



同濟大學
TONGJI UNIVERSITY

计算机网络课程报告 wireshark抓包分析

姓 名 郑博远

学 号 2154312

学 院 电子与信息工程学院

专 业 计算机科学与技术

授课老师 陆有军

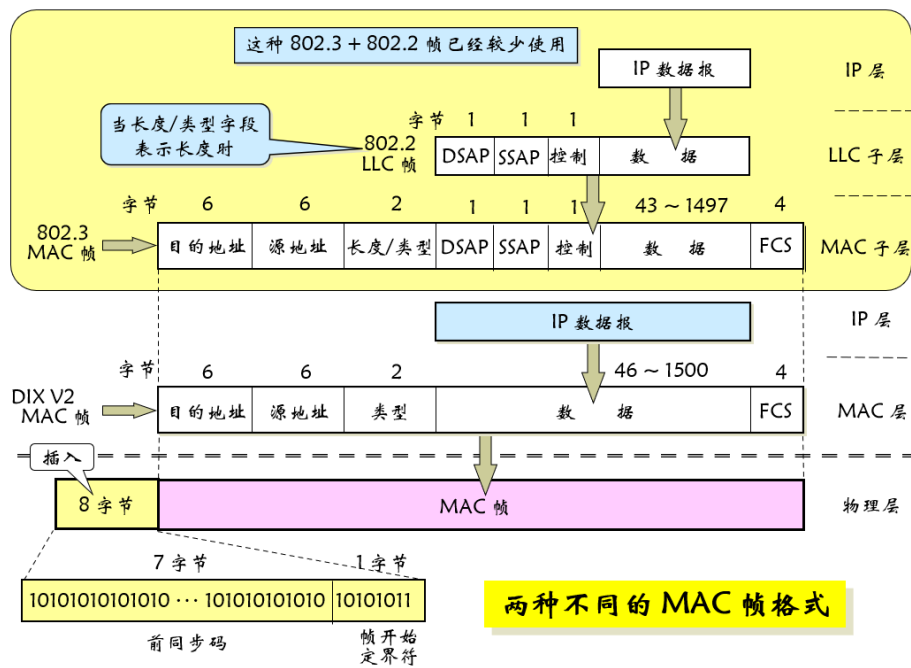
日 期 2024 年 6 月

一、以太网 MAC 帧。

1. 以太网 MAC 帧格式简介：

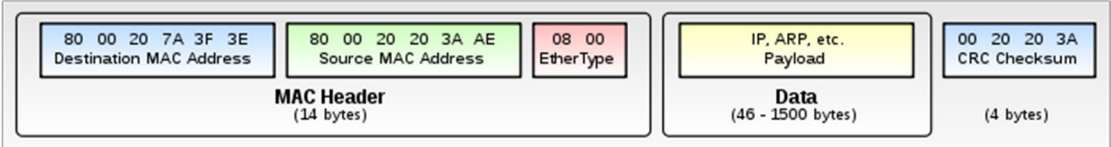
在以太网链路上的数据包称作以太帧，其位于 OSI 七层模型的第二层。

以太网有 DIX V2 和 IEEE 802.3 两种帧格式，其中后者目前很少使用。接下来的实验过程中捕获到的也是 DIX V2 帧格式的帧。二者具体区别见下图：



下面以前者为例介绍以太网 MAC 帧的帧格式：

以太网 MAC 帧的起始部分是由物理层插入的前同步码和帧开始定界符。此后，是包含目的 MAC 地址与源 MAC 地址字段以及类型字段的首部。其中类型字段描述了帧首部后面所跟数据包的类型，如 IP 协议包（0x8000）、ARP 协议包（0x8060）等。接着，帧的中部是该帧负载的包含高层协议（例如 IP 协议）首部的数据包。最后，MAC 帧由 32 位冗余校验码的 FCS 字段结尾。



2. 捕获以太网 MAC 帧实验步骤:

由于本次报告中所使用的笔记本电脑设备没有通过以太网连接其他设备,因此使用 Windows 系统的 Linux 子系统 (WSL2) 与 Windows 宿主机的虚拟以太网连接来完成以太网 MAC 帧的捕获分析。

1) 查看 Windows 与 WSL2 的 MAC 地址:

```
zhengboyuan@LAPTOP-CNS4I 命令提示符
描述. . . . . : Sangfor SSL VPN CS Support System VNIC
物理地址. . . . . : 00-FF-55-B9-CD-F1
DHCP 已启用. . . . . : 是
自动配置已启用. . . . . : 是

以太网适配器 蓝牙网络连接:

媒体状态. . . . . : 媒体已断开连接
连接特定的 DNS 后缀. . . . . :
描述. . . . . : Bluetooth Device (Personal Area Network)
物理地址. . . . . : CC-6B-1E-8C-05-C4
DHCP 已启用. . . . . : 是
自动配置已启用. . . . . : 是

以太网适配器 vEthernet (WSL (Hyper-V firewall)):

连接特定的 DNS 后缀. . . . . :
描述. . . . . : Hyper-V Virtual Ethernet Adapter
物理地址. . . . . : 00-15-5D-8B-3A-DD
DHCP 已启用. . . . . : 否
自动配置已启用. . . . . : 是
本地链接 IPv6 地址. . . . . : fe80::e9ee:adb3:7d1f:cb6a%62(首选)
IPv4 地址. . . . . : 172.20.32.1(首选)
子网掩码. . . . . : 255.255.240.0
默认网关. . . . . :
DHCPv6 IAID. . . . . : 1040192861
DHCPv6 客户端 DUID. . . . . : 00-01-00-01-2D-8A-AC-7F-C6-C1-7D-E3-F9-DB
TCP/IP 上的 NetBIOS. . . . . : 已启用

C:\Users\BoyuanZheng>
```

```
zhengboyuan@LAPTOP-CNS4NHBS:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:86:f4:b9 brd ff:ff:ff:ff:ff:ff
    inet 172.20.39.101/20 brd 172.20.47.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe86:f4b9/64 scope link
        valid_lft forever preferred_lft forever
zhengboyuan@LAPTOP-CNS4NHBS:~$
```

2) 在 Windows 系统中 ping WS2 对应 IP 地址, 在 wireshark 捕获该以太网:

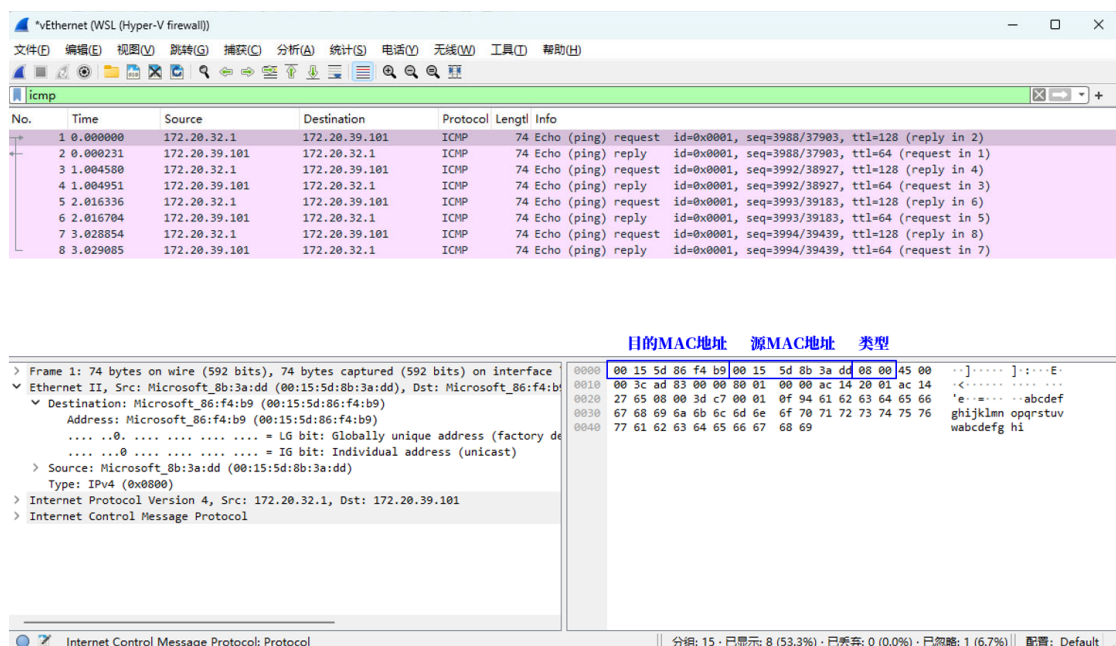
```
zhengboyuan@LAPTOP-CNS4I  命令提示符

C:\Users\BoyuanZheng>ping 172.20.39.101

正在 Ping 172.20.39.101 具有 32 字节的数据:
来自 172.20.39.101 的回复: 字节=32 时间=1ms TTL=64
来自 172.20.39.101 的回复: 字节=32 时间<1ms TTL=64
来自 172.20.39.101 的回复: 字节=32 时间<1ms TTL=64
来自 172.20.39.101 的回复: 字节=32 时间<1ms TTL=64

172.20.39.101 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 1ms, 平均 = 0ms

C:\Users\BoyuanZheng>
```



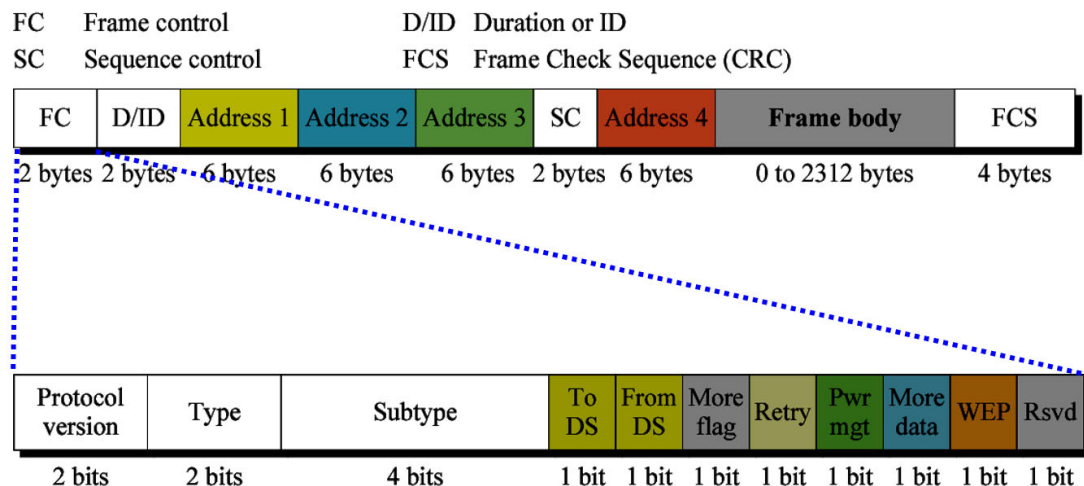
3) 帧格式分析:

如上图所示, 可以清晰地看到以太网 MAC 帧首部中, 按顺序包含了目标 MAC 地址 (**00:15:5d:86:f4:b9**)、源 MAC 地址 (**00:15:5d:8b:3a:dd**) 及数据包类型 (**0x0800**, 对应 IP 协议)。其次, 观察到起始的 8 字节前同步码等信息和末尾的 4 字节的校验部分没有出现, 这是由于网卡已经使用校验码进行了正确性校验, 并进行了对应部分的去除; 因此 wireshark 捕获到的包不包含对应字段。

二、无线局域网 MAC 帧。

1. 无线局域网 MAC 帧格式简介：

无线局域网（WLAN）是不使用任何导线或传输电缆连接的局域网。无线局域网最通用的标准是 IEEE 定义的 802.11 系列标准，也是本次捕获的帧格式。在 IEEE 802.11 WLAN 协议中，MAC 帧由公共字段和特定字段构成。



首先分析前两字节的帧控制字段：

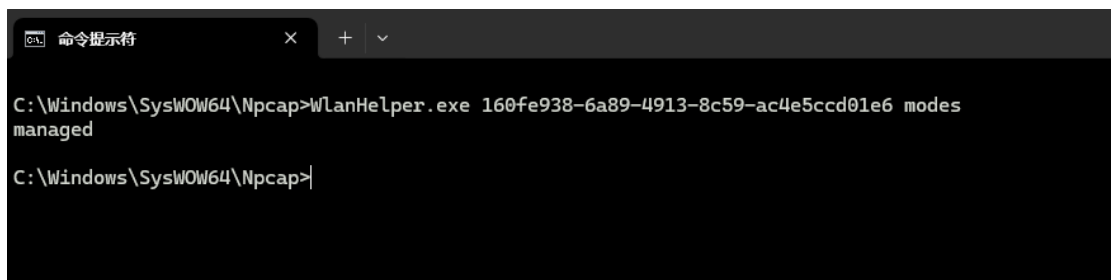
- 协议版本：2 bit，代表协议版本，当前使用的协议版本为 0，其他值保留；
- 类型：2 bit，识别 WLAN 帧的类型。IEEE 802.11 中定义的帧类型包含管理（00）、控制（01）和数据（10）；
- 子类型：4 bit，提供帧之间的额外区分。包含 RTS（1011）、CTS（1100）和 ACK（1101）；
- ToDS 和 FromDS：各 1 bit。表示数据帧是否前往分配系统。控制和管理帧将该字段设置为零。所有数据帧都将设置这些位中的一个；
- 更多片段：1 bit。当数据包分成多个帧进行传输时，非末尾帧该字段置 1；
- 重试：1 bit。有时帧需要重传，此时该字段设置为 1；
- 电源管理：1 bit。该位表示帧交换完成后发送方的电源管理状态；

- 更多数据：1 bit。表示更多数据位用于缓冲分布式系统中收到的帧；
- 受保护帧：1 bit。如果帧被有线等效保密（WEP）、Wi-Fi 保护接入（WPA）等机制加密，则该位设置为 1；
- 顺序：1 bit。仅在采用“严格排序”时被设置。

下面分析帧中的其他字段：

- 持续时间 ID（2 字节）：表明该字段的传输需要多长时间。这个字段可以采取三种形式之一：持续时间、无争论期（CFP）和协会 ID（AID）；
- 地址字段（各 6 字节）：一个帧最多可以有四个地址字段，每个字段可以携带一个 MAC 地址。地址 1 是接收方，地址 2 是发送方，地址 3 用于接收方的过滤目的。地址 4 只存在于扩展服务集的接入点之间或网状网络的中间节点之间传输的数据帧中；
- 顺序控制字段 SC（2 字节）：识别信息顺序和消除重复帧。前 4 位用于分片号，后 12 位为序列号；
- 帧检查序列 FCS（4 字节）：在帧的尾部。进行循环冗余检查（CRC），用于判断该帧在传输过程中没有失真。

2. 捕获无线局域网 MAC 帧实验步骤：



```
C:\Windows\SysWOW64\Npcap>WlanHelper.exe 160fe938-6a89-4913-8c59-ac4e5ccd01e6 modes managed
C:\Windows\SysWOW64\Npcap>
```

由于本次报告中所使用的笔记本电脑设备使用的网卡不支持 monitor 监听，无法用 wireshark 捕获。因此改用 Microsoft Network Monitor 进行下述实验。

1) 查看 Windows 的 MAC 地址:

```
命令提示符

无线局域网适配器 WLAN:

    连接特定的 DNS 后缀 . . . . . : tongji.edu.cn
    描述 . . . . . : MediaTek Wi-Fi 6 MT7921 Wireless LAN Card
    物理地址 . . . . . : CC-6B-1E-8C-05-C3
    DHCP 已启用 . . . . . : 是
    自动配置已启用 . . . . . : 是
    IPv6 地址 . . . . . : 2001:da8:8002:6bd1:7b5f:f329:a345:2d51(首选)
    临时 IPv6 地址 . . . . . : 2001:da8:8002:6bd1:1c70:d14e:23ed:f600(首选)
    本地链接 IPv6 地址 . . . . . : fe80::c896:8b34:bc00:f8e4%5(首选)
    IPv4 地址 . . . . . : 100.80.165.60(首选)
    子网掩码 . . . . . : 255.254.0.0
    获得租约的时间 . . . . . : 2024年6月6日 23:01:04
    租约过期的时间 . . . . . : 2024年6月7日 17:51:16
    默认网关 . . . . . : fe80::9e54:c2ff:fe0d:5002%5
    . . . . . : 100.81.255.254
    DHCP 服务器 . . . . . : 100.81.255.254
    DHCPv6 IAID . . . . . : 97282846
    DHCPv6 客户端 DUID . . . . . : 00-01-00-01-2D-8A-AC-7F-C6-C1-7D-E3-F9-DB
    DNS 服务器 . . . . . : 202.120.190.208
    . . . . . : 202.120.190.108
```

2) 在 Windows 系统中 ping baidu.com:

```
命令提示符

C:\Users\BoyuanZheng>ping baidu.com

正在 Ping baidu.com [110.242.68.66] 具有 32 字节的数据:
来自 110.242.68.66 的回复: 字节=32 时间=28ms TTL=47
来自 110.242.68.66 的回复: 字节=32 时间=28ms TTL=47
来自 110.242.68.66 的回复: 字节=32 时间=29ms TTL=47
来自 110.242.68.66 的回复: 字节=32 时间=27ms TTL=47

110.242.68.66 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 27ms, 最长 = 29ms, 平均 = 28ms
```

3) 在 Microsoft Network Monitor 对 WLAN 创建 capture, 将 baidu.com 对应的 IP 地址设为 filter 条件。捕获到如下 8 个帧 (4 次 ping):

Display Filter [Not Applied]:

Apply Remove History Load Filter Save Filter Clear Text

ipV4.address == 110.242.68.66

Frame Summary - ipV4.address == 110.242.68.66

Frame Number	Time Date Local Adjusted	Time Offset	Process Name	Source	Destination	Protocol Name	Description	Conv Id
51	0:52:14 2024/6/8	2.4673860		100.76.85.21	110.242.68.66	ICMP	ICMP-Echo Request Message, From 100.76.85.21 To 110.242.68.66	[IPv4...
52	0:52:14 2024/6/8	2.4950729		110.242.68.66	100.76.85.21	ICMP	ICMP-Echo Reply Message, From 110.242.68.66 To 100.76.85.21	[IPv4...
56	0:52:15 2024/6/8	3.4799191		100.76.85.21	110.242.68.66	ICMP	ICMP-Echo Request Message, From 100.76.85.21 To 110.242.68.66	[IPv4...
57	0:52:15 2024/6/8	3.5077159		110.242.68.66	100.76.85.21	ICMP	ICMP-Echo Reply Message, From 110.242.68.66 To 100.76.85.21	[IPv4...
60	0:52:16 2024/6/8	4.4926215		100.76.85.21	110.242.68.66	ICMP	ICMP-Echo Request Message, From 100.76.85.21 To 110.242.68.66	[IPv4...
61	0:52:16 2024/6/8	4.5200066		110.242.68.66	100.76.85.21	ICMP	ICMP-Echo Reply Message, From 110.242.68.66 To 100.76.85.21	[IPv4...
66	0:52:17 2024/6/8	5.5057310		100.76.85.21	110.242.68.66	ICMP	ICMP-Echo Request Message, From 100.76.85.21 To 110.242.68.66	[IPv4...
67	0:52:17 2024/6/8	5.5331193		110.242.68.66	100.76.85.21	ICMP	ICMP-Echo Reply Message, From 110.242.68.66 To 100.76.85.21	[IPv4...

Frame Details

Frame: Number = 51, Captured Frame Length = 124, MediaType = Wi-Fi

Wi-Fi: [Unencrypted Data] .T....., (I)

MetaData:

- FrameControl: Version 0, Data, Data, .T.....(0x108)
- Duration: 32769 (0x8000)
- BSSID: 948A12 5B08F1
- SA: CC6B1E 8C05C3
- DA: 9C54C2 0DC002
- SequenceControl: Sequence Number = 0

LLC: Unnumbered(U) Frame, Command Frame, SSAP = SNAP (Sub-Network Access Protocol), DS

Snap: EtherType = Internet IP (IPv4), OrgCode = XEROX CORPORATION

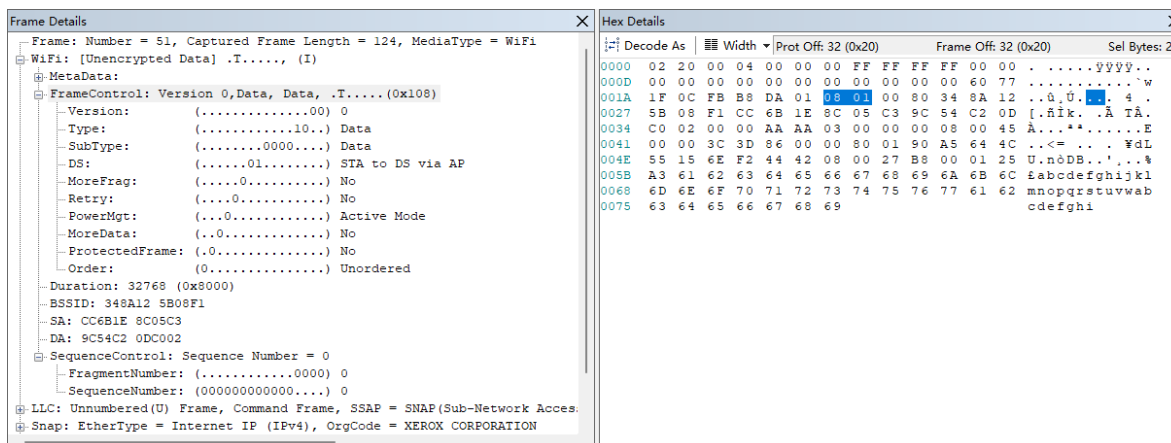
IPv4: Src = 100.76.85.21, Dest = 110.242.68.66, Next Protocol = ICMP, Packet ID = 157

ICMP: Echo Request Message, From 100.76.85.21 To 110.242.68.66

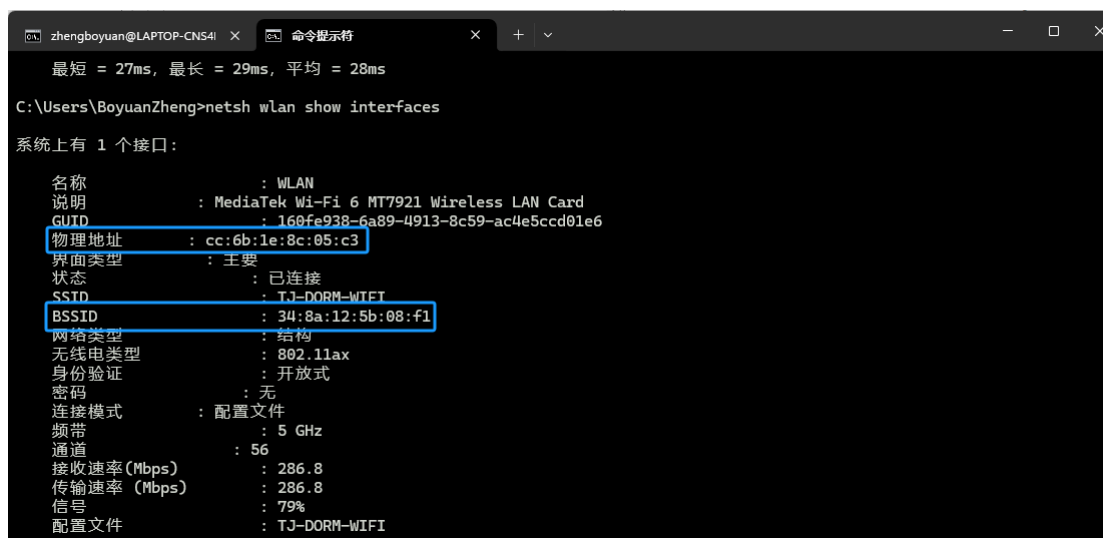
Hex Details

Offset	Hex	ASCII
0000	02 20 00 04 00 00 00 FF FF FF FF 00 00 00 00 00 009999....
0010	00 00 00 00 00 00 00 00 60 77 1F 0C FB B8 DA 01'w..u..
0020	08 01 00 80 34 8A 12 5B 08 F1 CC 6B 1E 8C 05 C3	...4...[.Aik..
0030	9C 54 C2 0D C0 02 00 00 AA AA 03 00 00 00 00 00	Tk..*****
0040	45 00 00 3C 3D 86 00 00 80 01 90 A5 64 4C 55 15	E.. ..WdLU
0050	6E F2 44 42 08 00 27 B8 00 01 25 A3 61 62 63 64	n0DB...'..%abod
0060	65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74	efghijklmnopqrst
0070	75 76 77 61 62 63 64 65 66 67 68 69	uvwxyzefghi

4) 帧格式分析:



上图左侧详细分析了 FC 字段的内容，具体如下：协议版本为 **0**，类型为数据 (**10**)，子类型仍是数据 (**0000**)。ToDS 和 FromDS 为 **01**，表示从站点 (STA，就是本电脑) 发往分布式系统 (DS，就是 WLAN)。更多片段、重试、电源管理、更多数据、更多数据、顺序字段均为 **0**，不存在这些字段对应含义。



SA 地址对应本主机的 MAC 地址 (**cc:6b:1e:8c:05:c3**)，BSSID 对应无线接入点的 MAC 地址 (**34:8a:12:5b:08:f1**)，都可以通过 “netsh wlan show interfaces” 的得到验证。DA 对应目的主机的 MAC 地址 (**9c:54:c2:0d:c0:02**)。

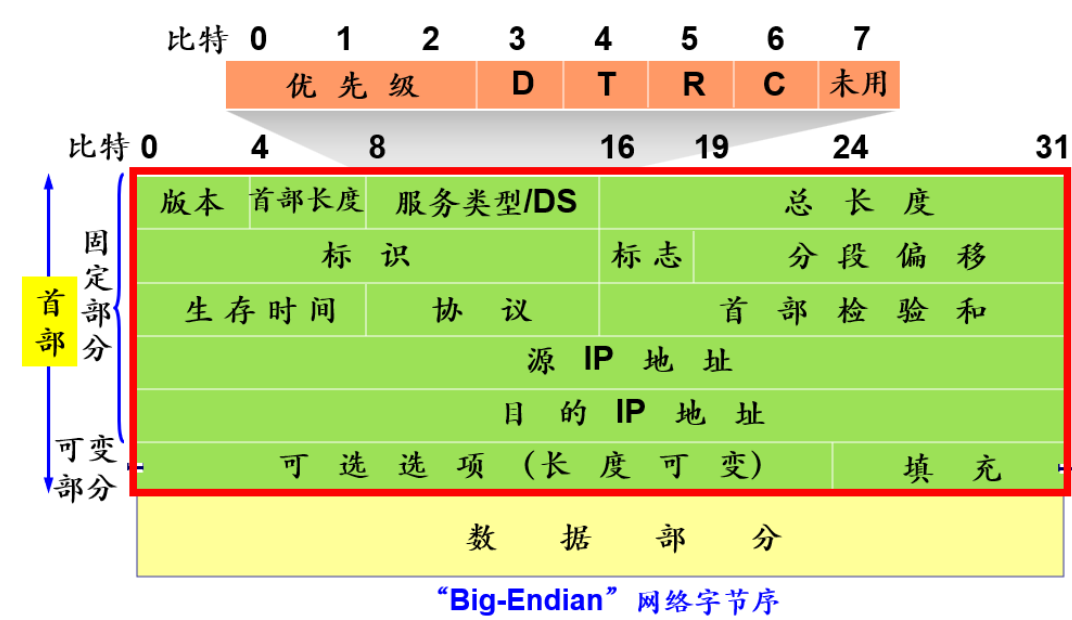
ID 字段表示要占用信道 32768 微秒 (**0x8000**)。SC 字段的分片号和序列号均为 0。同样地，FCS 字段已经被摘除。

三、IP 协议。

1. IP 协议帧格式简介：

网际协议（Internet Protocol, IP），又称互联网协议，是用于分组交换数据网络的协议。IP 是在 TCP/IP 协议族中网络层的主要协议，其任务是根据源主机和目的主机的地址传送数据。

网际协议版本 4（IPv4）是网际协议开发过程中的第四个修订版本，也是此协议第一个被广泛部署和使用的版本。其后继版本为 IPv6，目前仍在部署推进中。本次报告中分析的 IP 协议为 IPv4。



下面分析 IPv4 报文格式：

如上图所示，IPv4 报文分为固定首部（20B）和扩展首部（0~40B）。其中固定首部含 13 个字段，扩展首部包含长度可变的可选选项。首部中的字段均以大端序（Big-Endian）包装。具体字段含义如下：

- 版本： 4 bit。对于 IPv4，该字段值为 4。通信双方使用的版本必须一致；
- 首部长度（IHL）： 4 bit。该字段表示首部长度，其单位是 4 字节。由于 IPv4 首部可能包含数目不定的选项，这个字段也用来确定数据的偏移量。该字

段取值范围为 5~15，对应首部长度 20~60 字节；

- 服务类型/DS: 8 bit。IETF 已经改变了本字段的名称和内容。以前本字段称为服务类型，现在本字段称为区分服务 (DS)；

- 总长度 (Total Length): 16 bit。定义包含首部和数据的报文总长，单位为字节。这个字段的最小值是 20 (20 字节首部 + 0 字节数据)，最大值是 $2^{16} - 1 = 65535$ (字节)。IP 协议规定所有主机都必须支持最小 576 字节的报文 (假定上层数据长度 512 字节+最长 IP 首部 60 字节+4 字节)。当下层数据链路协议的最大传输单元 (MTU) 字段值小于 IP 报文长度时，报文会被分片；

- 标识: 16 bit。用于标识不同的 IP 数据报，同一 IP 数据报的所有分段具有相同的标识；

- 标志: 3 bit。用于控制和识别分段，具体如下：

- 第 1 bit: 保留，必须为 0；

- 第 2 bit: DF (Don't Fragment) 字段: 1 不允许分段，0 允许分段；

- 第 3 bit: MF (More Fragment) 字段: 1 表示后面还有分段，0 表示是最后一个分段。

- 分段偏移: 13bit，表示本分段在 IP 数据报中的相对位置 (度量单位为 8B)，即相对于用户数据字段的起点。

- 生存时间 (TTL): 8 bit。避免报文在互联网中永远存在 (例如，陷入路由环路)。报文经过的每个路由器都将此字段减 1；若路由器发现该字段等于 0，则报文被丢弃，不再向下一跳传送；

- 协议: 8 bit。定义使用 IP 服务的高层协议，以便目的主机的网络层上交数据。例如: UDP (17)、TCP (6)、ICMP (1) 等；

- 首部检验和: 16 bit。不采用 CRC 码而采用简单的计算方法。只对首部查错，不包括数据部分。在每一跳，路由器都要重新计算出的首部检验和并与此字段进行比对，若不一致则丢弃报文；

- 源地址：IPv4 地址由四个字节共 32 位构成，此字段的值是将每个字节的二进制合在一起所得的 32 位值。由于目前 IPv4 地址较为紧缺，源地址未必是源主机的 IP 地址，也可能是经由网络地址转换的结果；

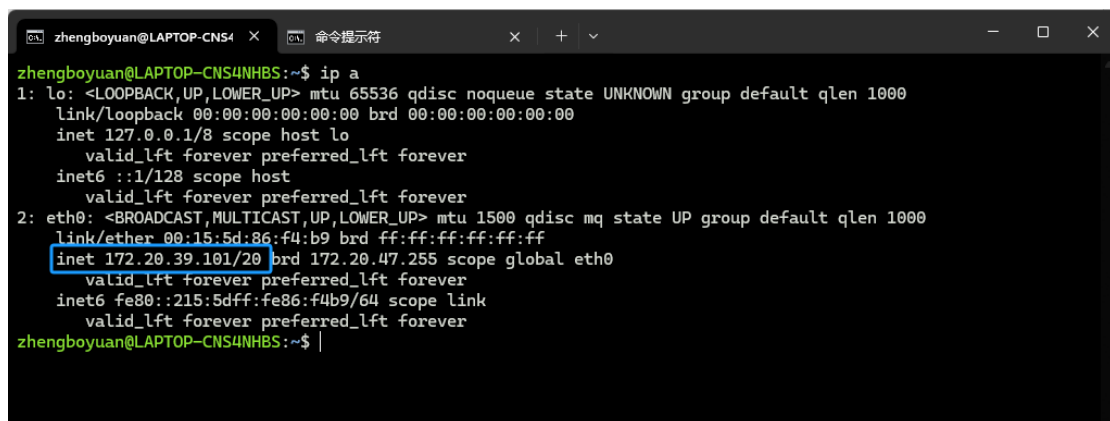
- 目的地址：与源地址格式相同，指出报文的接收端；

- 扩展首部：扩展首部是选项字段，长度可变，从 1B 到 40B 不等，取决于所选择的项目。最初设计时定义了 5 个选项：安全性、严格源路由、松散源路由、记录路由、时间戳。实际上这些选项很少被使用。

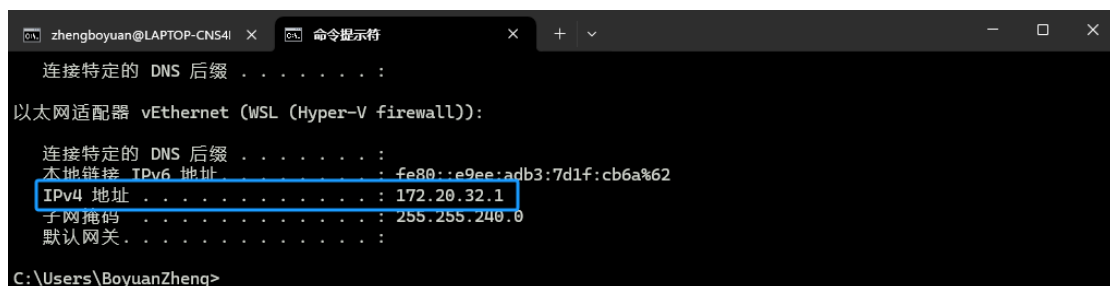
2. 捕获 IP 协议帧实验步骤：

本次捕获可以沿用捕获以太网 MAC 帧的实验，观察 IPv4 报头。

1) 观察 Windows 系统和 WSL2 对应的 IP 地址：



```
zhengboyuan@LAPTOP-CNS4:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:86:f4:b9 brd ff:ff:ff:ff:ff:ff
    inet 172.20.39.101/20 brd 172.20.47.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe86:f4b9/64 scope link
        valid_lft forever preferred_lft forever
zhengboyuan@LAPTOP-CNS4:~$
```



```
连接特定的 DNS 后缀 . . . . . :
以太网适配器 vEthernet (WSL (Hyper-V firewall)):

    连接特定的 DNS 后缀 . . . . . :
    本地链接 IPv6 地址 . . . . . : fe80::e9ee:adb3:7d1f:cb6a%62
    IPv4 地址 . . . . . : 172.20.32.1
    子网掩码 . . . . . : 255.255.240.0
    默认网关 . . . . . :

C:\Users\BoyuanZheng>
```

2) 在 Windows 系统通过终端 ping WSL2：

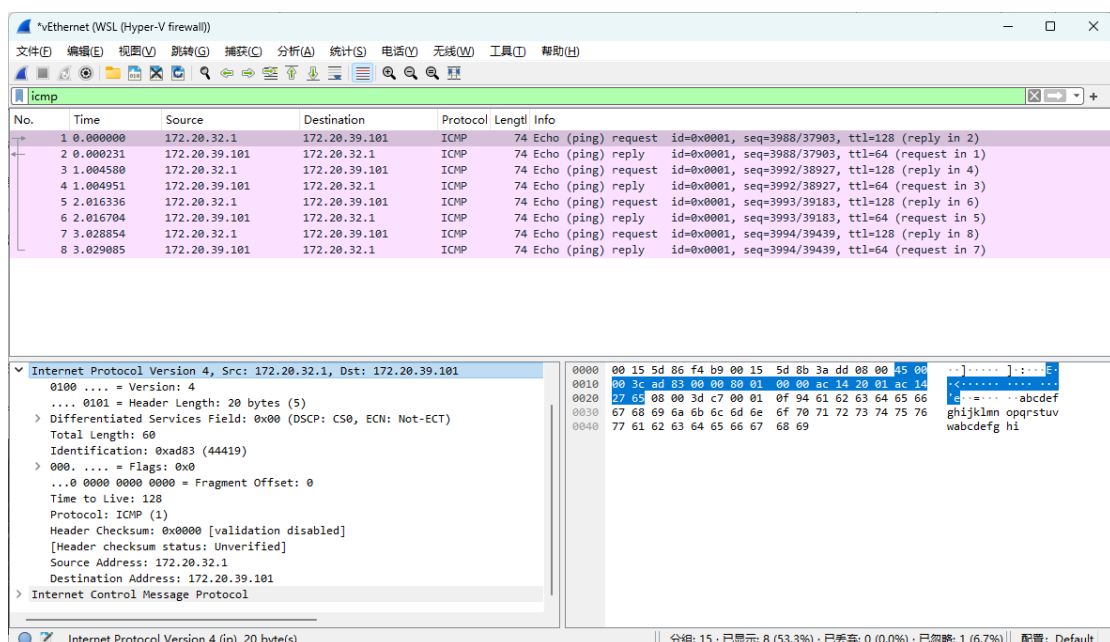
```
zhengboyuan@LAPTOP-CNS4I 命令提示符
C:\Users\BoyuanZheng>ping 172.20.39.101

正在 Ping 172.20.39.101 具有 32 字节的数据:
来自 172.20.39.101 的回复: 字节=32 时间<1ms TTL=64
来自 172.20.39.101 的回复: 字节=32 时间<1ms TTL=64
来自 172.20.39.101 的回复: 字节=32 时间<1ms TTL=64
来自 172.20.39.101 的回复: 字节=32 时间<1ms TTL=64

172.20.39.101 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 0ms, 最长 = 0ms, 平均 = 0ms

C:\Users\BoyuanZheng>
```

3) 在 wireshark 中捕获数据包:



4) 帧格式分析:

上图左侧详细分析了首部各字段的含义与内容，具体如下:

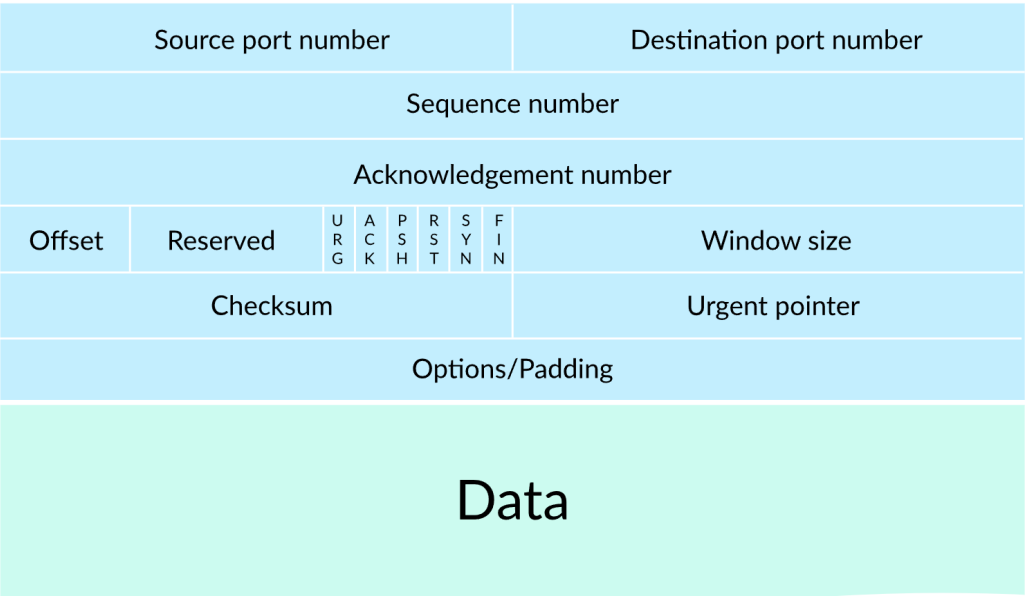
- 协议版本为 IPv4 (*0b0100*);
- 首部长度的 20 字节 (单位 4 字节, *0x0101*), 不含扩展部分;
- **DS 部分** (*0x00*) 表示差分服务代码点 (DSCP) 为 CS0, 无显示拥塞通告 (ECN);
- 数据包总长度为 60 字节 (单位为字节, *0x003c*);
- 标识该 IP 数据报的编号为 *0xad83*;

- 标志位 (0b000) 表示允许分段且当前为最后一个分段;
- 分段偏移 (13'b0) 为 0, 因为本身这个数据报没有被分段;
- 生存时间表示还可以在路由器间进行 128 跳 (0x80);
- 协议字段表示使用的高层协议是 ICMP (0x01), 对应 ping 所使用的协议;
- 首部校验和字段表示不校验 (0x0000);
- 源地址 172.20.32.1 (0xac142001) 与 Windows 系统中显示的一致, 这是一次 request;
- 目标地址 172.20.39.101 (0xac142765) 与 WSL2 中显示的一致;
- 无扩展首部, 由总长度字段也可知。

四、TCP 协议。

1. TCP 协议帧格式简介：

传输控制协议（TCP）是一种面向连接的、可靠的、基于字节流的传输层通信协议。TCP 层位于 IP 层之上、应用层之下，提供可靠连接。在简化的计算机网络 OSI 模型中，它完成第四层传输层所指定的功能。



下面分析 TCP 协议帧格式：

- 源端口：16 bit，表示发送端口；
- 目的地端口：16 bit，表示接收端口；
- 序列号：32 bit，具体含义如下：
 - 若 SYN 标志置 1，则是初始序列号。实际的第一个数据字节的序列号和相应 ACK 中的确认号是这个序列号加 1；
 - 若 SYN 标志置 0，则是当前会话中该段第一个数据字节的累积序列号。
- 确认号：32 bit。若 ACK 标志置 1，则该字段的值是 ACK 发送者所期望的下一个帧序列号；
- 数据偏移：4 bit，表示 TCP 头的大小（即数据段的偏移），单位是 4 字节。与

IP 协议类似，该字段取值范围为 5~15，对应首部长度 20~60 字节。

- 保留字段：3 bit；

- 标志：9 bit，具体如下：

- NS: ECN-nonce，隐蔽性保护；

- CWR: 拥塞窗口减少。由发送主机设置，表明它收到了设置了 ECE 标志的 TCP 段，并已在拥塞控制机制中作出反应；

- ECE: ECN-Echo 有双重作用：

- 若 SYN 标志置 1，说明 TCP 对等体有 ECN 能力；

- 若 SYN 标志置 0，说明在正常传输过程中收到了 IP 头中设置有拥塞经验标志（ECN=11）的数据包。

- URG: 表明紧急指针字段重要；

- ACK: 表示确认字段重要。客户端发送的初始 SYN 数据包之后的所有数据包都应设置这个标志；

- PSH: 推送功能，要求将缓冲的数据推送给接收的应用程序；

- RST: 表示重置连接；

- SYN: 同步序列号。只有从两端发送的第一个数据包设置该标志；

- FIN: 表明是来自发送方的最后一个数据包。

- 窗口大小：16 bit，即接收窗口的大小，表明发送方当前愿意接收的窗口大小单位的数量。

- 校验和：16bit，用于 TCP 头、有效载荷和 IP 伪头的错误检查；

- 紧急指针：16 bit。若 URG 标志置 1，则该字段是序列号的偏移，表示最后一个紧急数据字节。

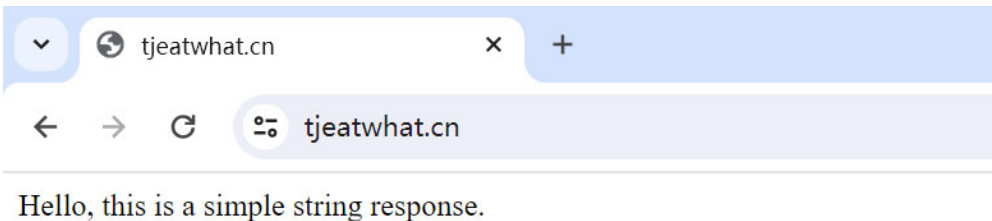
- 选项：0~320 bit（32 整数倍）。该字段长度由数据偏移字段决定，最多有三个

字段：Option-Kind（1B），Option-Length（1B），Option-Data（可变）。

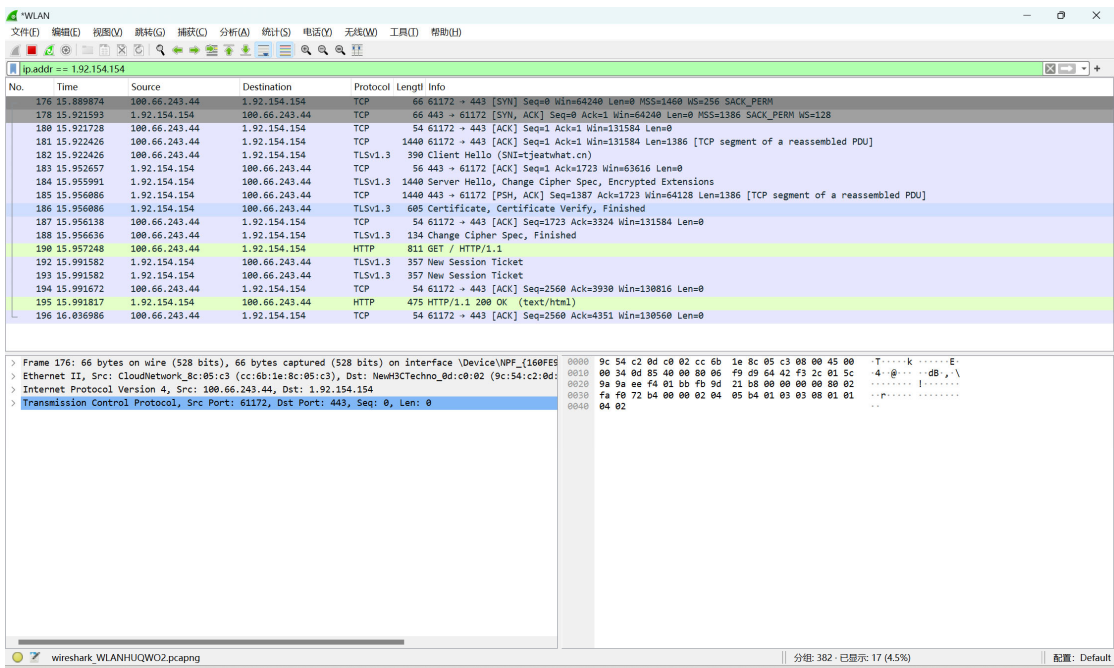
3. 捕获 TCP 协议帧实验步骤：

IP 地址为 1.92.154.154 的服务器是本学期我为软件工程课程所搭建的。访问 tjeatwhat.cn 可以得到一个字符串的简单响应。本部分实验在此基础上展开：

1) 通过浏览器访问 tjeatwhat.cn：

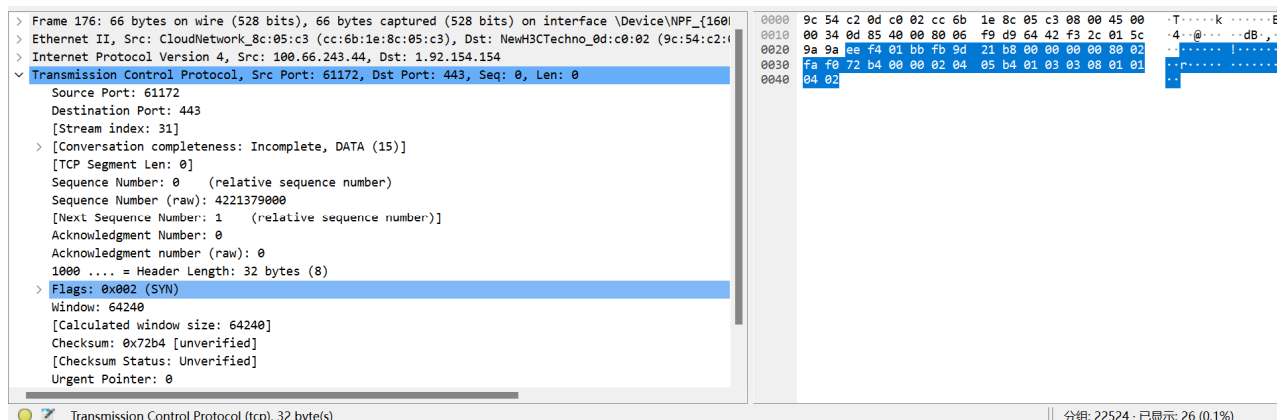


2) 监听 WLAN，将 filter 设置为 ip == 1.92.154.154：



可以观察到 3 次 TCP 的握手过程（即本机先发 SYN，服务端目标主机回复 SYN ACK，本机回复 ACK 建立连接）。

3) 以第一个 SYN 帧为例分析 TCP 帧格式：



注意蓝色部分是 TCP 协议的首部，此前是之前分析的 IP 协议等的首部。

上图左侧详细分析了首部各字段的含义与内容，具体如下：

- 源端口为 61172 (*0xeeef4*);
- 目标端口为 443 (*0x01bb*);
- 序列号 (*0x0159a9a*) 表示真实序列号为 4221379000 (初始序列号是随机值)，wireshark 显示了计算后的相对序列号为 0;
- 确认号 (*0x000000*), 该字段无意义，因为此后 ACK 字段置 0;
- 数据偏移表示首部长度的为 4×8 (*0x8*) = 32 字节;
- 保留字段 (*0b000*);
- 标志 (*0b000000010*) 字段中除 SYN 字段其他均置零。SYN 表示是发送的第一个数据包，其他如 ACK 置 0 表示不进行确认;
- 窗口大小为 64240 (*0xfaf0*);
- 校验和为 *0x72b4*;
- 紧急指针为 0 (*0x0000*);
- 可选部分共 32 bit，包含最大段长度等 TCP Option 信息。

五、HTTP 协议。

1) HTTP 协议帧格式简介:

超文本传输协议 (HTTP) 是一种应用层协议, 是万维网的数据通信的基础。HTTP 是客户端和服务端之间请求与应答的标准。HTTP 报文是以文本形式而非二进制形式传输, 其中的换行为 CRLF (0x0D 0x0A, \r\n), 与 Windows 文本文件的换行格式相同, 而与 Unix 中文本文件的格式不同。

对于请求信息, 有如下格式:

1. 请求行 (Request Line): 包括 HTTP 方法 (如 GET、POST)、请求的资源路径 (URL) 和 HTTP 版本。

格式: Method /path HTTP/version

2. 请求头 (Request Headers): 包含一系列的键值对, 用于提供更多请求信息, 如用户代理、接受的媒体类型等。

格式: Header-Name: Header-Value

3. 空行: 请求头和请求体之间用一个空行分隔, 表示请求头结束;

4. 请求体 (Request Body): 可选部分, 用于 POST 或 PUT 请求中发送数据给服务器。

对于响应信息, 有如下格式:

1. 响应行 (Status Line): 服务器响应时的第一部分, 包含 HTTP 版本、状态码和状态信息。

格式: HTTP/version Status-Code Status-Message

2. 响应头 (Response Headers): 类似于请求头, 包含服务器返回的额外信息, 如内容类型、内容长度等。

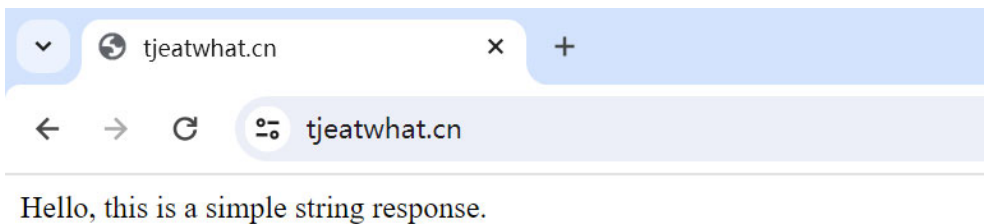
3. 空行: 响应头和响应体之间的分隔。

4. 响应体 (Response Body): 服务器返回的数据, 如 HTML 文档等。

2) 捕获 HTTP 协议帧实验步骤:

IP 地址为 1.92.154.154 的服务器是本学期我为软件工程课程所搭建的。访问 tjeatwhat.cn 可以得到一个字符串的简单响应。本部分实验在此基础上展开:

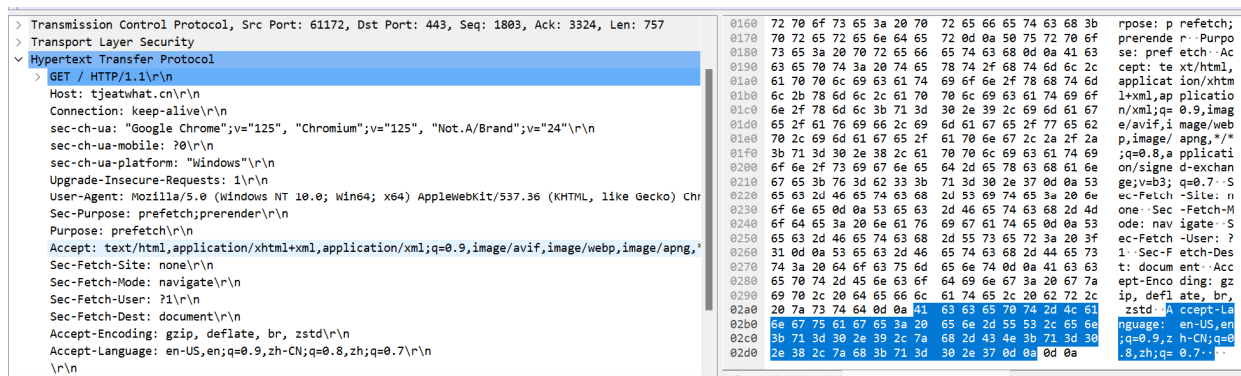
1) 通过浏览器访问 tjeatwhat.cn:



2) 监听 WLAN, 将 filter 设置为 ip == 1.92.154.154:

The screenshot shows a Wireshark capture of network traffic on a WLAN interface. The filter is set to 'ip.addr == 1.92.154.154'. The packet list shows several TCP and TLS segments, followed by an HTTP GET request (packet 190) and its response (packet 191). The packet details pane for packet 191 shows the HTTP response structure, including the status line '200 OK (text/html)' and various headers like 'Content-Type: text/html; charset=utf-8' and 'Content-Length: 26'. The packet bytes pane shows the raw data of the response, which is the string 'Hello, this is a simple string response.'.

3) 查看并分析 GET 请求信息:



```
GET / HTTP/1.1\r\n
Host: tjeatwhat.cn\r\n
Connection: keep-alive\r\n
sec-ch-ua: "Google Chrome";v="125", "Chromium";v="125", "Not.A/Brand";v="24"\r\n
sec-ch-ua-mobile: ?0\r\n
sec-ch-ua-platform: "Windows"\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36\r\n
Sec-Purpose: prefetch; prerender\r\n
Purpose: prefetch\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7\r\n
Sec-Fetch-Site: none\r\n
Sec-Fetch-Mode: navigate\r\n
Sec-Fetch-User: ?1\r\n
Sec-Fetch-Dest: document\r\n
Accept-Encoding: gzip, deflate, br, zstd\r\n
Accept-Language: en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7\r\n
\r\n
```

1. 请求行:

GET / HTTP/1.1 表示这是一个 HTTP GET 请求, 目标是服务器的根路径/, 使用的是 HTTP/1.1 协议版本。

2. 请求头:

- Host: tjeatwhat.cn 指定了请求的服务器域名。
- Connection: keep-alive 表示客户端希望与服务器保持连接, 以便可以发送多个请求, 减少连接建立和关闭的开销。
- sec-ch-ua 包含了客户端浏览器的用户代理字符串, 有助于服务器识别请求来自的浏览器和版本。
- sec-ch-ua-mobile 表示请求是否来自移动设备, ?0 表示不是。
- sec-ch-ua-platform 表明请求来自 Windows 平台。
- Upgrade-Insecure-Requests: 1, 表示请求服务器升级所有 HTTP 请求到 HTTPS, 增强安全性。
- User-Agent 提供了详细的客户端浏览器信息, 包括浏览器类型、操作系统和版本。
- Sec-Purpose 和 Purpose 表明请求可能是用于预加载或预渲染, 有助于服务器优化资源加载。
- Accept 指定了客户端能够接受的媒体类型。
- Sec-Fetch-Site、Sec-Fetch-Mode、Sec-Fetch-User、Sec-Fetch-Dest 等字段提供了额外的请求上下文, 用于内容安全策略和请求的同源策略检查。
- Accept-Encoding 列出了客户端支持的压缩算法, 服务器可以根据这些信息选择合适的压缩方式。
- Accept-Language 表明客户端语言偏好, 服务器可据此返回本地化内容。

3. 空行:

两个\r\n 字符表示请求头结束, 之后如果是 POST 或 PUT 请求, 这里会跟着请求体; 但在这个 GET 请求中, 请求体被省略。

4. 请求体 (空)

4) 查看并分析响应信息:

> Transmission Control Protocol, Src Port: 443, Dst Port: 61172, Seq: 3930, Ack: 2560, Len: 421

> Transport Layer Security

> Hypertext Transfer Protocol, has 2 chunks (including last chunk)

> HTTP/1.1 200 OK\r\n

Server: nginx/1.18.0 (Ubuntu)\r\n

Date: Sat, 08 Jun 2024 07:11:03 GMT\r\n

Content-Type: text/html; charset=utf-8\r\n

Transfer-Encoding: chunked\r\n

Connection: keep-alive\r\n

X-Frame-Options: DENY\r\n

X-Content-Type-Options: nosniff\r\n

Referrer-Policy: same-origin\r\n

Cross-Origin-Opener-Policy: same-origin\r\n

Content-Encoding: gzip\r\n

\r\n

[HTTP response 1/1]

[Time since request: 0.034569000 seconds]

[Request in frame: 190]

[Request URI: https://tjeatwhat.cn/]

> HTTP chunked response

Content-encoded entity body (gzip): 58 bytes -> 40 bytes

0000 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK\r\n

0010 0a 53 65 72 76 65 72 3a 20 6e 67 69 6e 78 2f 31 .Server: nginx/1

0020 2e 31 38 2e 30 20 28 55 62 75 6e 74 75 29 0d 0a .,18.0 (Ubuntu)\r\n

0030 44 61 74 65 3a 20 53 61 74 2c 20 30 38 20 4a 75 Date: Sat, 08 Jun

0040 6e 20 32 30 32 34 20 30 37 3a 31 31 3a 30 33 20 n 2024 07:11:03

0050 47 4d 54 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 GMT-Content-Type

0060 65 3a 20 74 65 78 74 2f 68 74 6d 6c 3b 20 63 68 e: text/html; ch

0070 61 72 73 65 74 3d 75 74 66 2d 38 0d 0a 54 72 61 arset=utf-8; Tra

0080 6e 73 66 65 72 2d 45 6e 63 6f 64 69 6e 67 3a 20 nsfer-En coding:

0090 63 68 75 6e 6b 65 64 0d 0a 43 6f 6e 6e 65 63 74 chunked; Connect

00a0 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d ion: keep-alive

00b0 0a 58 2d 46 72 61 6d 65 2d 4f 70 74 69 6f 6e 73 .X-Frame-Options

00c0 3a 20 44 45 46 59 0d 0a 58 2d 43 6f 6e 74 65 6e : DENY; X-Conten

00d0 74 2d 54 79 70 65 2d 4f 70 74 69 6f 6e 73 3a 20 t-Type-o ptions:

00e0 6e 6f 73 6e 69 66 66 0d 0a 52 65 66 65 72 72 65 nosniff; Referen

00f0 72 2d 50 6f 6e 69 63 79 3a 20 73 61 6d 65 2d 6f r-Policy : same-o

0100 72 69 67 69 6e 0d 0a 43 72 6f 73 73 2d 4f 72 69 rigin-C ross-Ori

0110 67 69 6e 2d 4f 70 65 6e 65 72 2d 50 6f 6e 69 63 gin-Open er-Polic

0120 79 3a 20 73 61 6d 65 2d 6f 72 69 67 69 6e 0d 0a y: same- origin

0130 43 6f 6e 74 65 6e 74 2d 45 6e 63 6f 64 69 6e 67 Content- Encoding

0140 3a 20 67 7a 69 70 0d 0a 0d 0a 33 61 0d 0a 1f 8b ; gzip; Ba...

0150 08 00 00 00 00 00 04 03 f3 48 cd c9 c9 d7 51 28H.....Q(

0160 c9 c8 2c 56 00 a2 44 85 e2 cc dc 82 9c 54 85 e2 ..V.D.....T=

0170 92 a2 cc bc 74 85 a2 d4 e2 82 fc bc e2 54 3d 00t....T=

Frame (475 bytes)

Decrypted TLS (399 bytes)

De-chunked entity body (58 bytes)

Hypertext Transfer Protocol (http), 330 byte(s)

分组: 98620 · 已显示: 26 (0.0%)

```
HTTP/1.1 200 OK\r\n
Server: nginx/1.18.0 (Ubuntu)\r\n
Date: Sat, 08 Jun 2024 07:11:03 GMT\r\n
Content-Type: text/html; charset=utf-8\r\n
Transfer-Encoding: chunked\r\n
Connection: keep-alive\r\n
X-Frame-Options: DENY\r\n
X-Content-Type-Options: nosniff\r\n
Referrer-Policy: same-origin\r\n
Cross-Origin-Opener-Policy: same-origin\r\n
Content-Encoding: gzip\r\n
\r\n
```

1. 响应行:

HTTP/1.1 200 OK 表示 HTTP 请求成功, 状态码 200 表示请求已成功处理。

2. 响应头:

- **Server: nginx/1.18.0 (Ubuntu)** 显示了服务器的软件和版本信息, 此处表示服务器是 nginx, 版本为 1.18.0。
- **Date: Sat, 08 Jun 2024 07:11:03 GMT** 响应生成的日期和时间。
- **Content-Type: text/html; charset=utf-8** 表明响应的内容类型是 HTML, 字符编码是 UTF-8。

- Transfer-Encoding: chunked 表示响应体使用"chunked"编码方式，这意味着响应体被分成多个块发送。
- Connection: keep-alive 表示服务器希望保持连接，以便可以发送多个响应。
- X-Frame-Options: DENY 表示不允许任何页面通过 frame 或 iframe 嵌入此响应。
- X-Content-Type-Options: nosniff 防止浏览器尝试猜测响应的类型。
- Referrer-Policy: same-origin 限制 Referer 头的发送。
- Cross-Origin-Opener-Policy: same-origin 限制跨域的窗口交互。
- Content-Encoding: gzip 表示响应体被 gzip 压缩。

3. 空行:

两个\r\n 字符表示响应头结束，之后是响应体。

4. 响应体:

原始响应体大小为 58 字节，经过 gzip 压缩后为 40 字节。

0000	48 65 6c 6c 6f 2c 20 74	68 69 73 20 69 73 20 61	Hello, t his is a
0010	20 73 69 6d 70 6c 65 20	73 74 72 69 6e 67 20 72	simple string r
0020	65 73 70 6f 6e 73 65 2e		esponse.