

**计算机网络**

**课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | HTTP 代理服务器的设计与实现 | | | | | |
| 姓名 | 石卓凡 | | 院系 | 软件学院 | | |
| 班级 | 2037101 | | 学号 | 120L021011 | | |
| 任课教师 | 李全龙 | | 指导教师 | 李全龙 | | |
| 实验地点 | 格物207 | | 实验时间 | 2022.10.7 | | |
| 实验课表现 | 出勤、表现得分(10) |  | 实验报告  得分(40) |  | 实验总分 |  |
| 操作结果得分(50) |  |
| 教师评语 | | | | | | |
|  | | | | | | |

****

|  |
| --- |
| 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。 |
| 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 |
| 实验过程： |
| 以文字描述、实验结果截图等形式阐述实验过程，必要时可附相应的代码截图或以附件形式提交。   1. Socket 编程的客户端和服务器端主要步骤；   **TCP客户端与服务器端的原理说明：**  服务器端   1. Socket（）实例化创建 2. Bind（IP，PORT）绑定IP和PORT 3. Listen（）使得socket处于监听状态 4. Accept（）进入阻塞状态等待连接 5. Recv()接收数据 6. Send（）发送数据 7. Close()关闭   客户端：   1. socket（）初始化socket对象 2. Connect（IP,PORT)连接服务器 3. Send（）给服务器发送数据 4. Recv（）接收数据 5. Clouse（）关闭     **具体TCP流程图：**  原图位于’./TCP客户端和服务器端流程.png’    （2）HTTP 代理服务器的基本原理：  **代理服务器的概念**  一台主机想要访问某个网址，该主机不会直接向对应的服务器请求该网址，而是通过一个中间服务器，主机告诉中间服务器我想要哪个网址的内容，中间服务器代替主机去请求该网址，中间服务器得到响应后将该响应发送给主机。这里的中间服务器就是代理服务器(****Proxy Server****)  IMG_256  **代理服务器的实现原理：**  代理服务器启动并且设置一个serversocket，  然后客户端启动clientsocket并且连接代理服务器端，  代理服务器获得这个客户端对应的datasocket此后可以通过datasocket向客户端发送数据，  客户端连接之后通过clientsocket发送请求，  代理服务器通过serversocket收到客户端的请求，将请求转发给目的网站，然后serversocket收到目的网站返回的响应报文，  代理服务器通过datasocket将响应报文发回给客户端  客户端clinetsocket接受代理服务器发来的数据   1. HTTP 代理服务器的程序流程图；   原图位于’./代理服务器程序流程图.png’  代理服务器   1. 实现 HTTP 代理服务器的关键技术及解决方案；   **代理服务器功能**  1.设定各参数  bind，listen初始化服务器socket  利用while True死循环等待连接，接到连接之后，发起多线程工作    进入socket\_tcp(dataSocket, address, threadId)  dataSocket对应的客户端socket  Address对于的客户端IP地址和客户端端口  ThreadId线程Id  接受请求报文message，然后将bytes解码为字符串，然后通过parseToHostnameAndPort(message)获得目标host和目标port    进入parseToHostnameAndPort(message)：  Message请求报文  由于请求报文格式固定，每一行结尾都是\r\n，所以可以拿到url  其中有两种情况  情况1 today.hit.edu.cn:443（https后续无需考虑） 情况2 cs.hit.edu.cn (默认端口80)  再次利用split(:)划分，拿到host和port并返回    返回继续执行socket\_tcp（）  **1.用户屏蔽功能：**  设定过滤IP    然后如果本次线程中的hostIP在应该被屏蔽的IP里面则  Socket.close()  关闭socket禁止访问    **2.网站屏蔽功能（白名单和黑名单功能）：**  设定白名单和黑名单    如果host解析出来之后，去检测是否在白名单和黑名单，然后根据内容进行处理    **3.钓鱼网站功能：**  判断host是否在钓鱼网站    如果是，则通过map拿到应该钓鱼目的host    将message中的原host全改为钓鱼目的host    利用修改后的message发送    代理服务器接受钓鱼目的服务器发回数据，然后转发给客户端    **4.实现缓存，非钓鱼情况：**  通过拿到的url格式化为文件filename  比如http://cs.hit.edu.cn/page/jquery2.js格式化为cshiteducnpagejquery2js  最后路径为path = cache\_dir + filename + '.txt'    path对应位置比如    利用modified判断是否服务器资源是否修改，是否可以读取代理服务器缓存  如果path路径已经存在文件，代表非第一次访问    获取path路径下系统的最后修改时候  然后按照报文格式将if-modified-since插入报文    利用新报文message向目的服务器发送，然后只需要获取头部内容    利用获取到头部内容，利用响应报文固定格式拿到响应码  如果响应码为304，则读取path路径下的缓存，以二进制方式读，并且socket.sendall()发送给客户端，实现读取缓存  如果不是304响应码，则将modified改为true代表服务器资源已修改需要继续执行    如果发现path路径为空，或者刚才的modified为true代表需要目的服务器给的数据    直接用原message报文发送    通过while死循环中的recv保证能够全部接受到socket发过来的数据，如果缓存buff为空代表目标服务器发送完毕，可以退出while关闭连接  接送到的数据写入path下的缓存，并且通过socket.sendall()发送给客户端    关闭socket完成所有功能 |
| 实验结果： |
| 采用演示截图、文字说明等方式，给出本次实验的实验结果。   1. HTTP 代理服务器实验验证过程以及实验结果；   **基本 HTTP 代理服务器功能：**  1打开代理服务器，ip设为127.0.0.1,端口为10086    2运行代理服务器，等待客户端连接    3打开浏览器访问cs.hit.edu.cn  发现网页正常访问连接成功  4控制台显示日志，代理服务器工作，进行访问请求，转发数据      **验证Cache功能：**   1. 关闭浏览器缓存      1. py自动建立空MyCache文件夹     2访问jwts.hit.edu.cn网页等待加载    3.首次访问成功    4控制台日志显示首次访问完毕，更新缓存    生成Mycache中的缓存文件    5.再次访问jwts.hit.edu.cn网页    6发现响应码为304，直接读取的缓存内容    **测试屏蔽用户IP:**  1.打开过滤功能    2.以127.0.01IP访问任意网页均显示无法使用    3日志显示：用户IP被禁止访问    **钓鱼功能测试：**  1.指定被钓鱼网站以及钓鱼目的网站    2.访问jwts.hit.edu.cn发现拿到的内容是cs.hit.edu.cn  3.查看控制台日志发现，启动钓鱼功能，代理服务器将请求报文的jwts.hit.edu.cn修改为cs.hit.edu.cn      4.钓鱼成功    测试网站过滤功能（白名单/黑名单）：  1.将info.cern.ch移除白名单    2发现无法访问    3控制台日志  黑名单同理 |
| 问题讨论： |
| Q1:实现钓鱼网站时候，需要在代理服务器修改请求报文中的URL和Host，但是第一次修改之后（比如http://jwts.hit.edu.cn改为http://cs.hit.edu.cn)并发回客户端，客户端后续发回的请求报文仍是http://jwts.hit.edu.cn/page/jquery2.js，而不是http://cs.hit.edu.cn/page/jquery2.js  A1:本来以为代码错了，后面思考认为，在客户端中被代理服务器钓鱼到cs网站，但是客户端以为是jwts，所以在后续发出的请求js等报文中，由客户端填写为cs.hit.edu.cn的host，还需要再次被代理服务器修改  Q2:在进行缓存时候，将某个网站的一系列的请求缓存文件比如http://cs.hit.edu.cn与http://cs.hit.edu.cn/page/jquery2.js都以cshiteducn.txt缓存是否可行  A2:个人尝试之后，发现这种想法不够好，当再次访问该网站时候，很难实现响应304，因为如果某一个比如http://cs.hit.edu.cn的请求中不满足读取缓存要求，则会修改txt缓存，由于多线程打开同一个文件机制，可能会有多线程安全问题，较为复杂。  不如以cshiteducnpagejquery2js再单独存一份靠谱  Q3:有时候无法访问www.baidu.com,拿到的是443端口，有时候可以访问www.baidu.com拿到80端口  A3:现在很多网站都支持http和https同时存在,而https是443端口需要使用SSL/TLS证书进行加密传输，按照指导书上说明无需考虑443端口 |
| 心得体会： |
| 结合实验过程和结果给出实验的体会和收获。   1. 明白了http和https的区别所在 2. 对于socket有了更深层次的体会和理解，对于利用socket作代理服务器，作为网络编程的技术更加的精进了些 3. 再次复习了代理服务器的原理和机制 |

附录：程序源代码

import os  
import socket  
import threading  
import time  
from urllib.parse import urlparse  
  
# 代理服务器相关参数  
PARAMETERS = {  
 'HOST': '127.0.0.1',  
 'PORT': 10086,  
 'MAX\_LISTEN': 50,  
 'MAX\_LENGTH': 4096,  
 'CACHE\_SIZE': 65507  
}  
# 仅可访问白名单  
White\_url = [  
 'cs.hit.edu.cn', # 可访问，访问快  
 'jwts.hit.edu.cn', # 可访问且可以缓存，但是访问慢  
 # 'today.hit.edu.cn', # 这里默认是https，需要人工加上http，但是显示全字符  
 'software.hit.edu.cn', # 可访问  
 'example.com',#可访问且可以缓存，多刷新几次，还挺快  
 'www.7k7k.com', # 可访问但是补全还有关联的子host "i1.7k7kimg.cn"  
 'www.7k7kjs.cn',  
 # 'info.cern.ch'  
]  
  
# 禁止访问的黑名单  
Black\_url = [  
 # 'today.hit.edu.cn'  
]  
  
# 过滤用户IP  
Blocked\_User\_IP = [  
 # '127.0.0.1'  
]  
  
# 钓鱼  
fishingWeb = {  
 #访问快  
 'jwts.hit.edu.cn': 'cs.hit.edu.cn'  
}  
  
# 缓存目录  
cache\_dir = './MyCache/'  
  
"""  
从请求报文来获得hostname,port  
"""  
  
  
def parseToHostnameAndPort(message):  
 message\_line = message.split('\r\n')[1].split()  
  
 # 情况1 today.hit.edu.cn:443  
 # 情况2 cs.hit.edu.cn (默认端口80)  
 target\_host\_with\_port = message\_line[1]  
  
 target\_host\_line = target\_host\_with\_port.split(':')  
 target\_host\_without\_port = target\_host\_line[0]  
  
 # 情况1 today.hit.edu.cn:443  
 if (len(target\_host\_line) == 2):  
 target\_port = int(target\_host\_line[1])  
 else: # 情况2 cs.hit.edu.cn,默认端口80  
 target\_port = 80  
  
 return target\_host\_without\_port, target\_port  
  
  
def socket\_tcp(dataSocket, address, threadId):  
 *"""  
 建立TCP连接* ***:param*** *dataSocket: socket* ***:param*** *address: IP地址和端口号组成的元组* ***:return****: 无  
 """* # 接受来自客户端的http请求报文  
 message = dataSocket.recv(PARAMETERS['MAX\_LENGTH'])  
  
 # 如果返回空bytes，请求报文为空，不需要处理  
 if len(message) == 0:  
 return  
  
 # 读取的字节数据是bytes类型，需要解码为字符串  
 message = message.decode('utf-8', 'ignore') # 对报文进行解码，忽略错误  
  
 # print("message:")  
 # print(message)  
 # message.split('\r\n')[0]拿到请求头  
  
 # GET http://www.sina.com/ HTTP/1.1  
 request\_line = message.split('\r\n')[0].split() # 获得请求行，去掉前后空格\  
  
 target\_host\_without\_port, target\_port = parseToHostnameAndPort(message)  
 # print('------target\_host：------')  
 # print(target\_host\_without\_port)  
 # print(target\_port)  
 # print('------target\_host：------')  
  
 # message.split('\r\n')[0].split()再以空格切分，request\_line[1]拿到url地址  
 target\_url = request\_line[1]  
 # 拿到情况1，http://cs.hit.edu.cn/  
 # 情况2，today.hit.edu.cn:443  
  
 url = urlparse(request\_line[1]) # 获得URLparse划分以辅助对比  
 hostIP = address[0] # 获得原主机IP  
  
 if hostIP in Blocked\_User\_IP: # 用户IP被过滤  
 print('线程:' + str(threadId) + '---------用户已被禁止访问------------')  
 print('线程:' + str(threadId) + '-->' + '该用户的IP' + str(hostIP) + '已经被代理服务器禁止')  
 print('线程:' + str(threadId) + '---------用户已被禁止访问------------')  
 dataSocket.close()  
 return  
  
 if target\_host\_without\_port in Black\_url: # 主机名被禁止访问  
 print('线程:' + str(threadId) + '-->' + str(target\_host\_without\_port) + '在黑名单，禁止访问')  
 dataSocket.close()  
 return  
  
 # 只允许白名单内的可以被访问  
 if target\_host\_without\_port not in White\_url:  
 # print('线程:' + str(threadId) + '-->' + str(target\_host\_without\_port) + '不在白名单，禁止访问')  
 dataSocket.close()  
 return  
  
 # if target\_host\_without\_port in Black\_url:  
 # print('线程:' + str(threadId) + '-->' + str(target\_host\_without\_port) + '黑名单，禁止访问')  
 # dataSocket.close()  
 # return  
 print('------target\_host：------')  
 print("目的host:" + target\_host\_without\_port)  
 print("目的port:" + str(target\_port))  
 print('------target\_host：------')  
 # print('线程:' + str(threadId) + '-->' + '真实要访问的地址' + str(target\_url))  
 # print('线程:' + str(threadId) + '-->' + 'urlparse为：' + str(url))  
 # print(request\_line[1])  
  
 # 钓鱼情况  
 if target\_host\_without\_port in fishingWeb: # 主机名为钓鱼网站  
 fishing\_host\_without\_port = fishingWeb[target\_host\_without\_port]  
 # http默认80  
 fishing\_host\_port = 80  
  
 print('线程:' + str(threadId) + '---------钓鱼启动------------')  
 print('线程:' + str(threadId) + '-->' + '对' + str(target\_host\_without\_port) + '进行钓鱼，钓鱼到-> ' + str(  
 fishing\_host\_without\_port))  
  
 print('------------改之前')  
 print(message)  
 # # 更换报文头请求地址  
 # message = message.replace(request\_line[1], 'http://' + fishing\_host\_without\_port + '/')  
 # 把host全改成钓鱼的  
 message = message.replace(target\_host\_without\_port, fishing\_host\_without\_port)  
 print('------------改之后')  
 print(message)  
  
 fishDataSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) # 初始化钓鱼网站的socket  
 fishDataSocket.connect((fishing\_host\_without\_port, fishing\_host\_port)) # 与钓鱼网站的服务器建立连接  
 fishDataSocket.sendall(message.encode()) # 将报文编码发送到钓鱼网站服务器  
  
 print('线程:' + str(threadId) + '-->' + '正在从钓鱼服务器接收数据并转发,请稍后')  
 while True:  
 # 从钓鱼目的服务器接收数据,转发给客户端  
 buff = fishDataSocket.recv(PARAMETERS['MAX\_LENGTH'])  
 if not buff:  
 fishDataSocket.close()  
 break  
 dataSocket.sendall(buff)  
 print('线程:' + str(threadId) + '-->' + '接收数据完毕')  
 dataSocket.close()  
 fishDataSocket.close()  
 print('线程:' + str(threadId) + '---------钓鱼结束------------')  
 return  
  
 #######  
 # 非钓鱼情况  
 if target\_host\_without\_port is not None:  
 print('线程:' + str(threadId) + '---------代理服务器工作开始------------')  
 print('线程:' + str(threadId) + '-->' + 'target:目标host:' + target\_host\_without\_port,  
 end=' ')  
 print('线程:' + str(threadId) + '-->' + 'port:目标端口:' + str(target\_port),  
 end=' ')  
 print('线程:' + str(threadId) + '-->' + 'source:客户端hostIP:' + hostIP)  
 print('线程:' + str(threadId) + '-->' + '真实要访问的地址' + str(target\_url))  
 print('线程:' + str(threadId) + '-->' + 'urlparse为：' + str(url))  
  
 if target\_host\_without\_port is None: # 主机名为空  
 print('线程:' + str(threadId) + '-->' + 'target\_host\_without\_port为空,关闭该连接')  
 dataSocket.close()  
 return  
  
 # 拿到情况1，http://cs.hit.edu.cn/  
 # 情况2，today.hit.edu.cn:443  
  
 filename = target\_url.replace(":", '').replace('.', '').replace('/', '').replace('?', '').replace('\*', '').replace('<', '').replace('>', '').replace('|', '')  
 path = cache\_dir + filename + '.txt' # 组合缓存路径和文件名  
 modified = False # 第一次标记为未修改  
  
 # 不是第一次访问该网页  
 if os.path.exists(path): # 当已经存在该文件，需要判断服务器是否修改过此网页  
  
 # 在path路径下调用系统stat拿到最后修改时间  
 modified\_time = os.stat(path).st\_mtime # 缓存文件最后修改的时间  
  
 # 把modified-time按报文要求格式化  
 headers = str('If-Modified-Since: ' + time.strftime('%a, %d %b %Y %H:%M:%S GMT', time.gmtime(modified\_time)))  
  
 # '\r'  
 # 回车，回到当前行的行首，而不会换到下一行，如果接着输出的话，本行以前的内容会被逐一覆盖；  
 # '\n'  
 # 换行，换到当前位置的下一行，而不会回到行首；  
  
 # message[:-2]去掉最后两行空白的\r\n\r\n  
 message = message[:-2] + headers + '\r\n\r\n' # 把If-Modified-Since字段加入到请求报文中  
  
 # 向服务器发送报文  
 server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 server\_socket.connect((target\_host\_without\_port, target\_port))  
 server\_socket.sendall(message.encode())  
 data = server\_socket.recv(PARAMETERS['MAX\_LENGTH']).decode('utf-8', 'ignore')  
 # print(data)  
 server\_socket.close()  
  
 # data是HTTP/1.1 304 ok  
 print('线程:' + str(threadId) + '-->' + '响应码为' + data[9:12])  
 if data[9:12] == '304': # 响应码为304，表示网页未变化，从cache中读取网页  
 print('--------------------------------------------')  
 print('线程:' + str(threadId) + '-->' + '响应码304-当前服务器资源未被修改，直接读取代理服务器cache返回')  
 print('304')  
 print('--------------------------------------------')  
 print('线程:' + str(threadId) + '---------代理服务器，304读取缓存，工作结束------------')  
 # 二进制文件就用二进制方法读取'rb'  
 with open(path, "rb") as f:  
 # 直接把代理服务器缓存里的目标服务器资源发送给客户端  
 dataSocket.sendall(f.read())  
 else: # 网页变化，标记为已修改  
 modified = True  
 # 使得进行下面的代码段运行  
  
 if not os.path.exists(path) or modified: # 如果没有该网页的缓存，或者网页已被修改（拿到modified=true)  
 # 向服务器发送数据，才能接收到服务器发回来的数据  
 print('线程:' + str(threadId) + '-->' + '给真正要访问的地址:' + target\_host\_without\_port + '发送访问请求')  
 server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 server\_socket.connect((target\_host\_without\_port, target\_port))  
 server\_socket.sendall(message.encode())  
  
 print('线程:' + str(  
 threadId) + '-->' + '代理服务器准备接受来自' + target\_host\_without\_port + ':' + str(  
 target\_port) + '的数据，随后转发给客户端，并且更新代理服务器cache')  
 f = open(path, 'wb') # 重写缓存  
 while True:  
 buff = server\_socket.recv(PARAMETERS['MAX\_LENGTH'])  
 if not buff:  
 # print(buff)  
 # 如果返回空bytes，表示对方关闭了连接  
 print('线程:' + str(threadId) + '-->' + '对方服务器发送完毕数据，对方服务器关闭连接')  
 f.close()  
 server\_socket.close()  
 break  
 f.write(buff) # 将接收到的数据写入缓存  
 dataSocket.sendall(buff) # 通过代理服务器的dataSocket将接收到的数据转发给客户端  
 print('线程:' + str(threadId) + '-->' + '代理服务器转发给客户端数据，转发已完毕')  
 dataSocket.close()  
 print('线程:' + str(threadId) + '---------代理服务器，工作结束------------')  
  
  
def main():  
 print("初始化socket")  
  
 # 实例化一个socket对象  
 # 参数 AF\_INET 表示该socket网络层使用IP协议  
 # 参数 SOCK\_STREAM 表示该socket传输层使用TCP协议  
 listenSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 # 绑定IP地址和端口号  
 listenSocket.bind((PARAMETERS['HOST'], PARAMETERS['PORT']))  
 # 使socket处于监听状态，等待客户端的连接请求  
 # 参数 MAX\_LISTEN 表示 最多可以有MAX\_LISTEN处于排队等待连接  
 # socket的排队个数  
 listenSocket.listen(PARAMETERS['MAX\_LISTEN'])  
  
 # 创建cache目录  
 if not os.path.exists(cache\_dir):  
 os.mkdir(cache\_dir)  
  
 print('初始化成功')  
 # 线程id  
 threadId = 1  
 while True:  
 # 在循环中监听9999端口，接收到客户端请求则创建一个新线程处理  
 print('等待连接')  
 # datasocket数据socket  
 # address--客户端IP地址和端口号  
 dataSocket, address = listenSocket.accept()  
 print('接受来自主机' + str(address) + '的连接')  
 threading.Thread(target=socket\_tcp, args=(dataSocket, address, threadId)).start()  
 threadId = threadId + 1  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()