第二次作业

章节后习题 P3、P4、P7、P12、P28

P3考虑一个要获取给定URL的Web文档的HTTP客户。该HTTP服务器的IP地址开始时并不知道。在这种情况下，除了HTTP外，还需要什么运输层和应用层协议？

传输层协议TCP 提供可靠连接，应用层协议DNS 负责将域名解析为 TCP/IP可以使用的 IP 地址。

P4，考虑当浏览器发送一个HTTP GET报文时，通过 Wireshark 俘获到下列ASCI字符串（即这是一个HTTP GET报文的实际内容）。字符<cr><lf>是回车和换行符（即下面文本中的斜体字符串<cr>表示了单个回车符，该回车符包含在HTTP首部中的相应位置）。回答下列问题，指出你在下面HTTP GET报文中找到答案的地方。

GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0(Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex

t/xml, application/xml, application/xhtml+xml, text

/html;q=0.9, text/plain;q=0.8,image/png,\*/\*;g=0.5

<cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept-

Encoding: zip,deflate<cr><lf>Accept-Charset: IS0

-8859-1,utf-8:q=0.7,\*;q=0.7<cr><lf>Keep-Alive: 300<cr>

<lf>Connection:keep-alive<cr><lf><cr><lf>

1. 由浏览器请求的文档的URL是什么？

/cs453/index.html

1. 该浏览器运行的是HTTP的何种版本？

HTTP/1.1

1. 该浏览器请求的是一条非持续连接还是一条持续连接？

keep-alive 持续连接

d该浏览器所运行的主机的IP地址是什么？

无法获取 IP

e.发起该报文的浏览器的类型是什么？在一个HTTP请求报文中，为什么需要浏览器类型？

User-Agent： Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax)

浏览器类型信息有助于服务器根据不同浏览器的功能和要求理解并适当处理请求。

P7.假定你在浏览器中点击一条超链接获得Web页面。相关联的URL的IP地址没有缓存在本地主机上，因此必须使用DNS lookup 以获得该IP地址。如果主机从DNS得到IP地址之前已经访问了n个DNS服务器：相继产生的RTT依次为RTT1、…、RTTn。进一步假定与链路相关的Web页面只包含一个对象，即由少量的HTML文本组成。令RTT0表示本地主机和包含对象的服务器之间的RTT值。假定该对象传输时间为零，则从该客户点击该超链接到它接收到该对象需要多长时间？

DNS 查询总时间 = RTT1 + RTT2 + ... + RTTn

假设传输一个小 HTML 文本对象不需要传输时间；数据传输时间、连接建立时间忽略不计。则总时间 = DNS 查询总时间 + 连接建立时间 + HTTP 请求/响应时间 + 数据传输时间= RTT1 + RTT2 + ... + RTTn

P12 Py写一个简单的TCP程序，使服务器接收来自客户的行并将其打印在服务器的标准输出上。（可以通过修改本书中的TCPServer.py程序实现上述任务。）编译并执行你的程序。在另一台有测览器的机器上，设置浏览器的代理服务器为你正在运行服务器程序的机器，同时适当地配置端口号。这时你的浏览器向服务器发送CET请求报文，你的服务器应当在其标准输出上显示该报文。使用这个平台来确定你的浏览器是否对本地缓存的对象产生了条件GET报文。

Server.py 代码：

import socket

HOST = "192.168.1.106" # 服务器的主机名或 IP 地址

PORT = 8888 # 服务器使用的端口号

# 创建一个 TCP/IP socket 对象

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

# 绑定服务器地址和端口号

s.bind((HOST, PORT))

# 开始监听客户端连接

s.listen()

print(f"Server listening on {HOST}:{PORT}...")

# 等待客户端连接

conn, addr = s.accept()

with conn:

print(f"Connected by {addr}")

while True:

# 接收客户端发送的数据

data = conn.recv(1024)

if not data:

break

# 打印客户端发送的数据

print(data.decode("utf-8"))

结果：

$ C:/Users/lxl/scoop/apps/python/current/python.exe z:/test/server.py

Server listening on 192.168.1.106:8888...

Connected by ('192.168.1.110', 59044)

GET http://www.baidu.com/ HTTP/1.1

Host: www.baidu.com

Proxy-Connection: keep-alive

Cache-Control: max-age=0

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 Edg/118.0.2088.46

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7

Accept-Encoding: gzip, deflate

Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6

Cookie: BAIDU\_WISE\_UID=wapp\_1638112931475\_409; BIDUPSID=1933BD9A7B5B60B80B428BA381FAC5B5; PSTM=1654579898; \_\_bid\_n=1841240e9b3b275d284207; FEID=v10-3363e793d746570e41de74752f450eb1735aae26; \_\_xaf\_fpstarttimer\_\_=1672745140791; \_\_xaf\_thstime\_\_=1672745140796; \_\_xaf\_fptokentimer\_\_=1672745140839; BDUSS=…; BAIDUID=0BE56

00E2B5313DD02421AAE3CCA51F6:FG=1; FPTOKEN=…

P28 在一台主机上安装并编译 TCPClient 和UDPClient Python 程序，在另一台主机上安装并编译TCPServer 和UDPServer程序。

a.假设你在运行TCPServer 之前运行TCPClient，将发生什么现象？为什么？

b.假设你在运行UDPServer之前运行UDPClient，将发生什么现象？为什么？

c.如果你对客户端和服务器端使用了不同的端口，将发生什么现象？

1. TCPserver:

from socket import \*

serverPort = 12000

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

serverSocket.bind(("", serverPort))

serverSocket.listen(1)

print("The server is ready to receive")

while True:

connectionSocket, addr = serverSocket.accept()

sentence = connectionSocket.recv(1024).decode()

capitalizedSentence = sentence.upper()

connectionSocket.send(capitalizedSentence.encode())

connectionSocket.close()

结果：在运行TCPClient一段时间后运行TCPServer，两者能建立连接。而超出该一段时间后将发生错误：

C:\Users\DELL\AppData\Local\Programs\Python\Python310\python.exe D:/互联网/tcpclient.py

Traceback (most recent call last):

File "D:\互联网\tcpclient.py", line 6, in <module>

clientSocket.connect((serverName,serverPort))

ConnectionRefusedError: [WinError 10061] 由于目标计算机积极拒绝，无法连接。

Process finished with exit code 1

所以在该一段时间内，TCP 会尝试重传，当重传超出最大次数后即退出。

1. UDPclient:

import socket

import time

# 服务器地址和端口

server\_host = "192.168.1.110"

server\_port = 8888

# 创建UDP套接字

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# 发送Ping请求和记录发送时间

message = b"Ping"

start\_time = time.time()

client\_socket.sendto(message, (server\_host, server\_port))

# 设置超时时间为1秒

client\_socket.settimeout(1)

try:

# 等待接收Pong响应

data, server\_address = client\_socket.recvfrom(1024)

end\_time = time.time()

# 计算往返时间

rtt = end\_time - start\_time

print(f"Received: {data.decode()}")

print(f"Round-Trip Time: {rtt:.4f} seconds")

except socket.timeout:

print("Request Timed Out")

# 关闭套接字

client\_socket.close()

在运行UDPServer之前运行UDPClient，将直接报错：

$ C:/Users/lxl/scoop/apps/python/current/python.exe z:/test/udpclient.py

Traceback (most recent call last):

File "z:\test\udpclient.py", line 21, in <module>

data, server\_address = client\_socket.recvfrom(1024)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^

ConnectionResetError: [WinError 10054] 远程主机强迫关闭了一个现有的连接。

1. 如果使用不同端口，会报错：远程主机关闭现有连接 或 积极拒绝。