哈尔滨工业大学

**<<计算机网络>>**

**实验报告**

**(2017年度春季学期)**

|  |  |
| --- | --- |
| **姓名：** | **楼雨京** |
| **学号：** | **1140310415** |
| **学院：** | **计算机科学与技术学院** |
| **教师：** | **李全龙** |

## 实验一 HTTP代理服务器的设计与实现

1. **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。

1. **实验环境**

接入Internet的实验主机；

操作系统MacOS；

开发语言：Python。

1. **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）

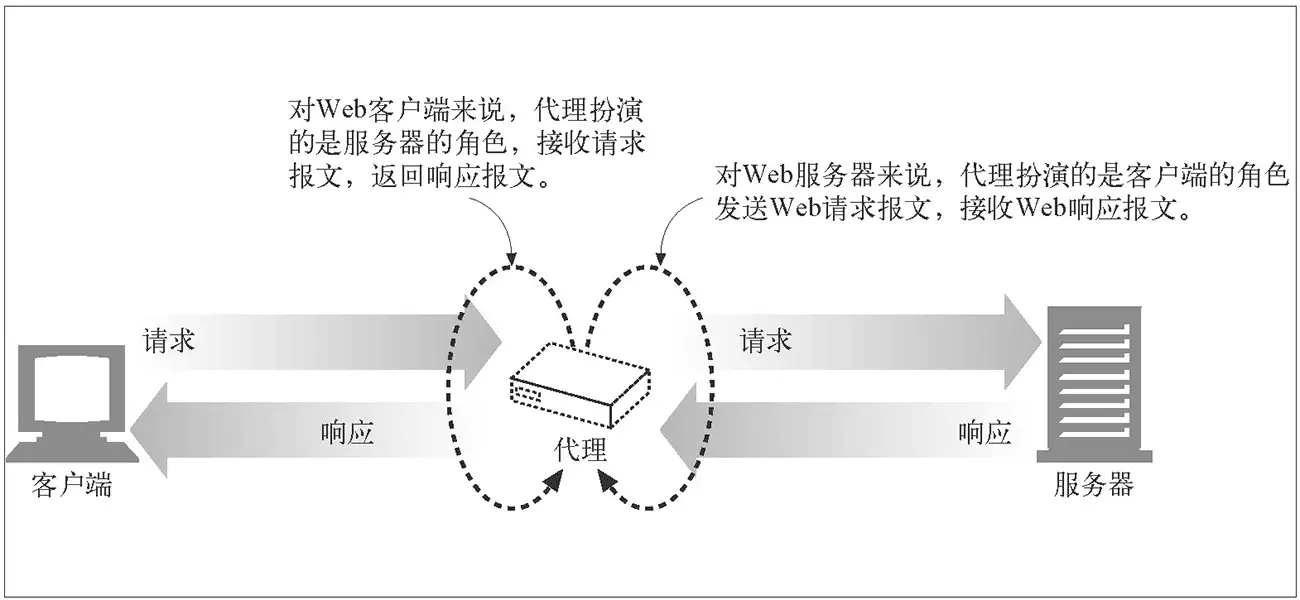
(3) 扩展 HTTP 代理服务器，支持如下功能：（选作内容，加分项目， 可以当堂完成或课下完成）

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；

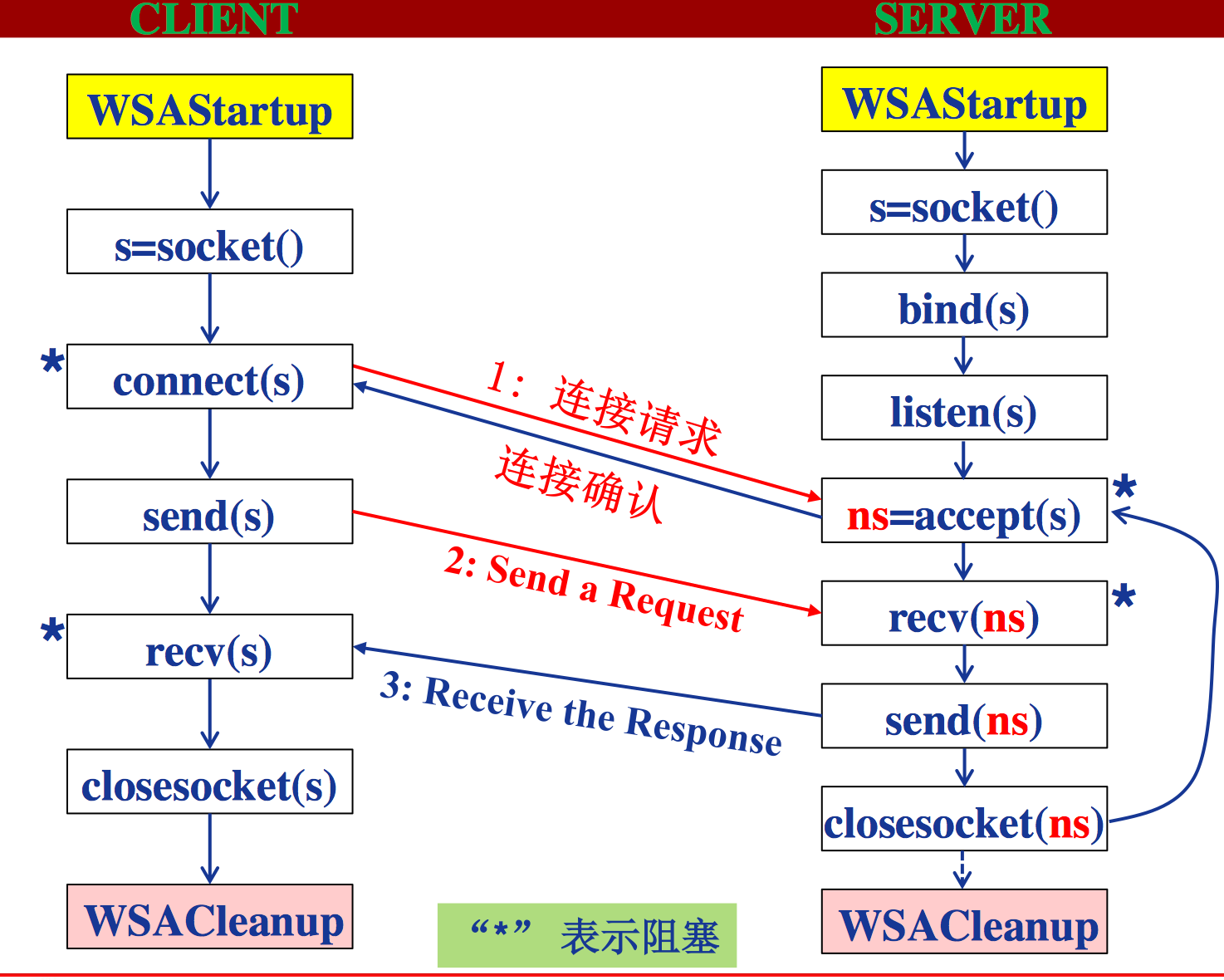
c) 网站引导：将用户对某个网站的访问引导至一个模拟网站（钓鱼）

1. **http代理服务器的基本原理**

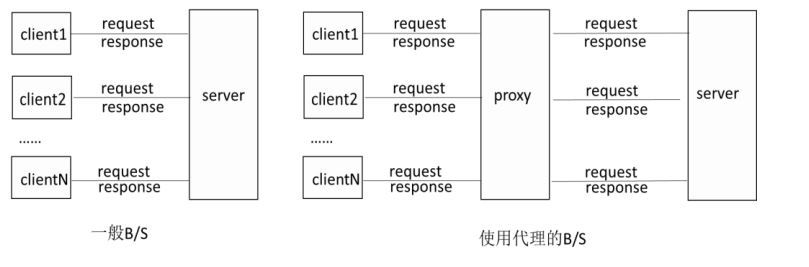


代理服务器从客户获得HTTP请求报文，解析报文中目的主机的地址，向目的主机建立连接，再从服务器端获取数据，将内容转发给客户端。

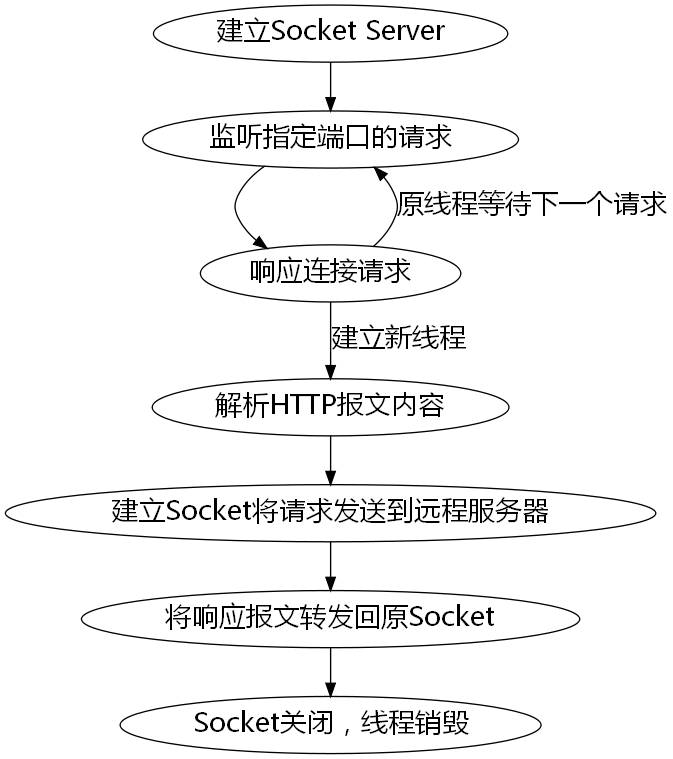
在没有加入代理服务器的时候，客户端与服务器分别建立Socket进行数据的传输，如下图。



代理服务器实现的就是接受来自客户端的connect连接请求，进行解析以后，再向服务器发送相应的请求，然后服务器根据请求信息进行响应，代理服务器接受来自服务器的数据，最后再转发给客户端。这样完成一个代理服务器的功能。

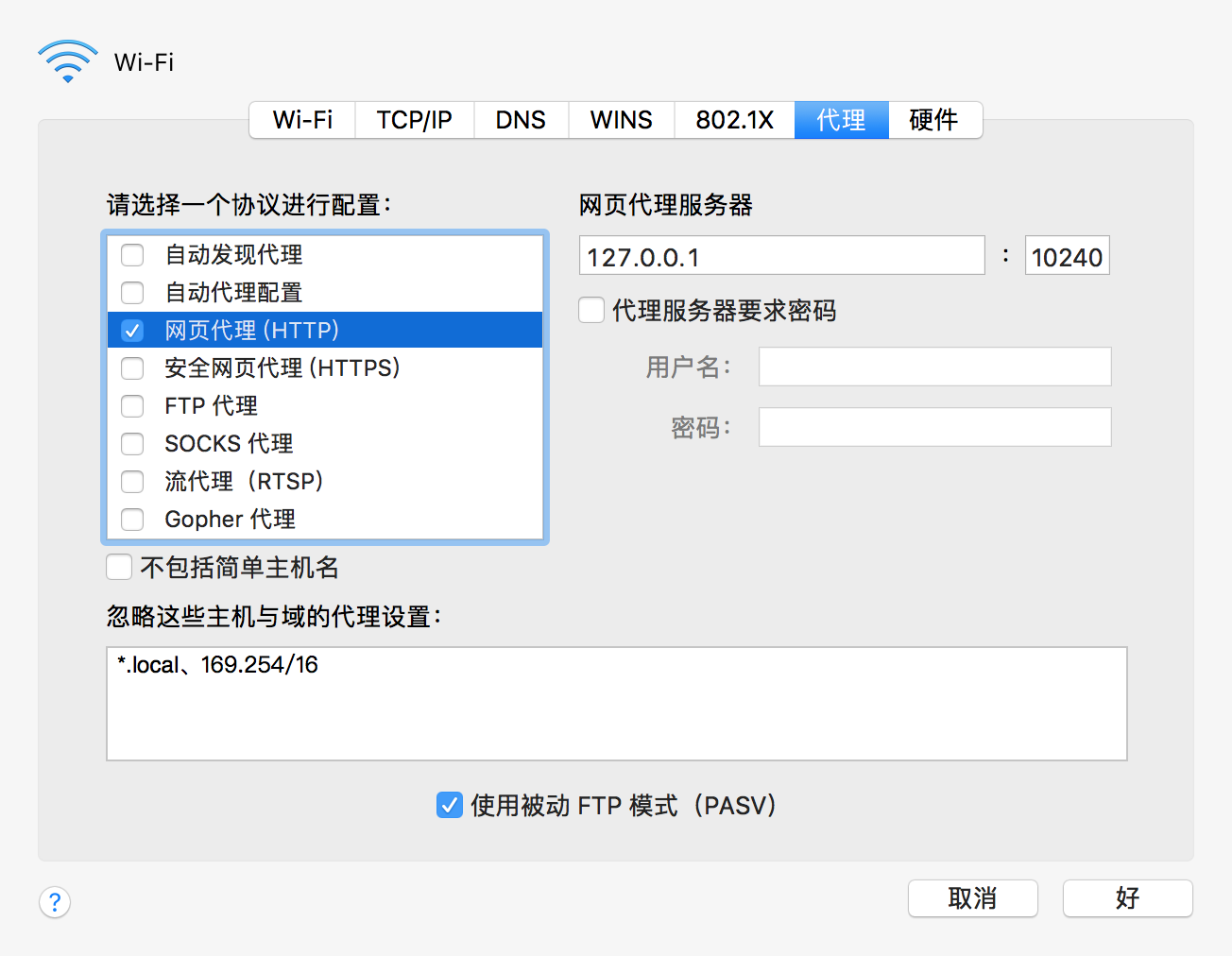


1. **流程图**

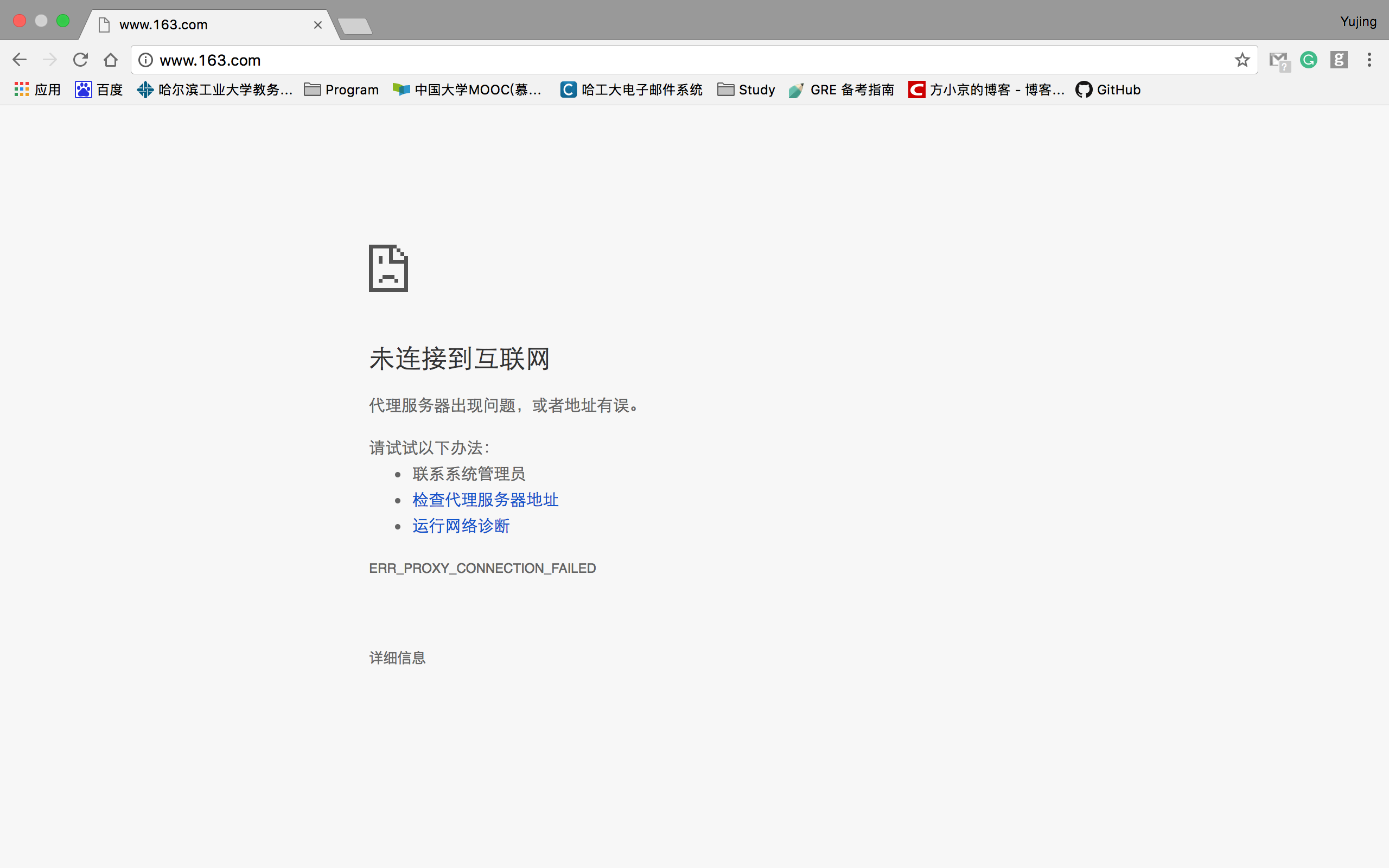


1. **实验过程**

设置代理服务器地址与端口，地址为本地地址127.0.0.1，端口号为10240.



在没有打开服务器时，访问网站www.163.com，可以发现无法连接到网站。



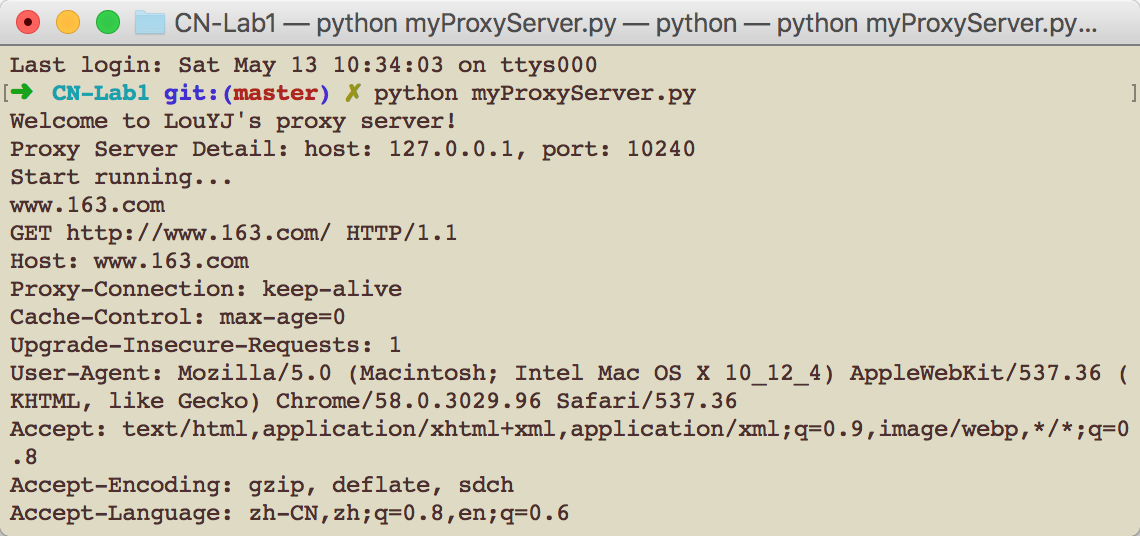
接着运行代理服务器。



此时，再次访问网站。



成功连接到对应网站，接受到服务器所返回的数据。



上图是代理服务器运行时发出的HTTP请求消息，然后之后根据网页的请求，再依次获取网页内部的其他内容。

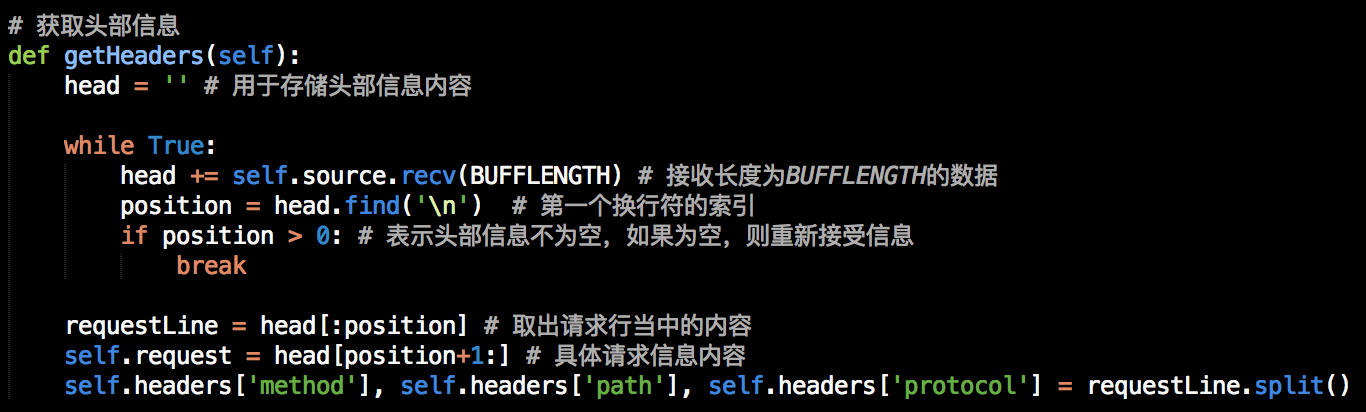


1. **主要功能的实现**

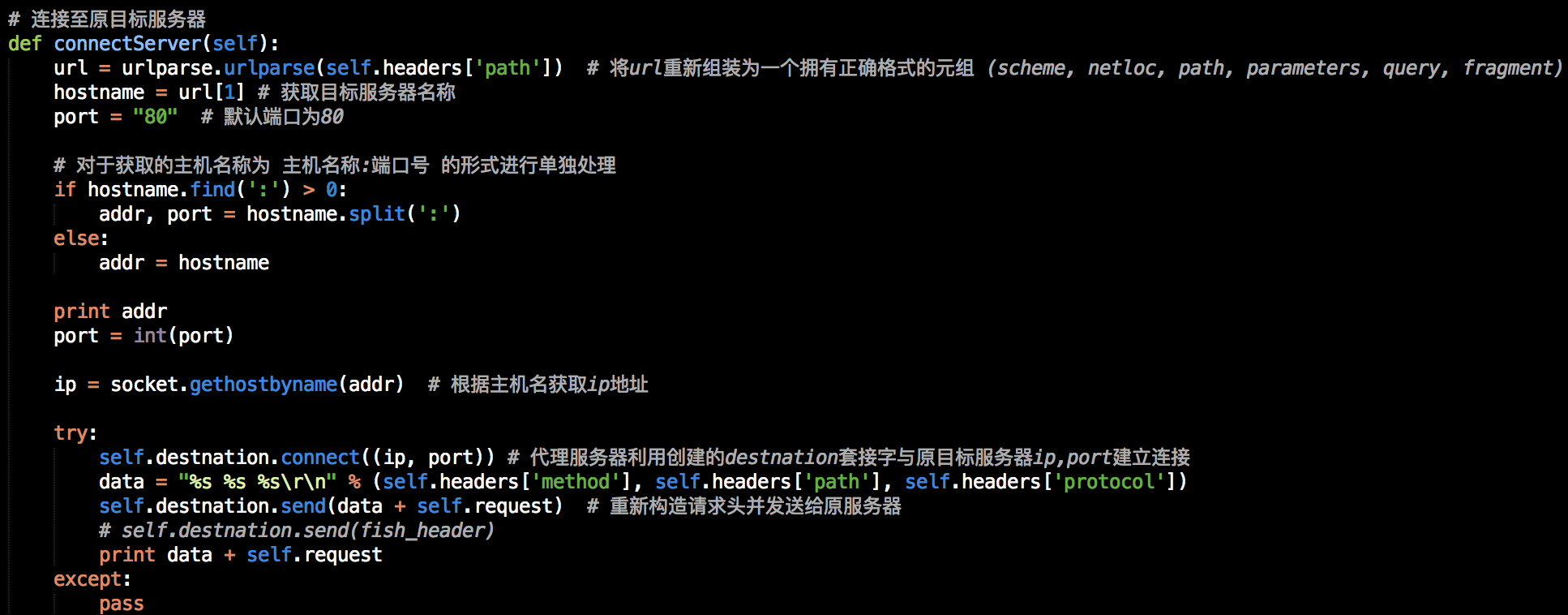
代理服务器建立套接字，监听并接受来自客户端的请求。



代理服务器解析请求信息的头部信息，获取目标地址。



根据头部信息，解析路径获得目标IP地址以及端口号，与原目标服务器建立连接并发送请求。



最后接受来自服务器相应的数据，并将其转发给客户端。



1. **实验代码**

# -\*- coding:utf-8 -\*-

import socket

import thread

import urlparse

import select

BUFFLENGTH = 8192

class ProxyToServer(object):

def \_\_init\_\_(self, conn, address):

# 新建目的套接字，socket.AF\_INET为协议族，socket.SOCK\_STREAM指定套接字类型为流

self.destnation = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.source = conn # 用于存储来自客户端的请求

self.request = ""

self.headers = {} # 用于存储HTTP请求消息的请求行内容：方法，路径，协议版本

self.execute()

# 获取头部信息

def getHeaders(self):

head = '' # 用于存储头部信息内容

while True:

head += self.source.recv(BUFFLENGTH) # 接收长度为BUFFLENGTH的数据

position = head.find('\n') # 第一个换行符的索引

if position > 0: # 表示头部信息不为空，如果为空，则重新接受信息

break

requestLine = head[:position] # 取出请求行当中的内容

self.request = head[position+1:] # 具体请求信息内容

self.headers['method'], self.headers['path'], self.headers['protocol'] = requestLine.split()

# 连接至原目标服务器

def connectServer(self):

url = urlparse.urlparse(self.headers['path']) # 将url重新组装为一个拥有正确格式的元组 (scheme, netloc, path, parameters, query, fragment)

hostname = url[1] # 获取目标服务器名称

port = "80" # 默认端口为80

# 对于获取的主机名称为 主机名称:端口号 的形式进行单独处理

if hostname.find(':') > 0:

addr, port = hostname.split(':')

else:

addr = hostname

print addr

port = int(port)

ip = socket.gethostbyname(addr) # 根据主机名获取ip地址

try:

self.destnation.connect((ip, port)) # 代理服务器利用创建的destnation套接字与原目标服务器ip,port建立连接

data = "%s %s %s\r\n" % (self.headers['method'], self.headers['path'], self.headers['protocol'])

self.destnation.send(data + self.request) # 重新构造请求头并发送给原服务器

# self.destnation.send(fish\_header)

print data + self.request

except:

pass

# 接受来自服务器的数据

def acceptData(self):

readsocket = [self.destnation]

while True:

data = ''

(rlist, wlist, elist) = select.select(readsocket, [], [], 3)

if rlist:

data = rlist[0].recv(BUFFLENGTH) # 代理服务器从原服务器接受数据

if len(data) > 0:

self.source.send(data) # 向客户端发送数据

else:

break

# 执行

def execute(self):

self.getHeaders()

self.connectServer()

self.acceptData()

class ClientToServer(object):

def \_\_init\_\_(self, host, port, handler=ProxyToServer):

self.host = host

self.port = port

self.server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) # 代理服务器主套接字

self.server.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1) # 打开地址复用功能

self.server.bind((host, port)) # 绑定代理服务器套接字的本地IP地址和端口号

self.server.listen(5) # 设置代理服务器套接字为监听状态且队列大小为5

self.handler = handler

def execute(self):

while True:

try:

conn, addr = self.server.accept() # 代理服务器调用accept函数从处于监听状态的流套接字sd的客户连接请求队列中取出排在最前的一个客户请求

thread.start\_new\_thread(self.handler, (conn, addr)) # 创建代理线程，即每针对一个接受的连接请求创建子线程实现一对一代理

except:

pass

if \_\_name\_\_ == '\_\_main\_\_':

host = '127.0.0.1'

port = 10240

myProxyServer = ClientToServer(host, port)

print "Welcome to LouYJ's proxy server!"

print "Proxy Server Detail: host: %s, port: %d" % (host, port)

print 'Start running...'

myProxyServer.execute()