实验报告：IPV4

课程名称：计算机网络	**年级：**2023 级	**实践成绩：**
指导教师：章玥	**姓名：**张建夫	
实践名称：IPV4	**学号：**10235101477	**实践日期：**2024/12/01
实践编号：	**组号：**	**实践时间：**8:00~9:30

一、目的

1.学会通过 Wireshark 分析 ip 协议

2.了解 wireshark、curl、wget、traceroute、tracert 等常用软件的使用，掌握网络抓包的方法，能在所用电脑上进行抓包;

3.了解 IP 数据包格式，能应用该软件分析数据包格式，查看抓到的包的内容，并分析对应的 IP 数据包格式;

4.了解 IP 各部分的含义

二、实验内容与实验步骤

1.背景知识：

(1)IP 报文格式

(2)Traceroute 工具 traceroute，现代 Linux 系统称为 tracepath，Windows 系统称为 tracert，是一种电脑网络工具。 它可显示数据包在 IP 网络经过的路由器的 IP 地址。 程序是

利用增加存活时间（TTL）值来实现其功能的。每当数据包经过一个路由器，其存活时间就会 减 1。当其存活时间是 0 时，主机便取消数据包，并发送一个 ICMP TTL 数据包给原数据包的发出者。 程序发出的首 3 个数据包 TTL 值是 1，之后 3 个是 2，如此类推，它便得到一连串数据包路径。注 意，IP 不保证每个数据包走的路径都一样。

(3)检验和算法 （1）把 IP 数据包的校验和字段置为 0。（2）把首部看成以 16 位为单位的数字组成，依次进行二进制求和（注意：求和时应将最高位的 进位保存，所以加法应采用 32 位加法）。 （3）将上述加法过程中产生的进位（最高位的进位）加到低 16 位（采用 32 位加法时，即为将 高 16 位与低 16 位相加，之后还要把该次加法最高位产生的进位加到低 16 位）。 （4）将上述的和取反，即得到校验和。

2.实验步骤：

(1)捕获 IP Packets

1、启动 Wireshark，在菜单栏的捕获->选项中进行设置，选择已连接的以太网，设置捕获过滤器为 “tcp port 80”，将混杂模式设为关闭,勾选 enable network name resolution.然后开始捕获。

2、打开 windows 的命令行，在里面输入 wget www.sina.com 3、停止捕获。

(2)捕获 Trace

1.启动 Wireshark，在菜单栏的捕获->选项中进行设置，选择已连接的以太网，设置捕获过滤器为 “icmp”，将混杂模式设为关闭,勾选 enable network name resolution.然后开始捕获。

2.执行 tracert www.baidu.com 。

3. 当命令执行结束，则 Wireshake 中停止捕获，查看 Wireshark 界面中的封包列表中如果出现数据 包则说明抓包成功；

三、实验环境

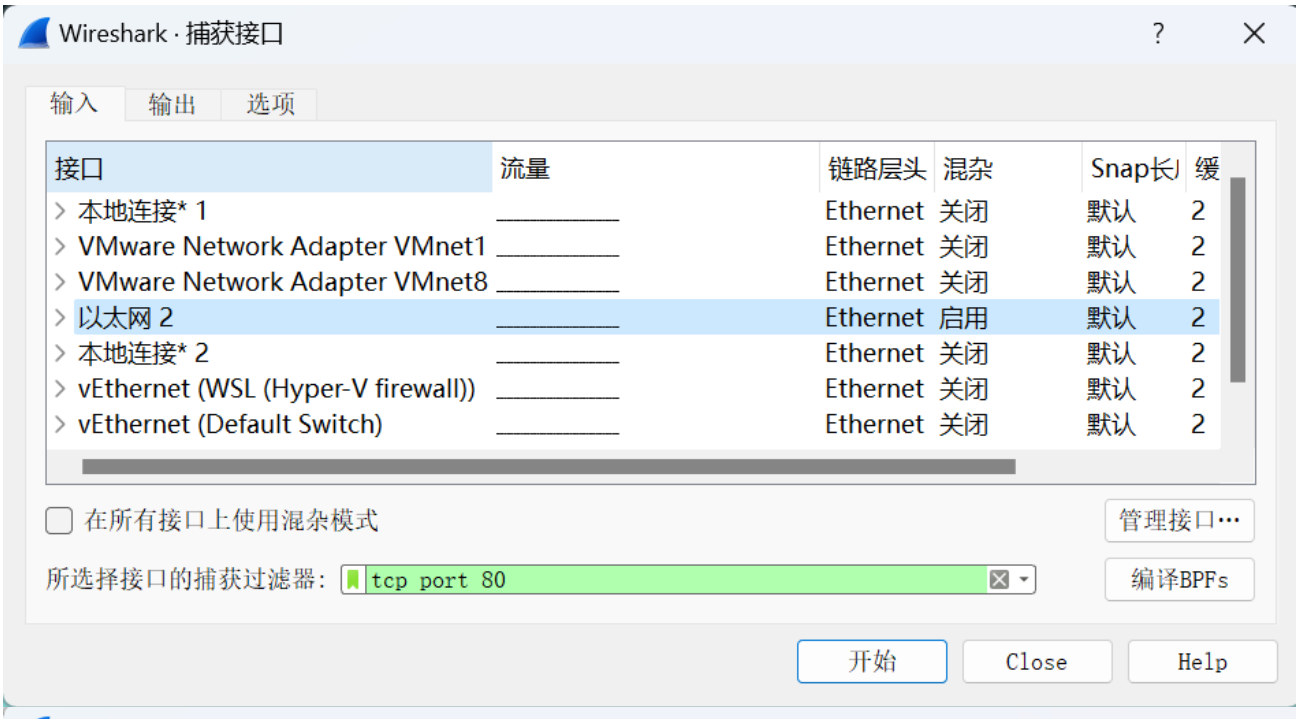
调用 dxdiag 工具：
Operating System: Windows 11 家庭中文版 64-bit (10.0, Build 22621) (22621.ni_release.220506-1250)
Language: Chinese (Simplified) (Regional Setting: Chinese (Simplified))
System Manufacturer: HP
System Model: HP Pavilion Aero Laptop 13-be2xxx
BIOS: F.13 (type: UEFI)
Processor: AMD Ryzen 5 7535U with Radeon Graphics (12 CPUs), ~2.9GHz
Memory: 16384MB RAM
Available OS Memory: 15574MB RAM
Page File: 27604MB used, 5685MB available
Windows Dir: C:\WINDOWS
DirectX Version: DirectX 12
DX Setup Parameters: Not found
User DPI Setting: 144 DPI (150 percent)
System DPI Setting: 192 DPI (200 percent)
DWM DPI Scaling: UnKnown

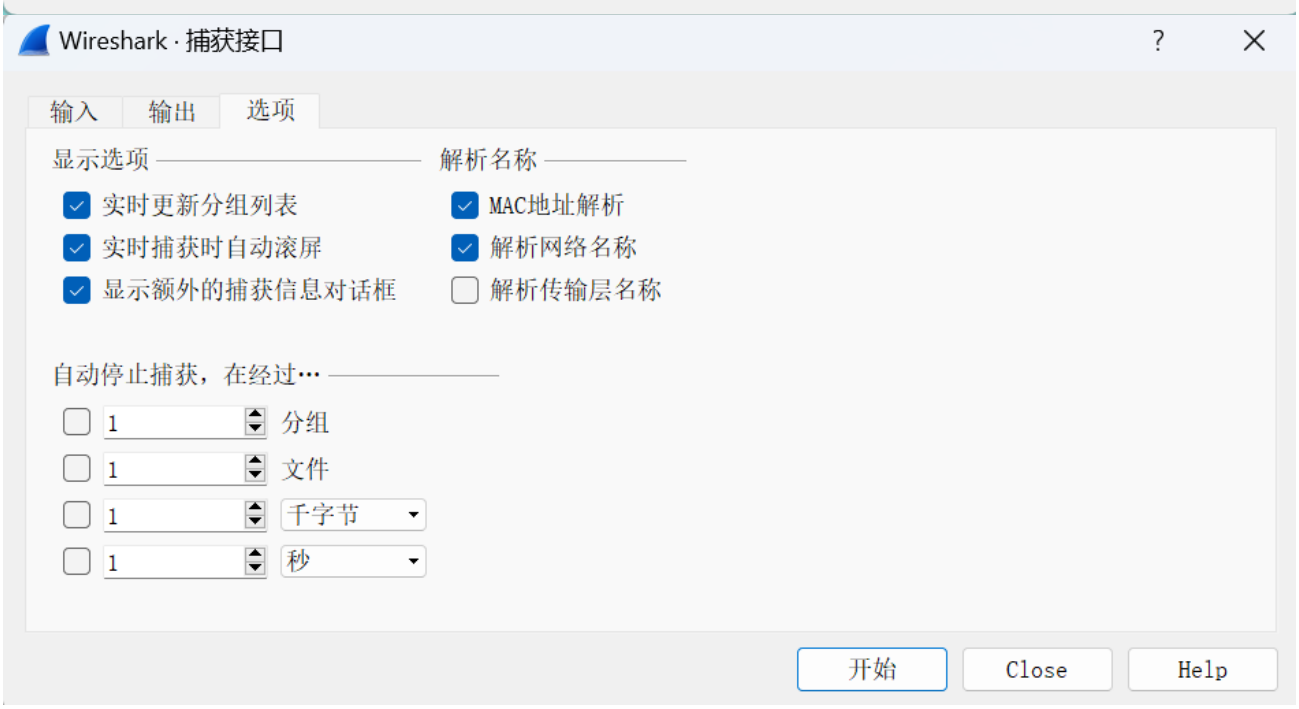
Miracast: Available, with HDCP
Microsoft Graphics Hybrid: Not Supported

四、实验过程与分析

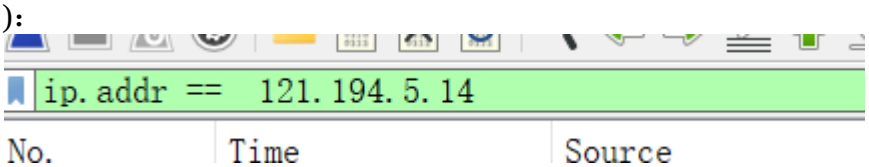
1）捕获 ip 包：

先进行捕获选项设置：

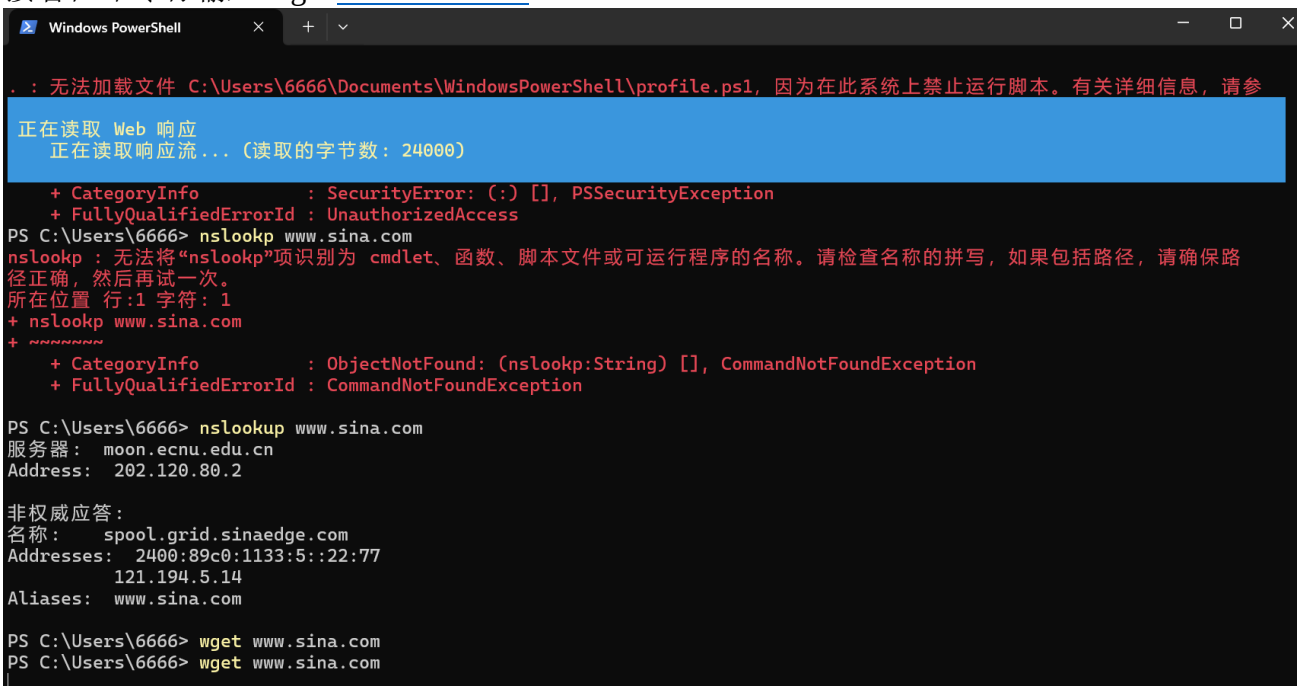




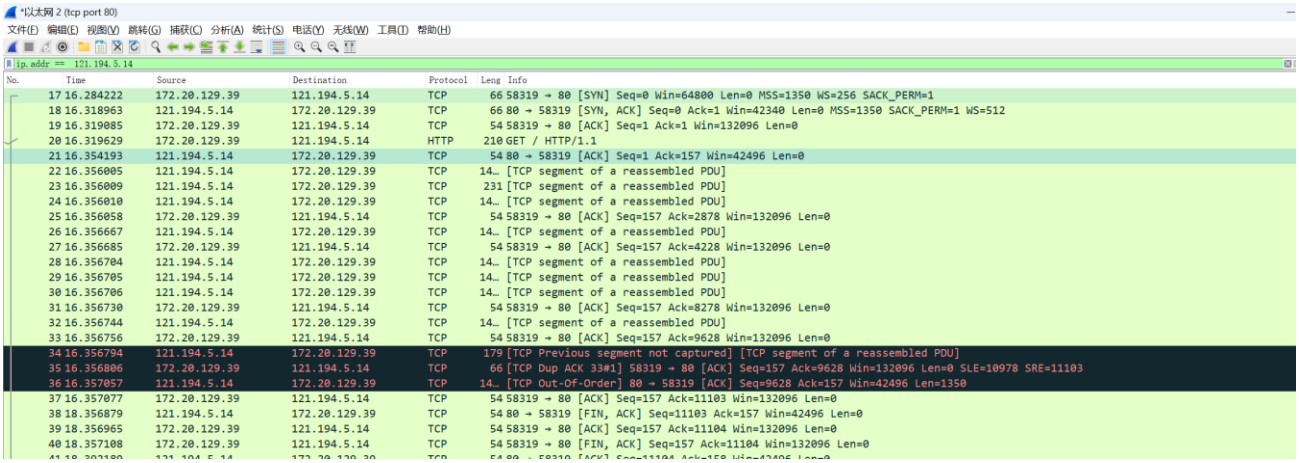
然后设置过滤器只显示 sina 服务器的流量(sina 服务器的 ip 地址可以通过命令 nslookup 拿到)：



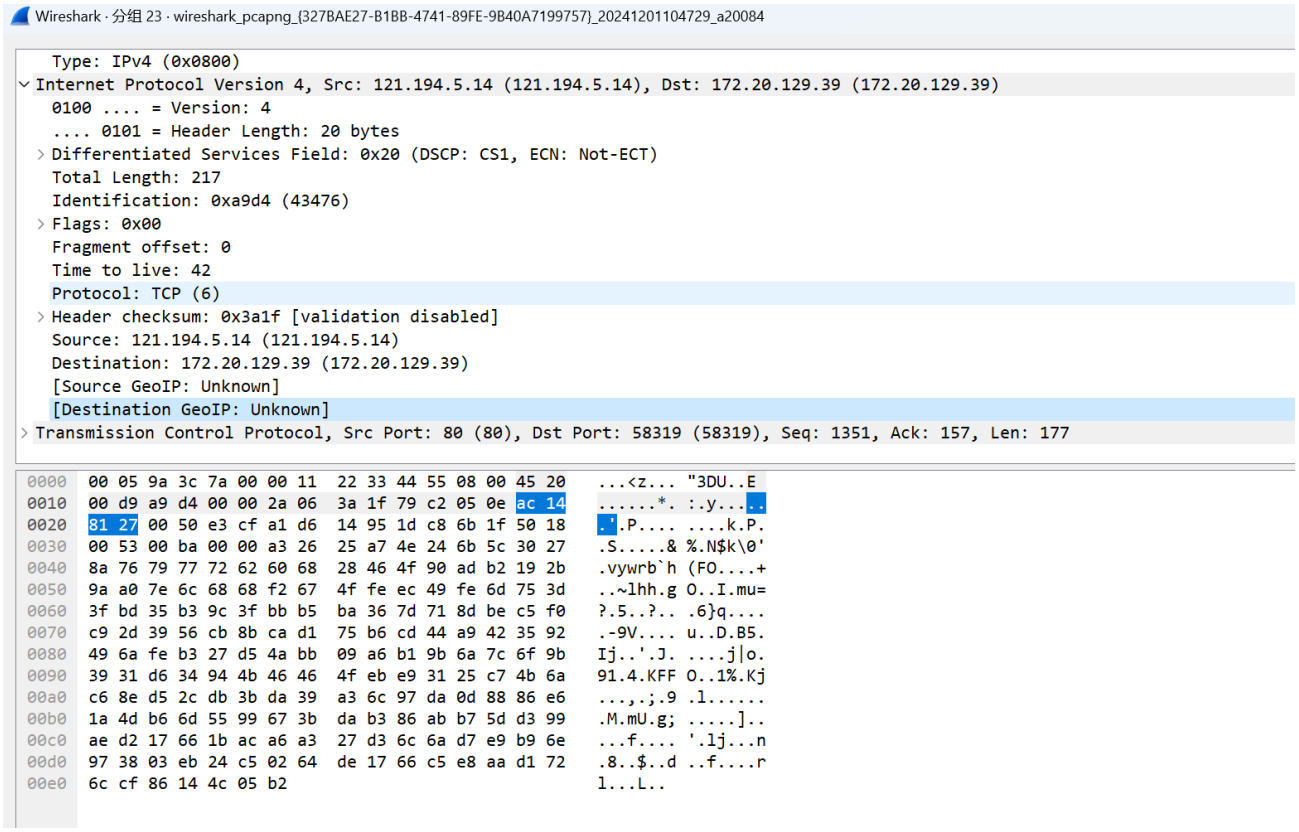
接着在命令行输入 wget www.sina.com：



在 wireshark 中抓到如下包：



接下来分析其中一个包：



对报文信息进行分析：

1. 尽量不要打开教科书，请根据抓到的报文，描绘 IP 报文头的图片，并指出报文中每个 fields 的含义所在。

Ip 报文头的描述如下

4 位 版本 0100（4）	4 位 首部长度 0101（5）	1 字节 区分服务 0x20	2 字节 总长度 0x00d9
2 字节 标识符 0xa9d4		3 位 标志 000	13 位 片偏移 0000000000000
1 字节 生存时间 0x2a	1 字节 协议 0x06	2 字节 首部检验和 0x3a1f	
4 字节 源地址 121.194.5.14			
4 字节 目的地址 172.20.129.39			

2. 你的计算机和远程服务器的 IP 地址是什么？

本地计算机：172.20.129.39

远程服务器：121.194.5.14

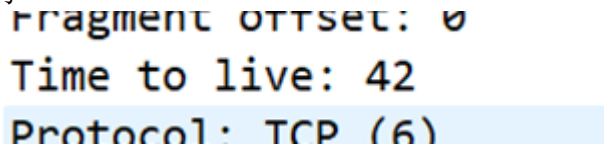
3.“总长度”字段是否包括 IP 报头加上 IP 有效负载，或者仅包括 IP 有效负载？

总长度为 217，远超 ip 头部的 20 字节，因此包含 ip 头部加上 ip 有效负载

4. 对于不同的数据包，“标识”字段的值如何变化，还是保持不变？ 例如，对于 TCP 连接中的所 有数据包，它一直保持相同的值，还是对于每个数据包都不同？ 双向通信的报文是否相同？ 如果值发生 变化，您能看到任何规律吗？

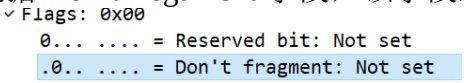
（1）同个连接中的标识字段可能相同（如果一个包被分片了，分成不同的数据包，则它们的标识符相同）；
（2）不同传输方向的标识符一定不同，因为发送端和接受端都是各自标识的；
（3）标识字段的规律是，IP 在存储器中维护一个计数器，每产生一个数据报，计数器就加 1，并将此值赋给标识字段。分片，保持一致。

5. 从您的计算机发送的数据包的 TTL 字段的初始值是多少？他们是 maximum possible value 吗？



由图可知 ttl 为 42，由于 ttl 字段是 1 个字节，所以最大是 255，42 不是最大可能值

6. 查看数据包时如何判断它是否被分段？
根据 Don’t fregament 字段，该字段通常为一位，1 表示没有分段，0 表示有分段。

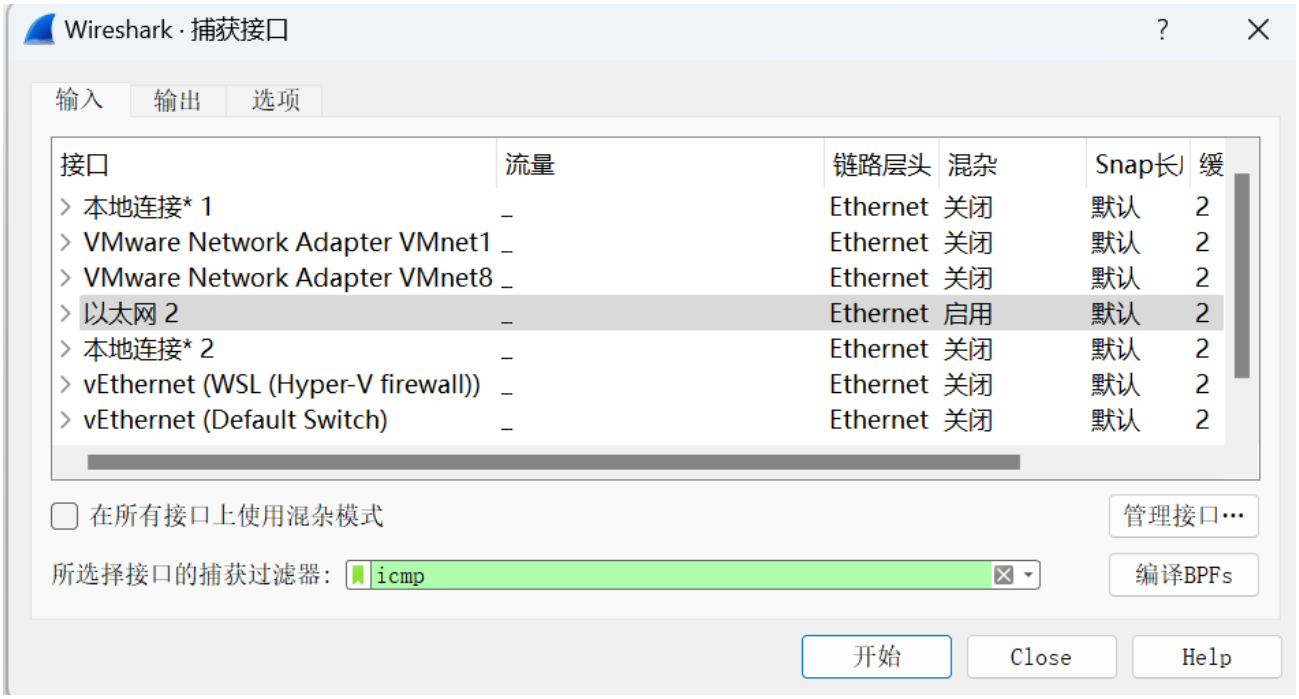


如 表示有分段

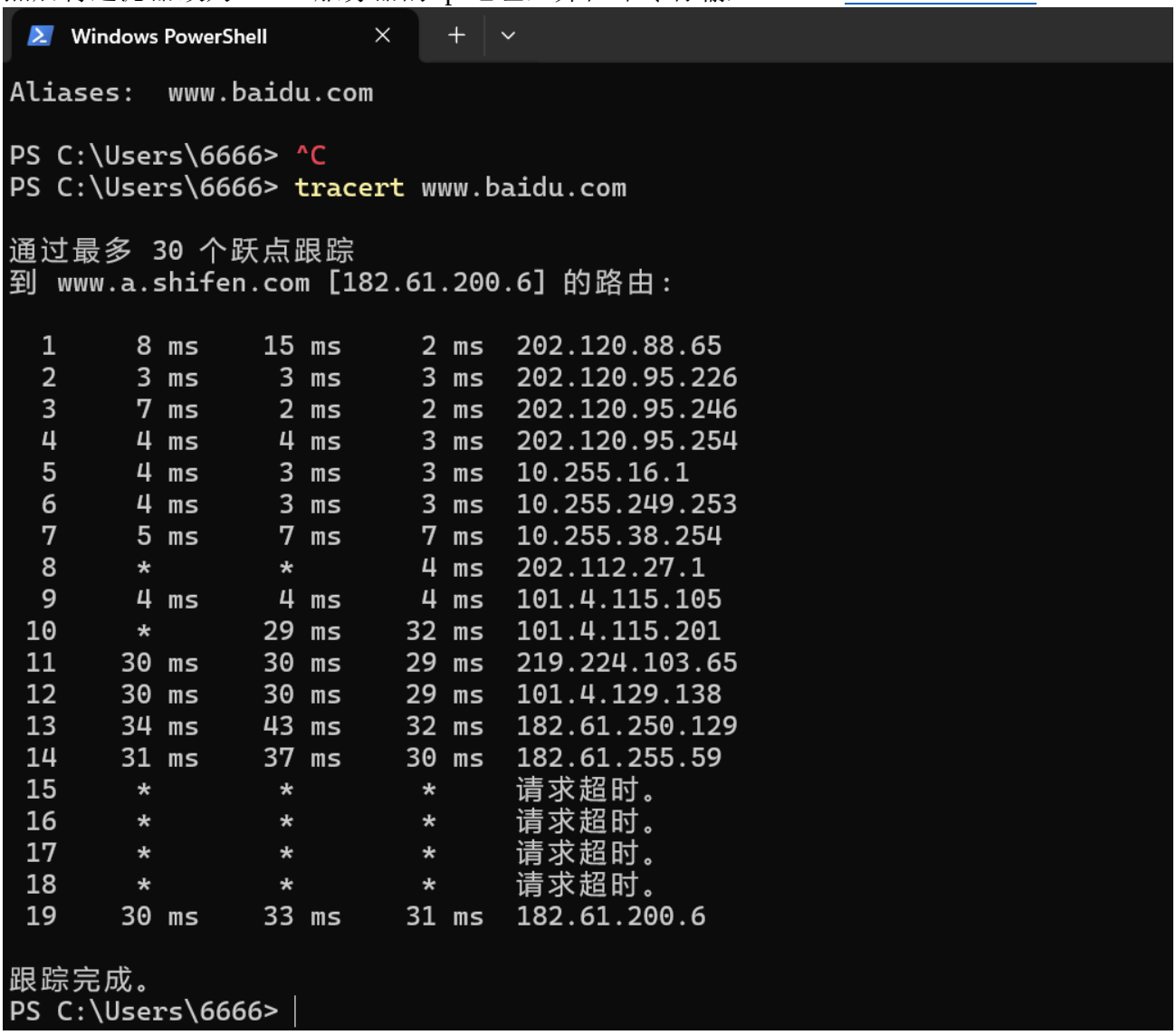
7. IP 数据报报头的长度是多少，它是如何被编码进报头长度域的？
（1）IP 报头分为 2 部分，前半部分为固定部分，固定部分的长度为 20 字节，而后半部分为填充部分，它的长度是不固定的。
（2）报头长度域以 4 字节（32 位）为单位进行计数，而固定部分有 20 字节，故报头长度域为 5（0101）

2）捕获 trace

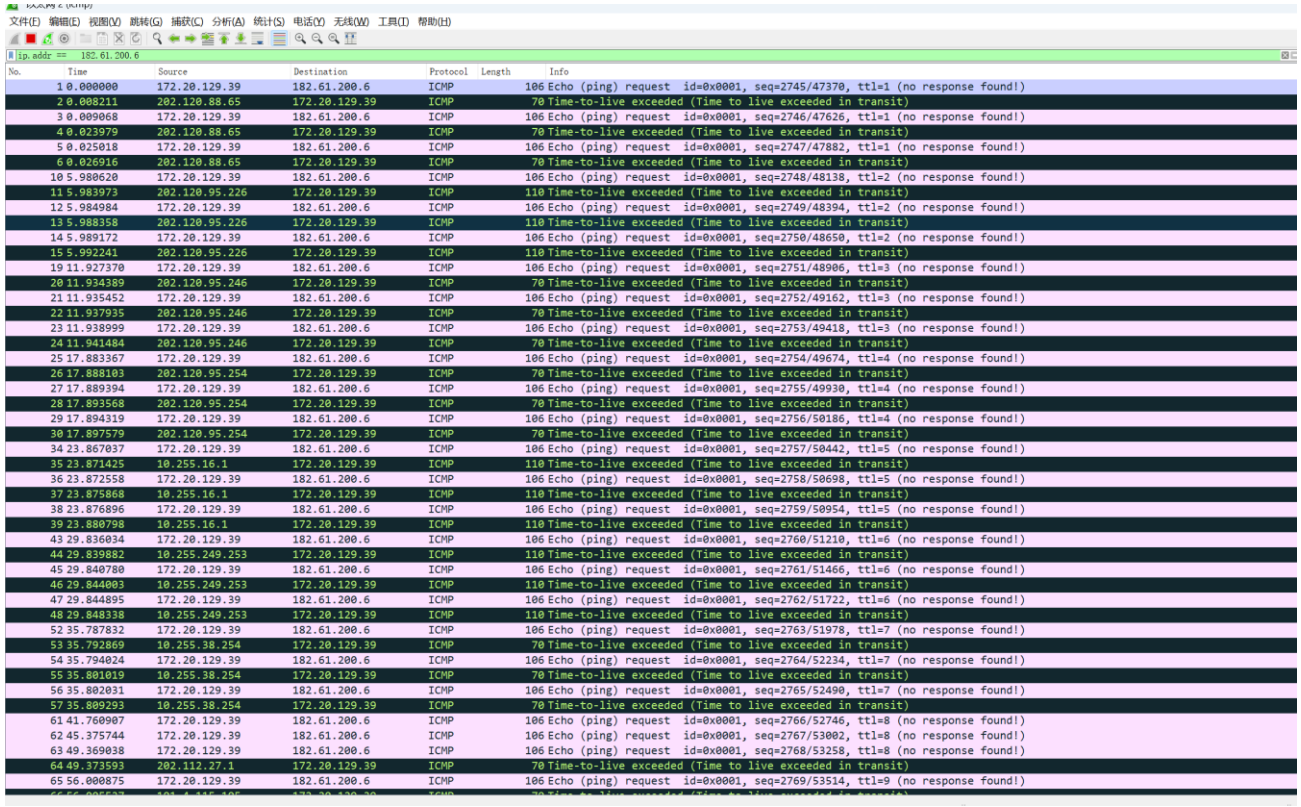
1.先设置捕获选项：



然后将过滤器改为 baidu 服务器的 ip 地址，并在命令行输入 tracert www.baidu.com：

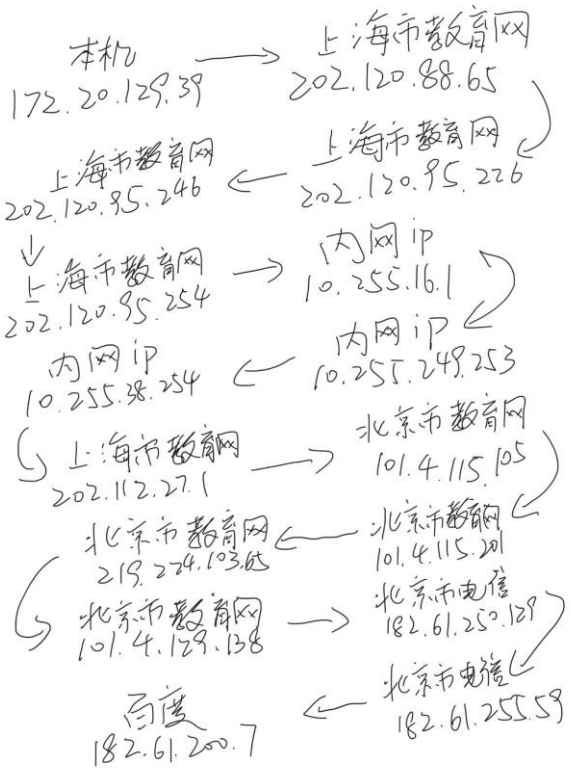


Wireshark 中抓包列表如下：



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2745/47370, ttl=1 (no response found!)
2	0.008211	202.120.88.65	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
3	0.009068	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2746/47626, ttl=1 (no response found!)
4	0.023979	202.120.88.65	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
5	0.025018	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2747/47882, ttl=1 (no response found!)
6	0.026916	202.120.88.65	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
10	5.980620	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2748/48138, ttl=2 (no response found!)
11	5.983973	202.120.95.226	172.20.129.39	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
12	5.984984	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2749/48394, ttl=2 (no response found!)
13	5.988358	202.120.95.226	172.20.129.39	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
14	5.989172	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2750/48650, ttl=2 (no response found!)
15	5.992241	202.120.95.226	172.20.129.39	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
19	11.927370	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2751/48906, ttl=3 (no response found!)
20	11.934389	202.120.95.246	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
21	11.935452	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2752/49162, ttl=3 (no response found!)
22	11.937935	202.120.95.246	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
23	11.938999	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2753/49418, ttl=3 (no response found!)
24	11.941484	202.120.95.246	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
25	17.883367	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2754/49674, ttl=4 (no response found!)
26	17.888103	202.120.95.254	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
27	17.889394	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2755/49930, ttl=4 (no response found!)
28	17.893568	202.120.95.254	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
29	17.894319	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2756/50186, ttl=4 (no response found!)
30	17.897579	202.120.95.254	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
34	23.867037	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2757/50442, ttl=5 (no response found!)
35	23.871425	10.255.16.1	172.20.129.39	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
36	23.872558	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2758/50698, ttl=5 (no response found!)
37	23.875868	10.255.16.1	172.20.129.39	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
38	23.876896	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2759/50954, ttl=5 (no response found!)
39	23.880798	10.255.16.1	172.20.129.39	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
43	29.836034	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2760/51210, ttl=6 (no response found!)
44	29.839882	10.255.249.253	172.20.129.39	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
45	29.840780	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2761/51466, ttl=6 (no response found!)
46	29.844003	10.255.249.253	172.20.129.39	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
47	29.844895	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2762/51722, ttl=6 (no response found!)
48	29.848338	10.255.249.253	172.20.129.39	ICMP	110	Time-to-live exceeded (Time to live exceeded in transit)
52	35.787832	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2763/51978, ttl=7 (no response found!)
53	35.792869	10.255.38.254	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
54	35.794024	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2764/52234, ttl=7 (no response found!)
55	35.801019	10.255.38.254	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
56	35.802031	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2765/52490, ttl=7 (no response found!)
57	35.809293	10.255.38.254	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
61	41.760907	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2766/52746, ttl=8 (no response found!)
62	45.375744	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2767/53002, ttl=8 (no response found!)
63	49.369038	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2768/53258, ttl=8 (no response found!)
64	49.373593	202.112.27.1	172.20.129.39	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
65	56.000875	172.20.129.39	182.61.200.6	ICMP	106	Echo (ping) request id=0x0001, seq=2769/53514, ttl=9 (no response found!)

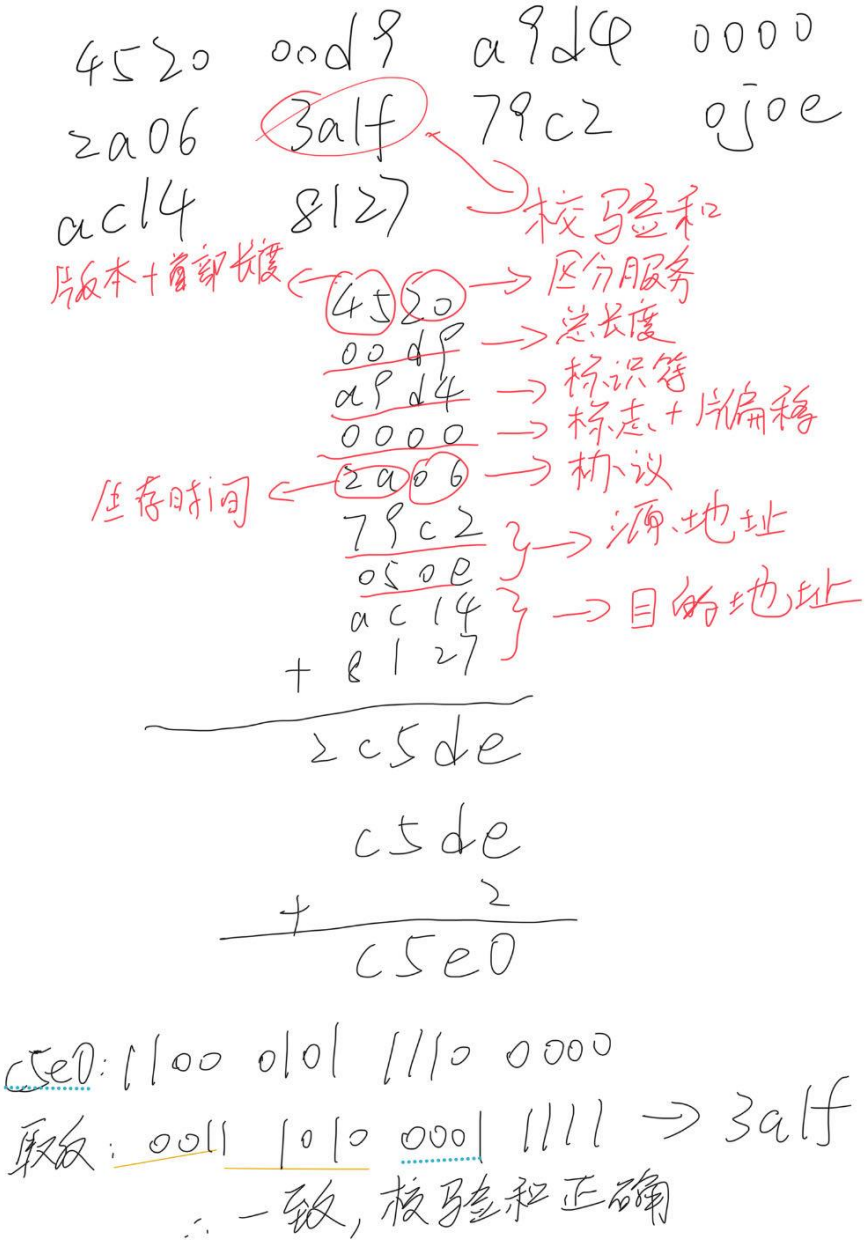
接下来分析 traceroute 的结果图：（使用 traceroute 的结果，绘制网络路径图。图中，显示您的计算机（放在最左侧）和远程服务器（放在最右侧），均显示 IP 地址，以及它们之间的路径上的路由器，这些路由器以从本机开始的跳数作为距离编号。您可以在捕获的跟踪数据包中找到计算机和远程服务器的 IP 地址。）



最后计算捕获 ip 包中的校验和是否正确（IP 报头的校验和可以用来验证一个数据包是否正确。选择一个从远程服务器发送到本计算机的包，计算它的 checksum。在计算过程中，请添加注释，表明每个 word 对应的字段。）：

Wireshark · 分组 23 · wireshark_pcapng_{327BAE27-B1BB-4741-89FE-9B40A7199757}_20241201104729_a20084

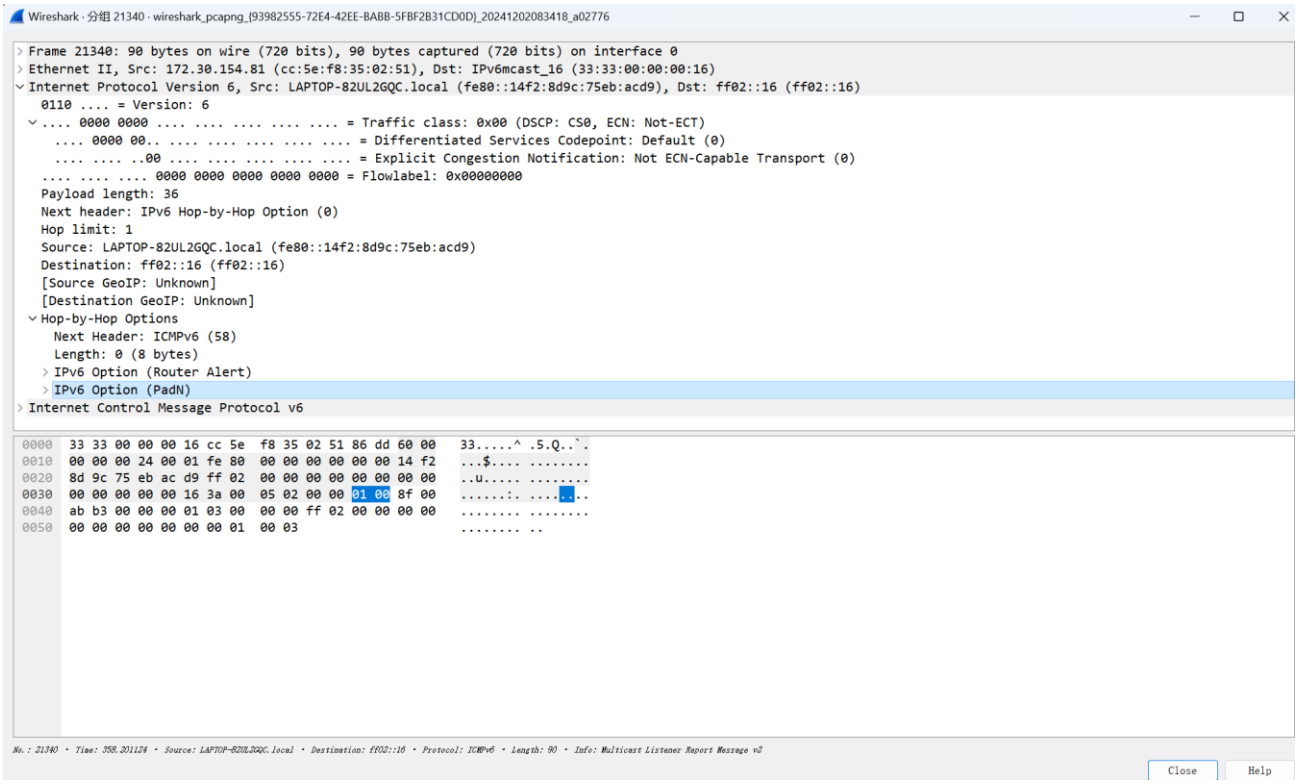
```
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 121.194.5.14 (121.194.5.14), Dst: 172.20.129.39 (172.20.129.39)
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
  > Differentiated Services Field: 0x20 (DSCP: CS1, ECN: Not-ECT)
    Total Length: 217
    Identification: 0xa9d4 (43476)
  > Flags: 0x00
    Fragment offset: 0
    Time to live: 42
    Protocol: TCP (6)
  > Header checksum: 0x3a1f [validation disabled]
    Source: 121.194.5.14 (121.194.5.14)
    Destination: 172.20.129.39 (172.20.129.39)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
> Transmission Control Protocol, Src Port: 80 (80), Dst Port: 58319 (58319), Seq: 1351, Ack: 157, Len: 177
```



五、实验结果总结

思考题：

1. 了解并尝试使用 IPv6。 现代操作系统已经包含对 IPv6 的支持，因此您可能能够捕获网络上的 IPv6 流量。 您还可以通过 tunnels 连接到 IPv6



我并没有使用 tunnels 实现 ipv6 的抓包，也可以使用 teredo 进行 ipv6 的抓包。

2. 了解 tunnels 技术。

Tunnels 技术通常指的是通过网络将数据包从一个地点安全传输到另一个地点的方法。它可以用 于加密和保护数据通信，以确保在公共网络上的安全性。常见的应用包括 VPN（虚拟专用网络）和 SSH（安全外壳协议）等，用于安全地连接远程网络或主机。这些技术通过在通信路径上创建加密的 "隧道"，防止第三方截取或篡改传输的数据。

3. 了解有关 IP 的地理位置信息，即 IP 地址和它对应的地理位置之间的信息。

IP 地址的地理位置信息是通过将 IP 地址映射到地理坐标的过程来获取。这通常使用 IP 地理位置数据库进行，这些数据库包含了全球范围内的 IP 地址和相应地理位置的映射。这些数据库通过分析网络流量、合并多个信息源等方式收集数据。然后，通过查询这些数据库，可 以获取 IP 地址对应的大致地理位置，包括国家、地区、城市、经度和纬度等信息。一些服务和网站使用 这些信息来提供有关访问者所在位置的服务或内容。

4. 了解 IPsec 或 IP security。 它为 IP 数据包提供机密性和身份验证，通常用作 VPN 的一部分。

IPsec（Internet Protocol Security）是一组协议和标准，旨在为 IP（Internet Protocol）通信提供安全性服务，包括机密性、数据完整性和身份验证。它常用于构建虚拟专用网络（VPN）以保护通 过公共网络传输的数据。 IPsec 通过两个主要协议来实现其目标：

1. AH（Authentication Header）：提供身份验证和数据完整性，但不提供加密。AH 通过在 IP 包头 部添加认证信息来验证发送方的身份，并确保数据在传输过程中没有被篡改。 2. ESP（Encapsulating Security Payload）：提供机密性、身份验证和可选的数据完整性。ESP 通 过加密 IP 包中的数据来保护通信的隐私，并可以选择提供身份验证和数据完整性功能。 IPsec 可以用于点对点连接或作为 VPN 的一部分，创建安全的通信隧道。在 VPN 中，它允许远程用户 或分支机构通过公共网络安全地连接到企业网络，确保数据在传输过程中得到保护。

六、个人总结

本次实验让我对 ip 的头部有了很清晰的认识，对于每一个 ip 的字段的作用也有了较深的了解，同时也了解到了 tracert 命令的使用，还有一个最重要的收获就是真正计算了一次实际应用中的 checksum，这些都让我对计算机网络有了更深的理解，对于理论课的学习也有很大的帮助。