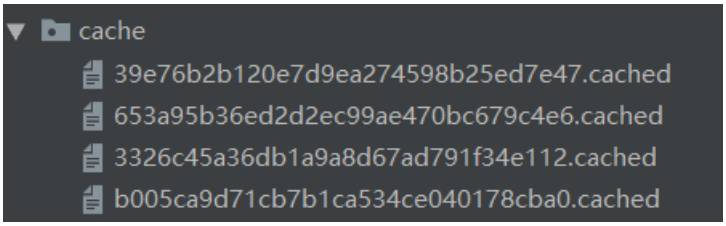
访问我的博客：http://www.xjlbest.cn，能够正常访问

访问今日哈工大：http://today.hit.edu.cn，能够正常访问

（2）验证缓存功能

访问计算机学院网站：http://cs.hit.edu.cn，观察 Cache 文件的产生


```
C:\Users\asus\Anaconda3\python.exe D:/PyCharm/PycharmProjects/ProxyServer/proxy.py
Proxy server is listening on 127.0.0.1:8888...
GET http://cs.hit.edu.cn/ HTTP/1.1
Cache miss: cs.hit.edu.cn/
GET http://cs.hit.edu.cn/themes/titan/images/main-menu.png HTTP/1.1
GET http://cs.hit.edu.cn/_upload/tpl/01/f7/503/template503/images/menu_line.png HTTP/1.1
Cache miss: cs.hit.edu.cn/themes/titan/images/main-menu.png
Cache miss: cs.hit.edu.cn/_upload/tpl/01/f7/503/template503/images/menu_line.png
GET http://cs.hit.edu.cn/_visitcount?siteId=176&type=1&columnId=8522 HTTP/1.1
Cache miss: cs.hit.edu.cn/_visitcount
```



再次访问，观察 Cache 文件的更新情况

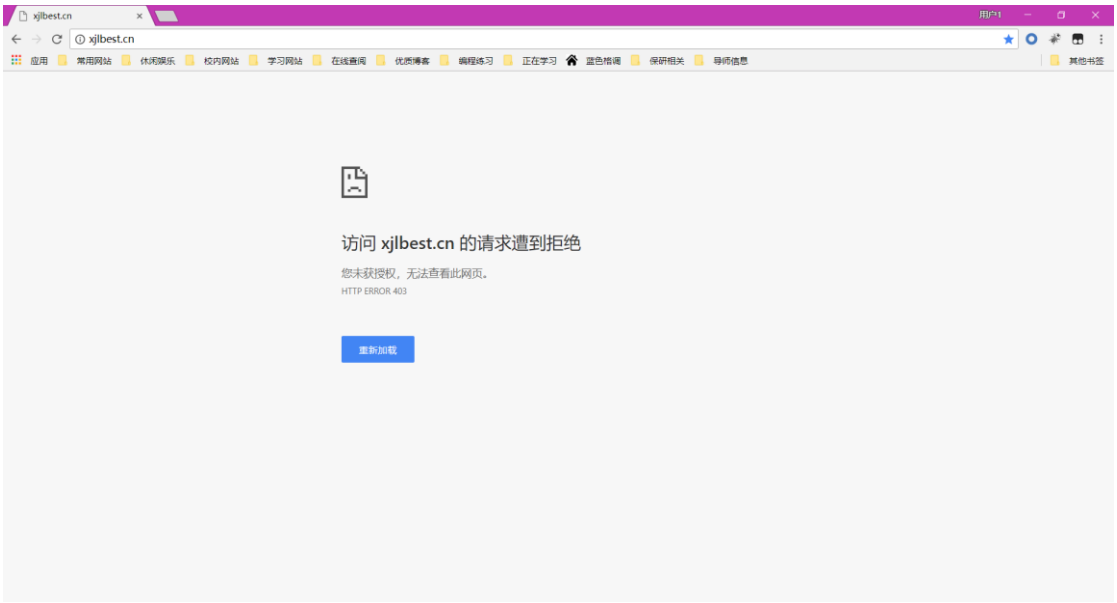
```
GET http://cs.hit.edu.cn/ HTTP/1.1
Cache updated: cs.hit.edu.cn/
GET http://cs.hit.edu.cn/themes/titan/images/main-menu.png HTTP/1.1
GET http://cs.hit.edu.cn/_upload/tpl/01/f7/503/template503/images/menu_line.png HTTP/1.1
Cache updated: cs.hit.edu.cn/_upload/tpl/01/f7/503/template503/images/menu_line.png
Cache updated: cs.hit.edu.cn/themes/titan/images/main-menu.png
GET http://cs.hit.edu.cn/_visitcount?siteId=176&type=1&columnId=8522 HTTP/1.1
Cache updated: cs.hit.edu.cn/_visitcount
```

（3）验证用户过滤功能

取消用户过滤功能的代码的注释，将本机地址 127.0.0.1 设置为禁止用户

```
33  BLOCKED_USER = [
34      '127.0.0.1'
35  ]
```

再次访问我的博客：http://www.xjlbest.cn，发现已经无法访问

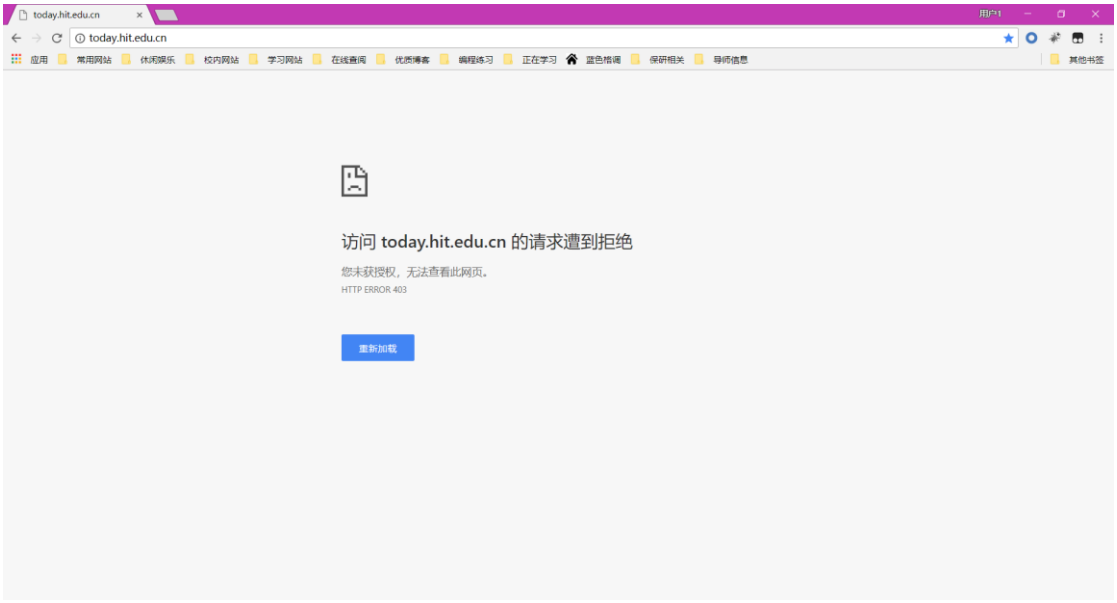


（4）验证网站过滤扩展功能

取消网站过滤功能代码的注释，将今日哈工大网址：http://today.hit.edu.cn 设置为禁止访问

```
29  BLOCKED_HOST = [
30      'today.hit.edu.cn'
31  ]
32
```

访问今日哈工大，发现已经无法访问

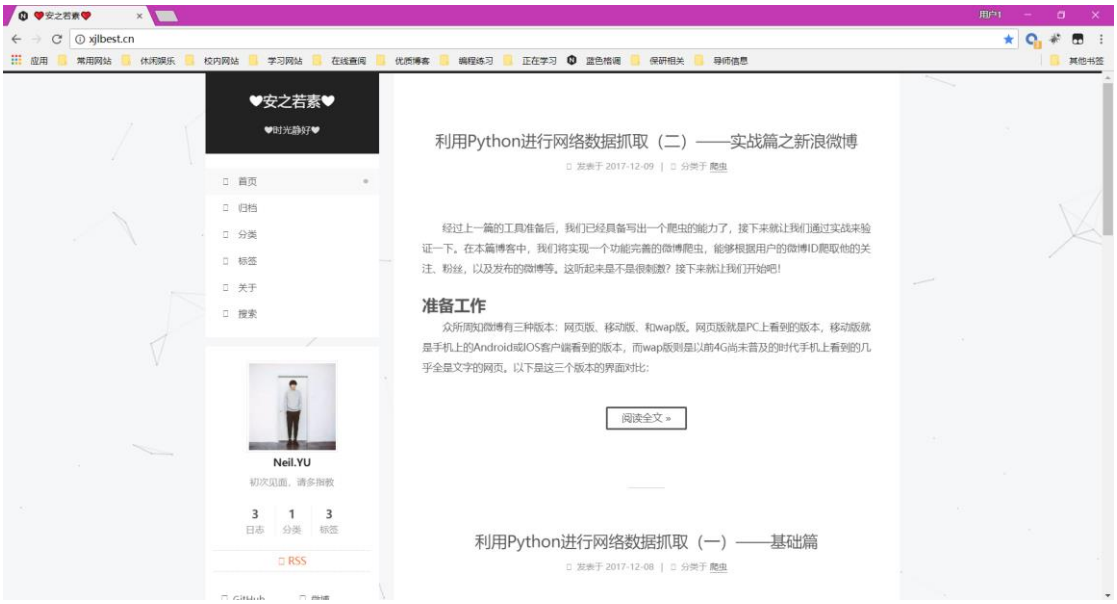


（5）验证网站钓鱼功能

首先，在代码中设置钓鱼规则，将我的博客：http://www.xjlbest.cn 引导至于晟健同学的博客：http://www.neilyu.cn

```
37      FISHING_RULE = {
38          'xjlbest.cn': 'www.neilyu.cn'
39      }
40
```

访问我的博客，发现实际上被引导到了于晟健的博客



6. HTTP 代理服务器源代码

```
import hashlib
import os
import socket
import time
import threading
```

```
from urllib.parse import urlparse


config = {
    'HOST': '127.0.0.1',
    'PORT': 8888,
    'MAX_LENGTH': 4096,
    'TIMEOUT': 100,
    'CACHE_SIZE': 100
}

CACHE_DIR = os.path.join(os.path.dirname(__file__), 'cache')
if not os.path.exists(CACHE_DIR):
    os.mkdir(CACHE_DIR)

BLOCKED_HOST = [
    # 'today.hit.edu.cn'
]

BLOCKED_USER = [
    # '127.0.0.1'
]

FISHING_RULE = {
    'xjlbest.cn': 'www.neilyu.cn'
}


def isHostBlocked(host):
    if host in BLOCKED_HOST:
        return True
    return False


def isUserBlocked(user):
    if user in BLOCKED_USER:
        return True
    return False


class ProxyServer:
    def __init__(self, host=config['HOST'], port=config['PORT']):
        self.serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.serverSocket.bind((host, port))
```

```
        self.serverSocket.listen(50)
        self.host = host
        self.port = port

    def start(self):
        print('Proxy server is listening on {host}:{port}...'.format(
            host=self.host, port=self.port
        ))
        while True:
            connect, address = self.serverSocket.accept()
            proxyThread = threading.Thread(target=self._proxyThread, args=(connect,
address))
            proxyThread.start()

    @staticmethod
    def _proxyThread(connect, address):
        request = connect.recv(config['MAX_LENGTH'])
        if len(request) == 0:
            return
        http = request.decode().split('\n')[0]
        if http.startswith('CONNECT'):
            return
        print(http)

        url = urlparse(http.split()[1])

        if url.hostname is None:
            connect.send(str.encode('HTTP/1.1 404 Not Found\r\n'))
            connect.close()
            return

        if isHostBlocked(url.hostname):
            connect.send(str.encode('HTTP/1.1 403 Forbidden\r\n'))
            connect.close()
            return

        if isUserBlocked(address[0]):
            connect.send(str.encode('HTTP/1.1 403 Forbidden\r\n'))
            connect.close()
            return

        if url.hostname in FISHING_RULE.keys():
            temp = request.decode().replace(url.hostname, FISHING_RULE[url.hostname])
            request = str.encode(temp)
```

```
        port = 80 if url.port is None else url.port

        m = hashlib.md5()
        m.update(str.encode(url.netloc + url.path))
        filename = os.path.join(CACHE_DIR, m.hexdigest() + '.cached')
        if os.path.exists(filename):
            forwardSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            forwardSocket.settimeout(config['TIMEOUT'])
            forwardSocket.connect((url.hostname, port))

            temp = http + '\n'
            t = (time.strptime(time.ctime(os.path.getmtime(filename)),
                               "%a %b %d %H:%M:%S %Y"))
            temp += 'If-Modified-Since: ' + time.strftime(
                '%a, %d %b %Y %H:%M:%S GMT', t) + '\n'
            for line in request.decode().split('\n')[1:]:
                temp += line + '\n'

            forwardSocket.sendall(str.encode(temp))

            first = True
            while True:
                data = forwardSocket.recv(config['MAX_LENGTH'])
                if first:
                    if data.decode('iso-8859-1').split()[1] == '304':
                        print('Cache hit: {path}'.format(path=url.hostname + url.path))
                        connect.send(open(filename, 'rb').read())
                        break
                    else:
                        o = open(filename, 'wb')
                        print('Cache updated: {path}'.format(path=url.hostname +
url.path))
                        if len(data) > 0:
                            connect.send(data)
                            o.write(data)
                        else:
                            break
                        first = False
                else:
                    o = open(filename, 'ab')
                    if len(data) > 0:
                        connect.send(data)
                        o.write(data)
```

```
                else:
                    break
        else:
            print('Cache miss: {path}'.format(path=url.hostname + url.path))
            forwardSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            forwardSocket.settimeout(config['TIMEOUT'])
            forwardSocket.connect((url.hostname, port))

            forwardSocket.sendall(request)

            o = open(filename, 'ab')
            while True:
                data = forwardSocket.recv(config['MAX_LENGTH'])
                if len(data) > 0:
                    connect.send(data)
                    o.write(data)
                else:
                    break
            o.close()

        connect.close()
        forwardSocket.close()

        cacheCounter = 0
        cacheFiles = []
        for file in os.listdir(os.path.join(os.path.dirname(__file__), 'cache')):
            if file.endswith('.cached'):
                cacheCounter += 1
                cacheFiles.append(file)
        if cacheCounter > config['CACHE_SIZE']:
            for i in range(len(cacheFiles)-1):
                for j in range (i+1, len(cacheFiles)):
                    if os.path.getmtime(cacheFiles[i]) <
os.path.getmtime(cacheFiles[j]):
                        temp = cacheFiles[i]
                        cacheFiles[i] = cacheFiles[j]
                        cacheFiles[j] = temp
            for file in cacheFiles[config['CACHE_SIZE']:]:
                os.remove(file)

    def stop(self):
        mainThread = threading.current_thread()
        for thread in threading.enumerate():
            if thread is mainThread:
```

```
                continue
            thread.join()
        self.serverSocket.close()
        exit(0)


if __name__ == '__main__':
    server = ProxyServer()
    server.start()
```

四、实验心得

通过本次实验，我有以下几点收获：

①更加深入地理解了 HTTP 协议，掌握了 HTTP 代理服务器的基本工作原理；

②掌握了 HTTP 代理服务器设计与编程实现的基本技能；

③熟悉并掌握了 Socket 网络编程的过程与技术；

④学会使用 Python 语言的库进行 socket 编程。