实践作业2

作业要求：提交word格式作业

作业内容：

**1 ）编程实现web服务器**

**可选: 多线程Web代理服务器**

**WebServer.py**

'''  
\* 功能：Web浏览器访问 Web服务器，获取 index.html 文件  
\* 目的：1、当一个客户（浏览器）联系时创建一个套接字  
\* 2、从这个连接接收 HTTP 请求  
\* 3、解释该请求以确定所请求的特定文件  
\* 4、从服务器的文件系统获得请求的文件  
\* 5、创建一个由请求文件组成的 HTTP 响应的报文，报文前面有首部行  
\* 6、经TCP连接向请求的浏览器发送响应。如果请求文件不在服务器中，返回 “404 Not Found”  
'''  
from socket import \* #引用套接字  
import os #引用文件管理  
serverHost = '127.0.0.1' #Web服务器IP地址  
serverPort = 12000 #服务端的端口号  
  
'''  
\* 函数功能：解释 HTTP 请求报文  
\* 返回值： 请求的文件名  
'''  
def HTTPExplain(Message):  
 Data = Message.decode() #解码报文  
 print(Data) #打印报文  
 Lines = Data.split("\r\n") #通过指定分隔符对字符串切片  
 Url = Lines[0].split(" ") #得到 URL [GET /index.html HTTP/1.1]  
 FileName = Url[1].split('/')[1] #得到 ['index.html']  
 print(FileName) #打印请求文件名  
 return FileName  
  
'''  
\* 函数功能：构造 HTTP 响应报文  
\* 返回值： 响应报文  
'''  
def GetFile(FileName):  
 path = "WebServer\\" + FileName #文件路径  
 if FileName in os.listdir("WebServer"): #判断文件是否存在  
 with open(path,'r',encoding="utf-8") as fp: #打开文件  
 data = fp.read() #读取文件  
 HttpResponse = "HTTP/1.1 200 OK\r\n\r\n" + data #请求文件携带请求成功首部  
 else:  
 HttpResponse = "HTTP/1.1 404 \r\n\r\n<h1>404 Not Found</h1>" #请求失败报文  
 return HttpResponse.encode()  
   
#握手套接字  
serverSocket = socket(AF\_INET,SOCK\_STREAM) #创建一个握手套接字  
serverSocket.bind((serverHost,serverPort)) #绑定地址（host,port）到握手套接字  
serverSocket.listen(1) #请求连接的最大数  
print('The server is ready,Port is ',serverPort) #提示服务器准备就绪  
#创建连接套接字  
connectSocket,addr = serverSocket.accept() #被动接受TCP客户端连接，创建连接套接字  
Message = connectSocket.recv(1024) #接收TCP数据，数据以字符串形式返回  
FileName = HTTPExplain(Message) #调用函数获取请求文件  
#连接套接字  
ResponseMessage = GetFile(FileName) #调用函数构造响应报文  
connectSocket.sendall(ResponseMessage) #完全发送响应报文  
connectSocket.close() #关闭连接套接字

**Index.html**

<!DOCTYPE html>

<html lang="zh-CN">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>标题使用</title>

</head>

<body>

<h1>一级标题</h1>

<h2>二级标题</h2>

<h3>三级标题</h3>

<h4>四级标题</h4>

<h5>五级标题</h5>

<h6>六级标题</h6>

</body>

</html>

**2）编程实现UDPping**

  # UDPPingerServer.py

from socket import \*

import random

import time

# SOCK\_DGRAM 表示使用UDP协议

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

# 绑定IP和端口

serverSocket.bind(('127.0.0.1', 10086))

while True:

# 生成一个随机数，用于模拟丢包的情况

rand = random.randint(0, 10)

# 接收信息

message, address = serverSocket.recvfrom(1024)

# 获取当前时间

cur\_time = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())

print(f"{cur\_time}\t{address}\t{message.decode()}")

# 如果随机数小于4，就不响应本次数据，模拟丢包

if rand < 4:

continue

# 收到ping就响应pong

if message == b"ping":

serverSocket.sendto(b"pong", address)

# UDPPingerClient.py

from socket import \*

import time

# SOCK\_DGRAM 表示使用UDP协议

clientSocket = socket(AF\_INET, SOCK\_DGRAM)

# 设置超时时间为1秒

clientSocket.settimeout(1)

# 服务端的ip和端口

server\_addr = ('127.0.0.1', 10086)

for i in range(10):

# 发送ping命令

clientSocket.sendto(b"ping", server\_addr)

send\_time = time.time()

try:

message, \_ = clientSocket.recvfrom(1024)

recv\_time = time.time()

print(f"{i}\t{message.decode()}\tRTT: {(recv\_time - send\_time) \* 1000:.2f}ms")

except timeout as e: # 超时就打印time out

print(f"{i}\ttime out...")

**3）Wireshark实验：http**

**打开浏览器并清空浏览器缓存。访问web网站 （http或https）**

**Wireshark 中启动数据包捕获**

**回答：**

**1 您的浏览器是否运行 HTTP 版本 1.0 或 1.1，2?服务器运行的是什么版本的 HTTP?**

**2 如果是http2，请给出并分析http格式**

**3 web服务器互联网地址是什么? 您的计算机的 Internet 地址是什么?**

**4 从服务器返回到浏览器的状态代码是什么?**

**5 在服务器上最后修改的 HTML 文件是什么时候?**

**6 如果是https，请给出并分析SSL过程**

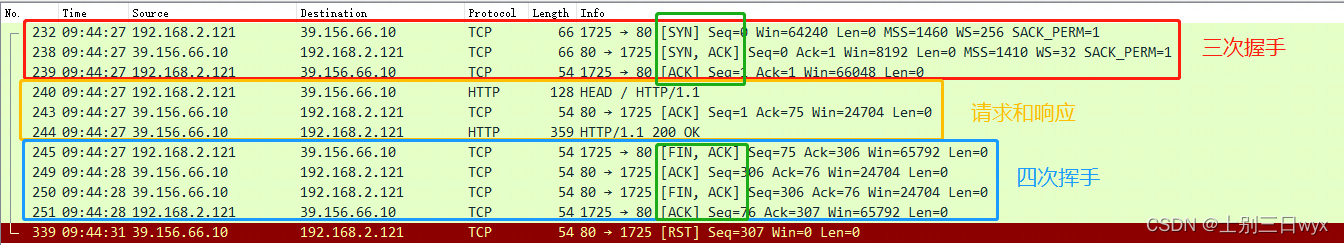
1. HTTP1.1

2.不是

3.39.156.66.10 112.65.99.117

4.200 OK

5.Tue, 12 Jan 2010 13:48:00 GMT

6.

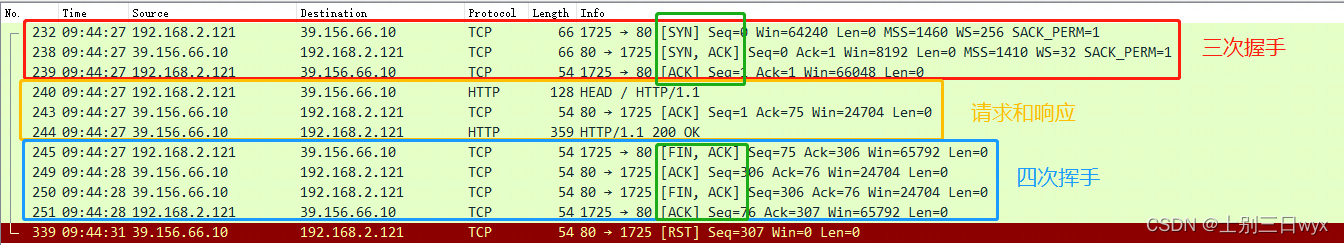
1）先看前三个包，HTTP是基于TCP的，需要先通过「三次握手」建立连接

第一个包是「第一次握手」：我（192.168.2.121）向百度（39.156.66.10）发送一个 SYN ，表示这是一个「建立连接的请求」

第二个包是「第二次握手」：百度（39.156.66.10）响应我（192.168.2.121）一个 SYN+ACK ，表示这是一个「接受连接的应答」

第三个包是「第三次握手」，我（192.168.2.121）向百度（39.156.66.10）发送一个 ACK ，表示这一个「确认请求」

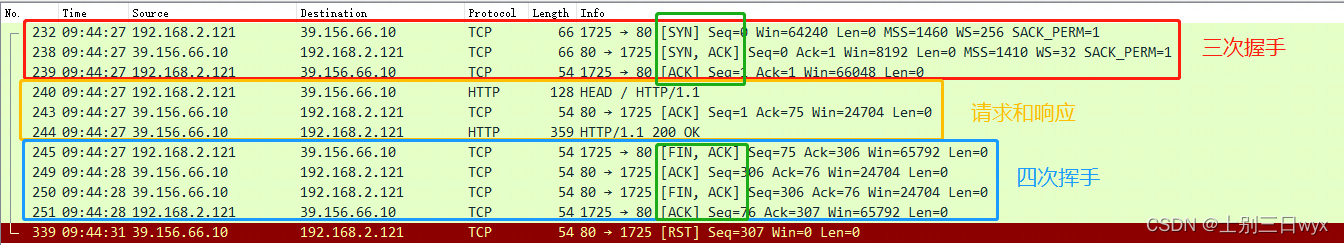
发送完确认请求后，我开启到百度的「单向连接通道」；百度收到我的确认请求后，就开启到我这边的单向连接通道；「两边通道」都开启以后，就可以通信了。



2）再看中间三个包，TCP连接建立以后，开始「HTTP的请求和响应」

第一个包是，我（192.168.2.121）向百度（39.156.66.10）发送了一个「HTTP请求」，请求类型是HEAD

第三个包是，百度（39.156.66.10）向我（192.168.2.121）发送了一个「HTTP响应」，响应状态码是 200 OK



3）再看最后四个包，请求响应结束后，「TCP四次挥手」断开连接。

第一个包是第一次挥手，我（192.168.2.121）向百度（39.156.66.10）发送一个FIN+ACK，表示这是一个释放连接的请求

第二个包是第二次挥手，百度（39.156.66.10）向我（192.168.2.121）响应一个ACK，表示这是一个确认请求；我收到后，就会释放我到百度的单向连接

第三个包是第三次挥手，百度（39.156.66.10）向我（192.168.2.121）发送一个FIN+ACK，表示这是一个释放连接的请求

第四个包是第四次挥手，我（192.168.2.121）向百度（39.156.66.10）响应一个ACK，表示这是一个确认请求；百度收到后，就会释放到我这边的单向连接

双向的连接都释放后，TCP连接就关闭了，此次通信结束。

**4）Wireshark实验：DNS**

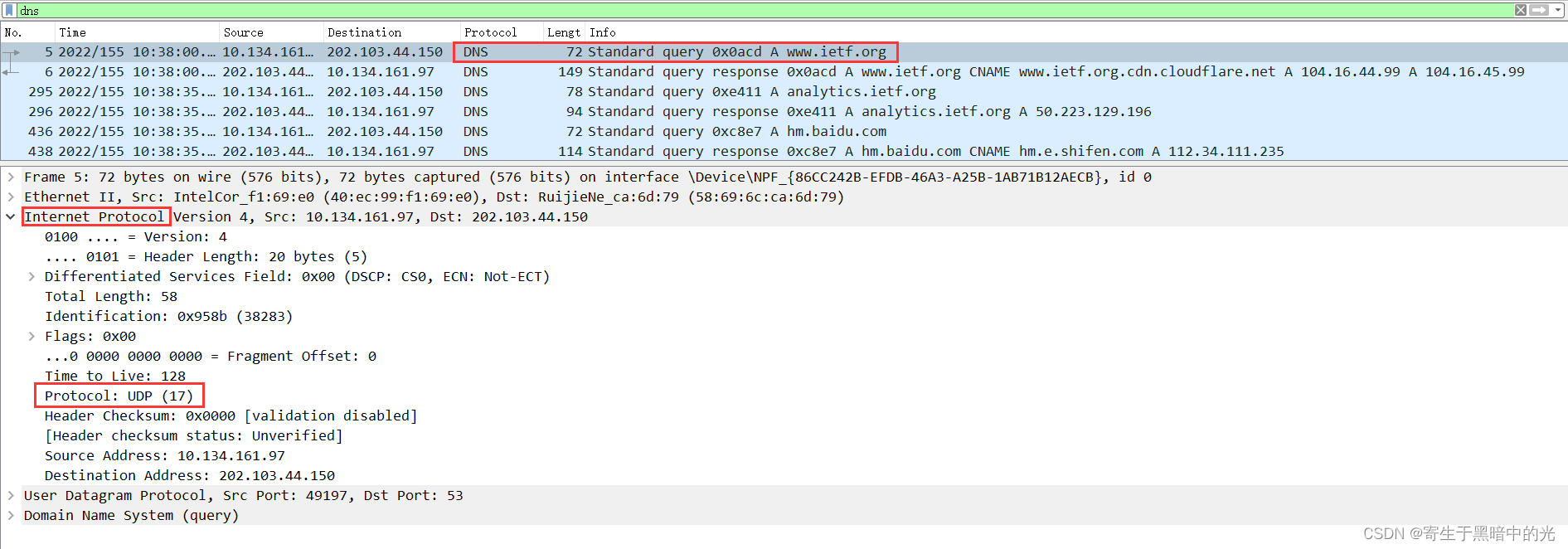
**ipconfig /flushdns清除缓存**

**打开浏览器并清空浏览器缓存。访问web网站**

**Wireshark 中启动数据包捕获**

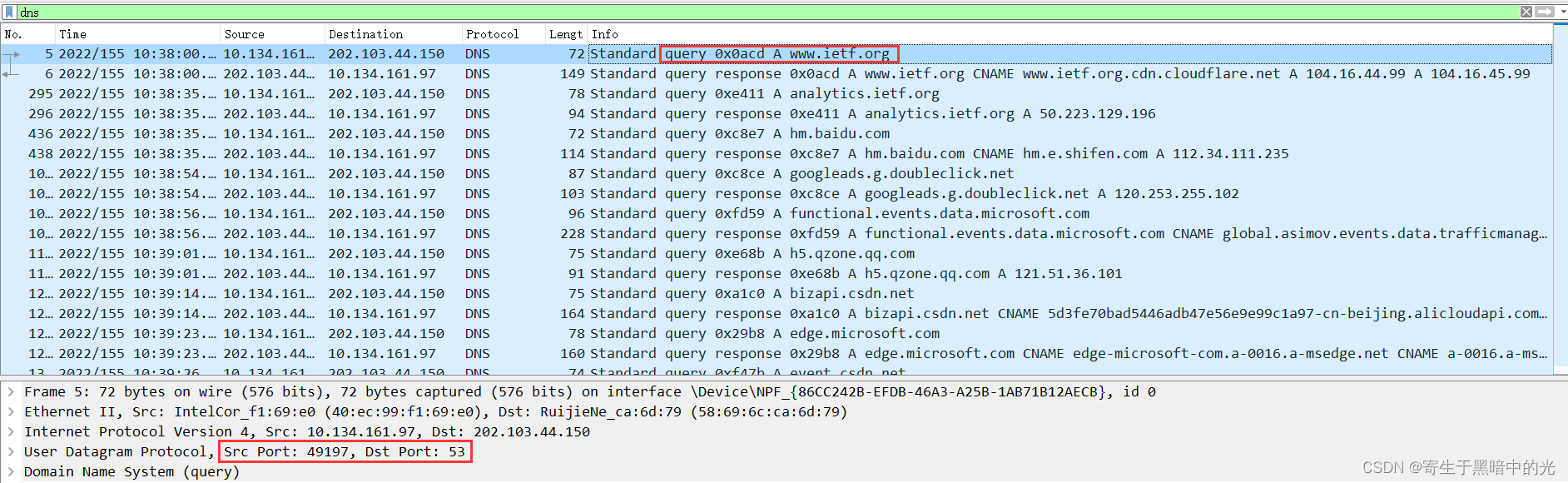
**回答：**

**1 找到 DNS 查询和响应消息。它们是否通过 UDP 或 TCP 发送?**

****

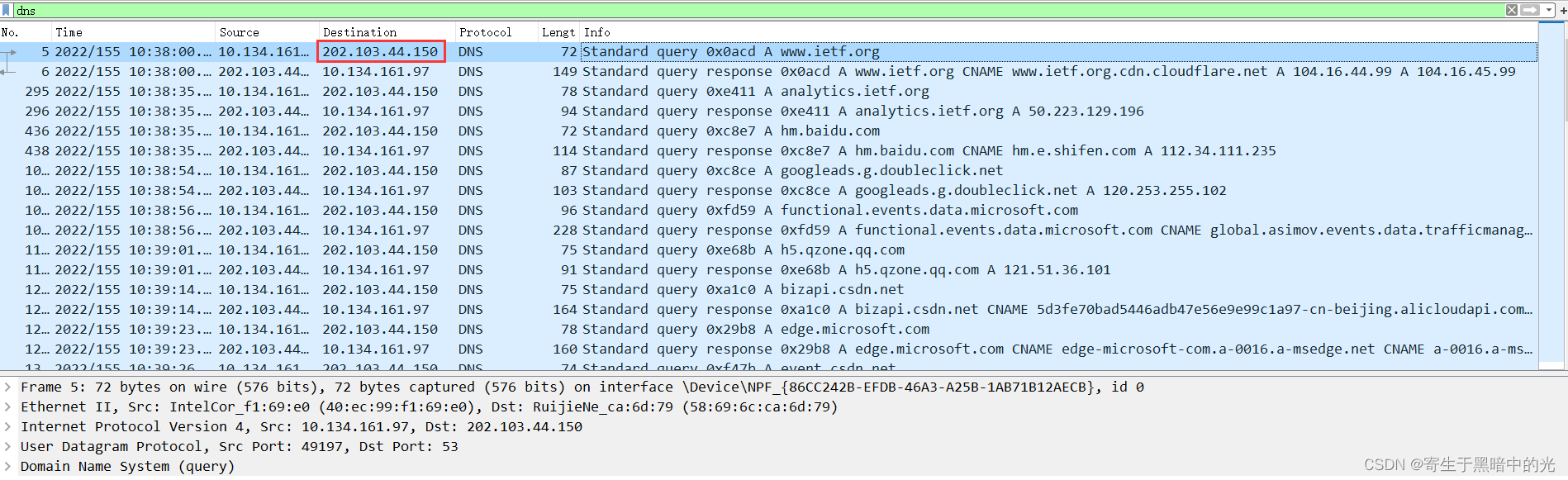
UDP

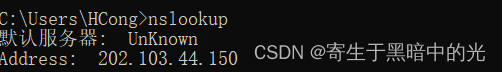
**2 DNS 查询消息的目标端口是什么? DNS 响应消息的源端口是什么?**

****

源端口：49197，目标端口：53

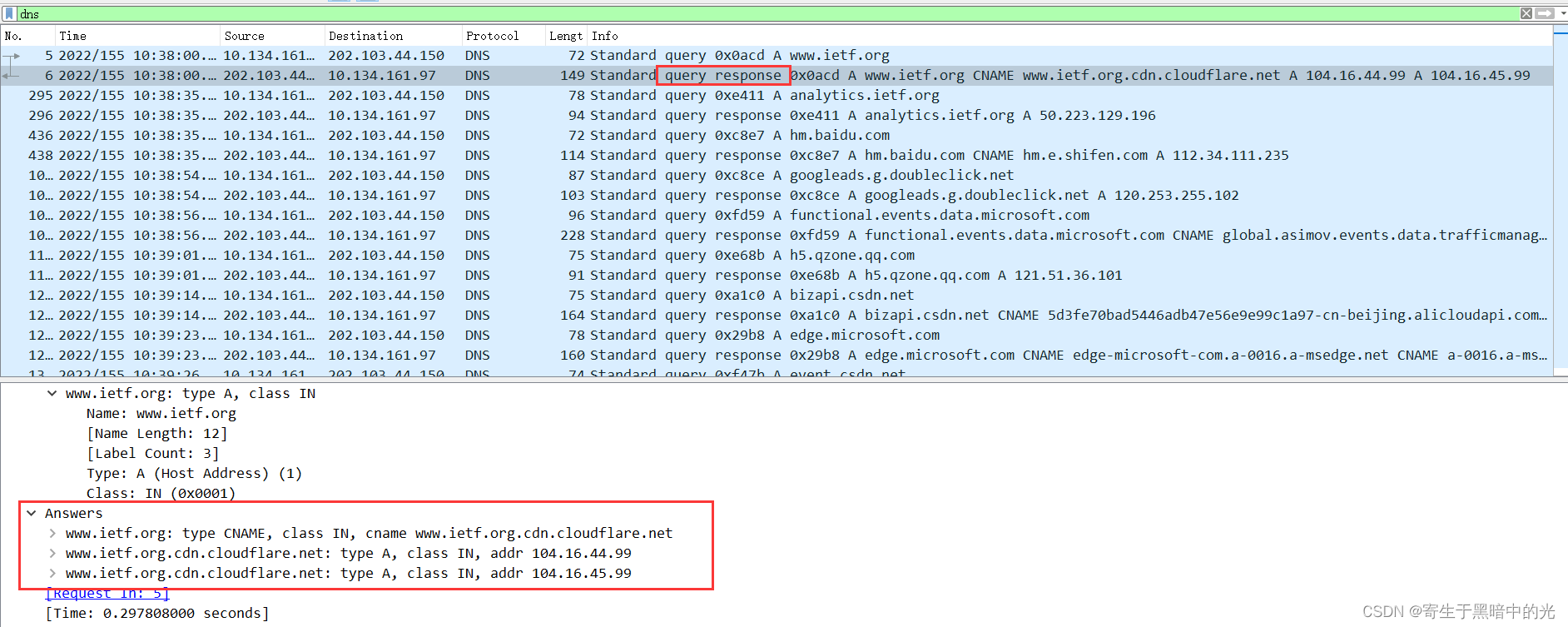
**3 DNS 查询消息发送到哪个 IP 地址?使用 ipconfig 来确定本地 DNS 服务器 的 IP 地址。这两个 IP 地址是否相同?**

****

****

相同

**4 检查 DNS 响应消息。提供了多少个"answers"?这些答案具体包含什么？**



3个