

**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 | 瞿久尧 | | 院系 | 计算学部 | | |
| 班级 | 2037101 | | 学号 | 120L022314 | | |
| 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（钓鱼）。 |
| 实验过程： |
| 1. 在Edge浏览器中选择设置代理，并创建如下代理服务器：   @$R6DF43{4P[AGP2)SV{{_F   1. 首先是HTTP 代理服务器。先是代理服务器初始化，   NEJY($I1_`}S8]0}07$4JZL  接着是代理服务器链接函数，（返回文件的部分在cache部分展示）  J]%J(Q{~XXK%U]4N`IM32)4  最后是在主函数中使用多线程  MIAYB5WC)VHPH`T_V]SQ5NN   1. 下面实现cache功能，在初始化时已经设置cache路径   1YAWES~VY(H%W6ZD$M]H2HA  接下来是cache功能的实现，在客户端向服务器发起请求时，先查看本地是否有缓存，如果有的话，查看缓存的时间，并向服务器核实是否过期，如果未过期，则直接返回文件，如果已过期或不存在该缓存文件，则再向服务器申请，代码如下：  170`$DYOIPBJ$~]69M79GYQ  Y50JYHR9~$(LKIF@)H[4SDI   1. 最后是网站过滤、用户过滤和网站引导三个功能。首先在配置文件中设置要过滤的网站、过滤用户的IP地址以及想要引导的网站   65@4DAOLLSHB9KOV[9D4WKY  如果需要过滤某个网站，则返回404界面  HF]1KAJ%37U499RBQTUX`VX  如果需要过滤某个IP，则返回403界面  P[2]8H$48%1T4WD64E4%GN8  如果访问的网址为被钓鱼网址，则返回钓鱼内容网址  0XPT$@D]L7NC9WLID0JPB0B |
| 实验结果： |
| 浏览器使用代理  WNKX0R8O@HR109)SL8@_[C6  运行代码，访问www.gov.cn  KX)RQQ2[R3}8}HYL2F%7O4V  cache文件夹  W6M5[ZF87S3G[P_2_O89ZK8  访问被过滤的清华大学网址  {3CD0]0A`)[Z[`K`GD9G%]M  下面是用户屏蔽功能，本机无法正常访问  I907C~~~HHOC3QM[1[$0V~S  最后是钓鱼功能，把www.hit.edu.cn引导到http://www.chinawriter.com.cn/  [Q27[BPH(OLEG$`Z~J0NKE9 |
| 问题讨论： |
| 1. 对如何读取URL没有头绪，通过查找资料解决了问题 2. 用户过滤和网站过滤不知道如何进行，最后选择当满足条件时返回本地的HTML文件 3. 超时事件的设置，通过查找资料，采取实验指导书中的方式解决 |
| 心得体会： |
| 结合实验过程和结果给出实验的体会和收获。  初步了解了 Socket 网络编程的过程与技术，帮助我课上的理解 HTTP 协议以及代理服务器的原理，同时知道了IP过滤、网站屏蔽以及钓鱼网站的原理，帮助我提升了网络安全意识。 |

附录：程序源代码

120L022314-瞿久尧-实验.py

import json  
import os  
import socket  
import threading  
import time  
import urllib.parse as urlparse  
  
import requests  
  
  
class ProxyServer(object):  
 def \_\_init\_\_(self):  
 self.severPort = 8080 # 代理服务器端口  
 self.main\_sock = socket.socket(  
 socket.AF\_INET, socket.SOCK\_STREAM) # 创建TCP主套接字  
 self.main\_sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)  
 self.main\_sock.bind(('', self.severPort)) # 绑定端口  
 self.MAX\_LISTEN = 20  
 self.main\_sock.listen(self.MAX\_LISTEN) # 最大连接数  
 self.HTTP\_BUFFER\_SIZE = 4096 # http缓存大小  
 self.default\_cache\_dir = r'D:\Rainbow\Study\大三秋\计算机网络\实验\lab1\120L022314-瞿久尧-实验1\cache\\'  
 self.making\_cache\_dir()  
  
 def making\_cache\_dir(self): # 缓存路径  
 if not os.path.exists(self.default\_cache\_dir):  
 os.mkdir(self.default\_cache\_dir)  
  
 def filter\_web(self, url): # 过滤网站  
 with open("config.json", "r") as f:  
 filter\_json = json.load(f)  
 host\_denied = filter\_json['host']  
 for url\_denied in host\_denied:  
 if url in url\_denied:  
 return True  
 return False  
  
 def filter\_userip(self, ip): # 过滤ip  
 with open("config.json", "r") as f:  
 filter\_json = json.load(f)  
 user\_denied = filter\_json['ip']  
 if ip in user\_denied:  
 return True  
 return False  
  
 def filter\_fishing(self, url): # 钓鱼  
 with open("config.json", "r") as f:  
 filter\_json = json.load(f)  
 fishing = filter\_json['fishing']  
 for fishes in fishing:  
 if url in fishes:  
 return True  
 return False  
  
 def proxy\_connect(self, sock\_to\_web, address):  
 message = sock\_to\_web.recv(  
 self.HTTP\_BUFFER\_SIZE).decode('utf-8', 'ignore')  
  
 print(f"Msg:{message}")  
 msgs = message.split('\r\n')  
 print(f"Format\_msgs:{msgs}")  
 request\_line = msgs[0].strip().split()  
 print(f"Request line:{request\_line}")  
 print(f"len(request line):{len(request\_line)}")  
 # print(f"request\_line[1]:{request\_line[1]}")  
 if len(request\_line) < 1:  
 print("Request Line not contains url!")  
 print(f"Full Request Message:{message}")  
 sock\_to\_web.close() # 关闭连接sock  
 return  
  
 else:  
 # scheme://netloc/path;parameters?query#fragment url的一般形式 urlparse还可以包括 username password hostname port  
 url = urlparse.urlparse(  
 request\_line[1][:-1] if request\_line[1][-1] == '/' else request\_line[1])  
 print(f"url.scheme:{url.scheme},type:{type(url.scheme)}")  
 print(f"url.hostname:{url.hostname},type:{type(url.hostname)}")  
 print(f"url.port:{url.port},type:{type(url.port)}")  
 print(f"url.scheme:{url.path},type:{type(url.path)}")  
 print(f"url.netloc:{url.netloc},type:{type(url.netloc)}")  
  
 if self.filter\_web(url.hostname): # 如果需要过滤某个网站  
 with open("404.html") as f:  
 sock\_to\_web.sendall(f.read().encode())  
 sock\_to\_web.close()  
 return  
  
   
 #if self.filter\_userip(address[0]): # 如果需要过滤某个IP  
 # with open("403.html") as f:  
 # sock\_to\_web.sendall(f.read().encode())  
 # sock\_to\_web.close()  
 # return  
   
 if self.filter\_fishing(url.hostname): # 将需要钓鱼的网站重定向至中国作家网  
 sock\_to\_web.sendall(requests.get("http://www.chinawriter.com.cn/").content) #也可以尝试将钓鱼网站换成其他的，  
 sock\_to\_web.close()  
 return  
  
 cache\_path = self.default\_cache\_dir + \  
 (str(url.hostname) + str(url.path)).replace('/', '\_')  
 flag\_modified = False # 默认缓存没有更改  
 flag\_exists = os.path.exists(cache\_path) # 检测缓存目录是否存在  
 sock\_to\_Client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
 if flag\_exists:  
 cache\_time = os.stat(cache\_path).st\_mtime # 获取缓存的时间  
 msgs = {  
 'If-Modified-Since': time.strftime('%a, %d %b %Y %H:%M:%S GMT', time.gmtime(cache\_time))}  
 response = requests.get(url.geturl(), headers=msgs)  
 if response.status\_code == 304: # 如果返回304 则无需进行重新访问  
 print("Read From Cache" + cache\_path)  
 with open(cache\_path, "rb") as f:  
 sock\_to\_web.sendall(f.read())  
 else:  
 flag\_modified = True # 否则证明缓存已经过时  
  
 if not flag\_exists or flag\_modified: # 如果没有缓存或者缓存文件已经发生变化  
 print("Attempt to connect", url.geturl())  
 sock\_to\_Client.connect(  
 (url.hostname, url.port if url.port else 80))  
 sock\_to\_Client.sendall(message.encode())  
 temp\_file = open(cache\_path, 'w') # 记录缓存  
 while True:  
 buff = sock\_to\_Client.recv(self.HTTP\_BUFFER\_SIZE)  
 if not buff:  
 temp\_file.close()  
 sock\_to\_Client.close()  
 break  
 temp\_file.write(buff.decode('gbk', 'ignore'))  
 sock\_to\_web.sendall(buff)  
 sock\_to\_web.close()  
  
  
# 主函数 使用threading实现多线程  
def main():  
 proxy = ProxyServer()  
 while True:  
 new\_sock, address = proxy.main\_sock.accept()  
 print(address)  
 threading.Thread(target=proxy.proxy\_connect,  
 args=(new\_sock, address)).start()  
  
  
# 程序入口  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

config.json

{  
 "host": [  
 "www.tsinghua.edu.cn"  
 ],  
 "ip": [  
 "127.0.0.1"  
  
 ],  
 "fishing": [  
 "www.hit.edu.cn"  
 ]  
}

301.html

<html>  
<head><title>301 Moved Permanently</title></head>  
<body bgcolor="white">  
<center><h1>301 Moved Permanently</h1></center>  
</body>  
</html>

403.html

<html>  
<head><title>403 Forbidden </title></head>  
<body bgcolor="white">  
<center><h1>403 Forbidden </h1></center>  
</body>  
</html>

404.html

<html>  
<head><title>404 Not Found</title></head>  
<body bgcolor="white">  
<center><h1>404 Not Found</h1></center>  
</body>  
</html>