

# 华东师范大学软件工程学院实验报告

实验课程:	计算机网络	年 级:	2023 级
实验编号:	Lab 01	实验名称:	Protocol Layer
姓 名:	王海生	学 号:	10235101559

## 1 实验目的

- 1) 学会通过 Wireshark 获取各协议层的数据包
- 2) 掌握协议层数据包结构
- 3) 分析协议开销

## 2 实验内容与实验步骤

### 2.1 实验内容

#### 2.1.1 获取协议层的数据包

使用 `wget` 命令发起 HTTP 请求，然后使用 Wireshark 抓包。

#### 2.1.2 绘制数据包结构

分析 HTTP GET 协议包的内容，并绘制协议包，分别标出 Ethernet, IP 和 TCP 协议的头部的位置、大小以及其负载的范围。

#### 2.1.3 分析协议开销

根据实验结构，分析 HTTP 应用协议额外开销。估计上面捕获的 HTTP 协议的额外开销。假设 HTTP 数据（头部和消息）是有用的，而 TCP, IP 和 Ethernet 头部认为是开销。对于下载的主要部分中的每一个包，我们需要分析 Ethernet, IP 和 TCP 的开销，和有用的 HTTP 数据的开销。根据以上的定义来估计下载协议的开销，你认为这种开销是必要的吗？

#### 2.1.4 分析解复用键

解复用指找到正确的上一层协议来处理到达的包。

观察下载的以太网和 IP 包头部信息，回答下面问题：

1. 以太网头部中哪一部分是解复用（解复用：找到正确的上一层协议来处理到达的包的行为叫做解复用）键并且告知它的下一个高层指的是 IP，在这一包内哪一个值可以表示 IP？

2. IP 头部中哪一部分是解复用键并且告知它的下一个高层指的是 TCP, 在这一包内哪一个值可以表示 TCP?

### 2.1.5 问题讨论

1. 查看不包含高层数据的短 TCP 数据包, 查看它发往哪? 不携带高层数据的数据包有用吗?
2. 在经典的分层模型中, 低层字段包装到高层数据包外面, 成为一条新消息。但这并非总是如此, Web 响应 (一个包含 HTTP 标头和 HTTP 有效负载的 HTTP 消息) 可能被转换为多个较低层的消息 (即多个 TCP 数据包)。假设你为 Web 响应的第一个和最后一个 TCP 数据包绘制了数据包结构, 那么该结构与经典分层模型有什么不同?
3. 在上述经典分层模型中, 低层字段包装到高层数据包外面, 如果较低层添加加密, 此模型将如何更改?
4. 在上述经典分层模型中, 低层字段包装到高层数据包外面, 如果较低的层添加压缩, 此模型将如何更改?

## 2.2 实验步骤

- 1) 安装实验所需软件 (为方便安装, 我们使用 Windows 下的包管理器 winget)

```
1 PS> winget install wget
2 PS> winget install wireshark
```

- 2) 打开 Wireshark, 在菜单栏的 捕获 -> 选项中进行设置, 选择已连接的网络, 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。
- 3) 使用 wget 命令发起 HTTP 请求

```
1 PS> wget http://www.baidu.com
```

- 4) 在 Wireshark 中停止捕获。
- 5) 分析 HTTP GET 协议包的内容, 并绘制协议包。
- 6) 分析协议开销
- 7) 分析解复用键
- 8) 问题讨论

## 3 实验环境

- 操作系统: Windows 11 家庭中文版 23H2 22631.2715
- 网络适配器: Killer(R) Wi-Fi 6 AX1650i 160MHz Wireless Network Adapter(201NGW)
- Wireshark: Version 4.2.0 (v4.2.0-0-g54eedfc63953)
- wget: GNU Wget 1.21.4 built on mingw32

## 4 实验过程与分析

### 4.1 获取协议层的数据包

问题：根据抓取的 HTTP 请求的 GET 方法的抓取结果，分析协议包的内容。

首先，我们打开 Wireshark，在菜单栏的 捕获 -> 选项中进行设置，选择已连接的网络，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。

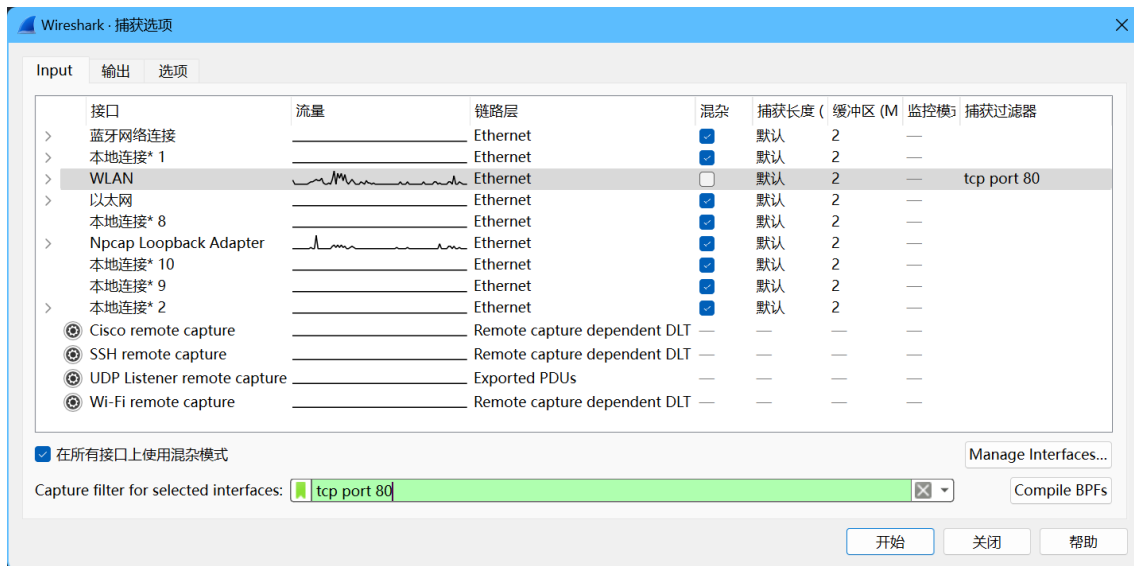


图 1: 开始捕获

然后，我们使用 wget 命令对 http://www.baidu.com 发起 HTTP 请求

```
lipen@lipen:~$ wget http://www.baidu.com
--2023-11-30 19:54:31-- http://www.baidu.com/
Connecting to 127.0.0.1:7890 ... connected.
Proxy request sent, awaiting response... 200 OK
Length: 2381 (2.3K) [text/html]
Saving to: 'index.html'

index.html          100%[=====>]  2.33K  --KB/s   in 0s

2023-11-30 19:54:31 (131 MB/s) - 'index.html' saved [2381/2381]
```

图 2: 使用 wget 命令发起 HTTP 请求

最后，我们在 Wireshark 中停止捕获，得到如下结果：

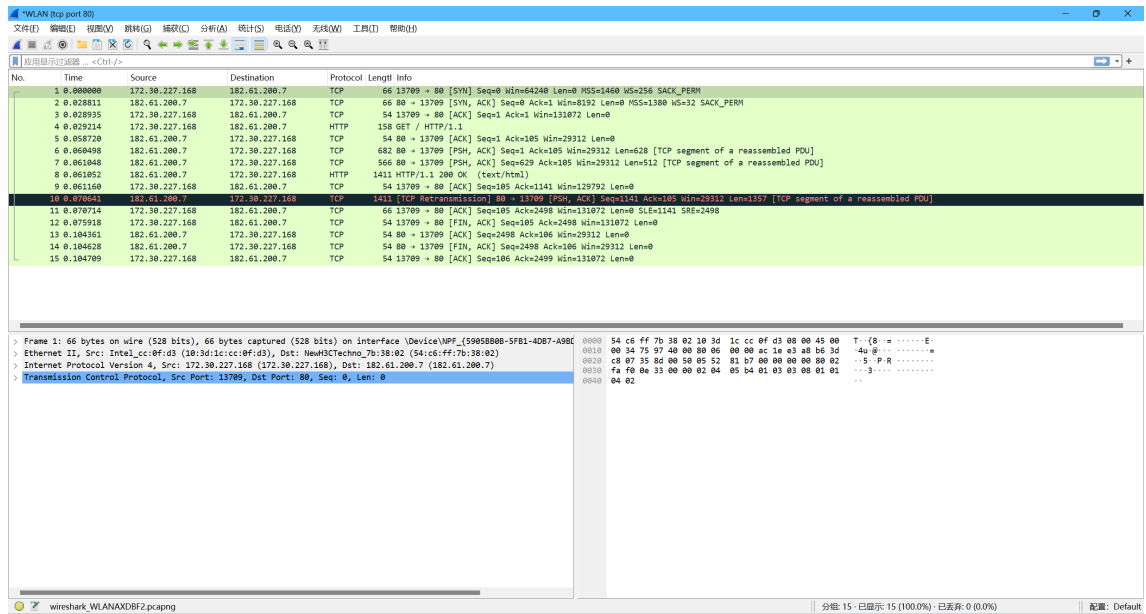


图 3: 捕获结果

## 4.2 绘制数据包结构

问题：画一个关于使用 GET 方法的 HTTP 请求的图 (与下图类似)，为了显示协议层的嵌套结构，请分别标出 Ethernet, IP 和 TCP 协议的头部的位置、大小以及其负载的范围。

接下来，我们分析数据包的结构。在 Wireshark 中，我们选择 HTTP GET 协议包，可以看到其内容，如下图所示：

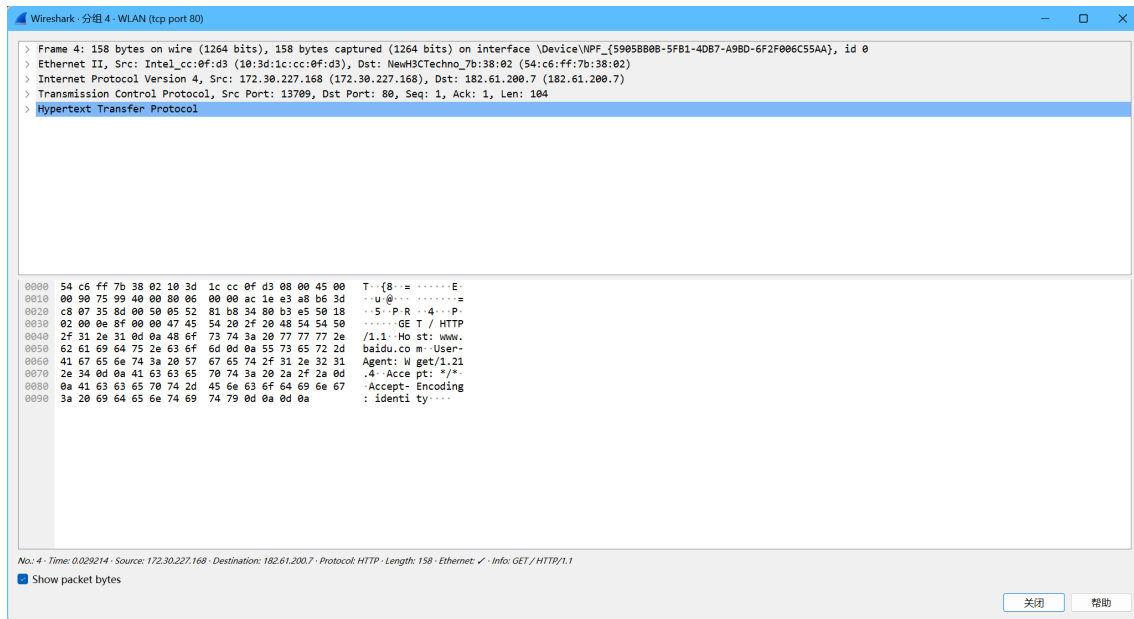


图 4: HTTP GET 协议包

其中，第一个块是 **Frame**，这不是一个协议，而是一个记录，描述有关数据包的整体信息，包括捕获时间和长度等。我们可以看到，整个数据包的长度是 **158 字节**。第二个块是 **Ethernet II**，这是以太网协议，可以看到这个标头的长度是 **14 字节**。第三个块是 **Internet Protocol Version 4**，这是 IP 协议，可以看到这个标头的长度是 **20 字节**。第四个块是 **Transmission Control Protocol**，这是 TCP 协议，可以看到这个标头的长度是 **20 字节**。第五个块是 **Hypertext Transfer Protocol**，这是 HTTP 协议，可以看到这个标头的长度是 **104 字节**。

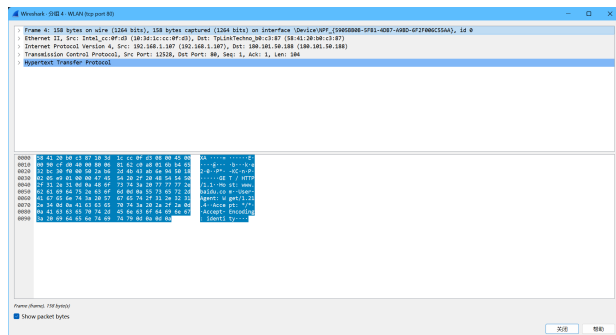


图 5: 数据包的整体信息

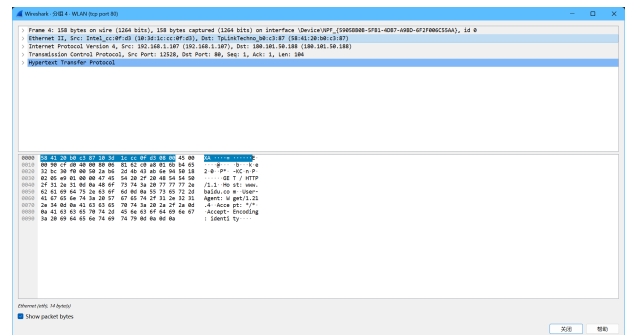


图 6: 以太网协议包

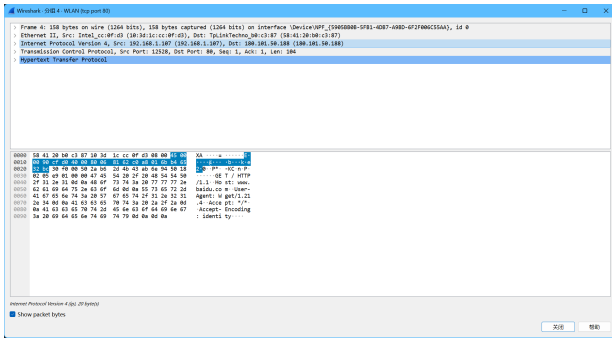


图 7: IP 协议包

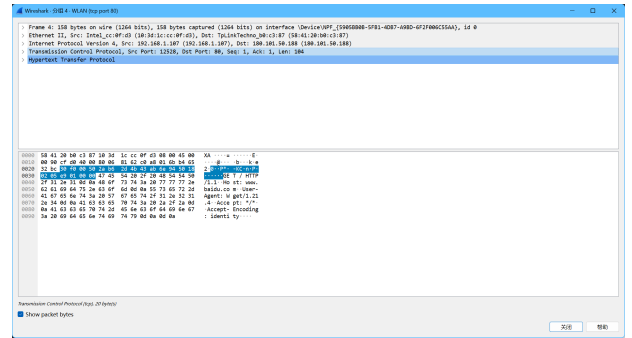


图 8: TCP 协议包

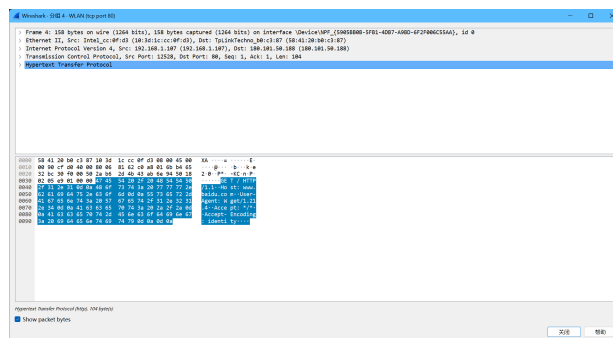


图 9: HTTP 协议包

协议包的结构如下图所示：

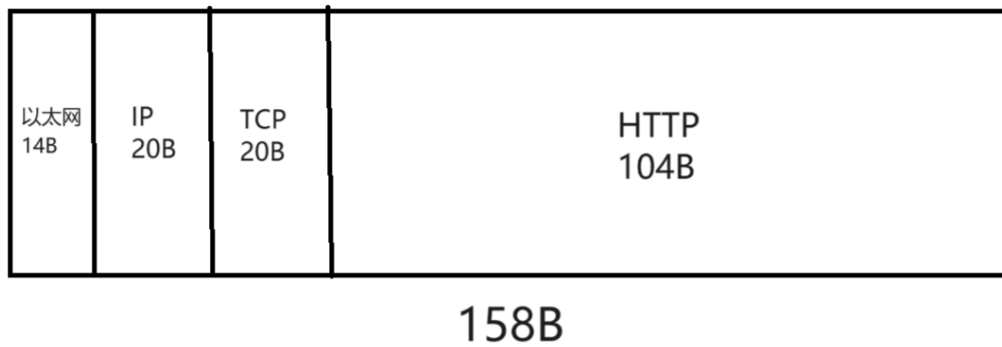


图 10: 协议包的结构

### 4.3 分析协议开销

问题：根据数据包的抓取结果，分析协议开销。

以太网的开销为

$$\frac{14}{158} = 8.86\%$$

IP 的开销为

$$\frac{20}{158} = 12.66\%$$

TCP 的开销为

$$\frac{20}{158} = 12.66\%$$

整个捕获过程（从 SYN ACK 开始，到 HTTP 后的第一个 TCP 数据包结束）的开销为

$$\frac{14 + 20 + 20 + 66 + 54 \times 2}{158 + 66 + 54 \times 2} = 68.7\%$$

有用的 HTTP 数据的开销为

$$\frac{104}{158 + 66 + 54 \times 2} = 31.3\%$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.107	180.101.50.188	TCP	66	12528 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.009812	180.101.50.188	192.168.1.107	TCP	66	80 → 12528 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1440 WS=32 SACK_PERM
3	0.009924	192.168.1.107	180.101.50.188	TCP	54	12528 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
4	0.018554	192.168.1.107	180.101.50.188	HTTP	158	GET / HTTP/1.1
5	0.018934	180.101.50.188	192.168.1.107	TCP	54	80 → 12528 [ACK] Seq=1 Ack=105 Win=78464 Len=0
6	0.019735	180.101.50.188	192.168.1.107	TCP	682	80 → 12528 [PSH, ACK] Seq=1 Ack=105 Win=78464 Len=628 [TCP segment of a reassembled PDU]
7	0.019736	180.101.50.188	192.168.1.107	TCP	566	80 → 12528 [PSH, ACK] Seq=629 Ack=105 Win=78464 Len=512 [TCP segment of a reassembled PDU]
8	0.019738	180.101.50.188	192.168.1.107	TCP	1078	80 → 12528 [PSH, ACK] Seq=1141 Ack=105 Win=78464 Len=1024 [TCP segment of a reassembled PDU]
9	0.019738	180.101.50.188	192.168.1.107	HTTP	387	HTTP/1.1 200 OK (text/html)

图 11: 整个捕获过程

#### 4.4 协议开销分析与必要性讨论

问题：估计协议的开销或者是协议开销占用下载字节的百分比。对于下载的主要部分中的每一个包，我们需要分析 Ethernet, IP 和 TCP 的开销，和有用的 HTTP 数据的开销，你认为这种开销是必要的吗？（假设 HTTP 数据（头部和消息）是有用的，而 TCP, IP 和 Ethernet 头部认为是开销。）

##### 4.4.1 协议开销估计

根据以太网、IP、TCP 等协议的标准文档，我们可以估计协议开销占用下载字节的百分比。具体如下：

**以太网：**以太网帧头部通常包含 14 字节（前导码 1 字节、目标 MAC 地址 6 字节、源 MAC 地址 6 字节、类型 2 字节），尾部包含 4 字节的 FCS（帧校验序列）。因此，以太网的开销为：

$$\frac{14 + 4}{158} = \frac{18}{158} = 11.39\%$$

**IP：**IPv4 头部通常为 20 字节（最小值），因此 IP 的开销为：

$$\frac{20}{158} = 12.66\%$$

**TCP:** TCP 头部通常为 20 字节（最小值），因此 TCP 的开销为：

$$\frac{20}{158} = 12.66\%$$

综合来看，整个捕获过程中（从 SYN ACK 开始，到 HTTP 后的第一个 TCP 数据包结束），包括以太网、IP、TCP 头部及后续的握手过程，总开销为：

$$\frac{18 + 20 + 20 + 66 + 54 \times 2}{158 + 66 + 54 \times 2} = \frac{242}{386} = 62.69\%$$

有用的数据，即 HTTP 数据的开销为：

$$\frac{104}{158 + 66 + 54 \times 2} = \frac{104}{386} = 26.94\%$$

#### 4.4.2 协议开销的必要性

对于下载的主要部分中的每一个包，以太网、IP 和 TCP 的头部信息确实被视为开销。然而，这些开销是确保数据能够正确无误地从发送方传输到接收方所必需的。具体来说：

- 1) 以太网头部包含了 MAC 地址等信息，用于局域网内的设备识别和寻址。
- 2) IP 头部则提供了互联网级别的路由信息，确保数据包可以跨多个网络进行传递。
- 3) TCP 头部负责建立连接、确认机制、流量控制等功能，保证了数据传输的可靠性。

因此，虽然这些头部信息占用了部分传输容量，但它们的存在是实现有效、可靠通信的基础。没有这些协议头部提供的功能，数据传输将无法正常工作，网络服务也将变得不可靠。综上所述，尽管存在一定的开销，但这些开销是网络通信不可或缺的一部分，具有其必要性。

#### 4.5 分析解复用键

1. 以太网头部中哪一部分是解复用（解复用：找到正确的上一层协议来处理到达的包的行为叫做解复用）键并且告知它的下一个高层指的是 IP，在这一包内哪一个值可以表示 IP？

答：以太网头部中的 type 字段是解复用键，它的值为 0x0800，表示 IP 协议。

2. IP 头部中哪一部分是解复用键并且告知它的下一个高层指的是 TCP，在这一包内哪一个值可以表示 TCP？

答：IP 头部中的 protocol 字段是解复用键，它的值为 0x06，表示 TCP 协议。



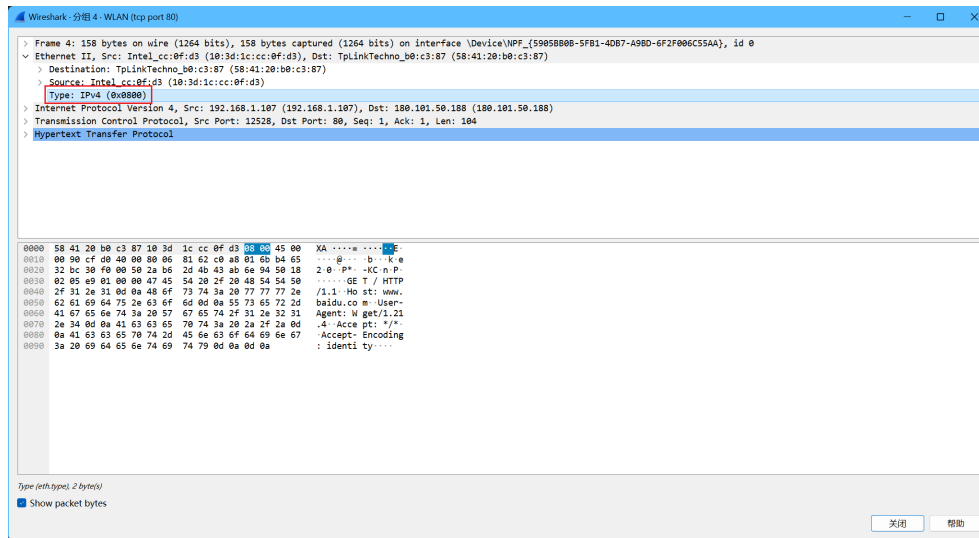


图 12: 以太网头部

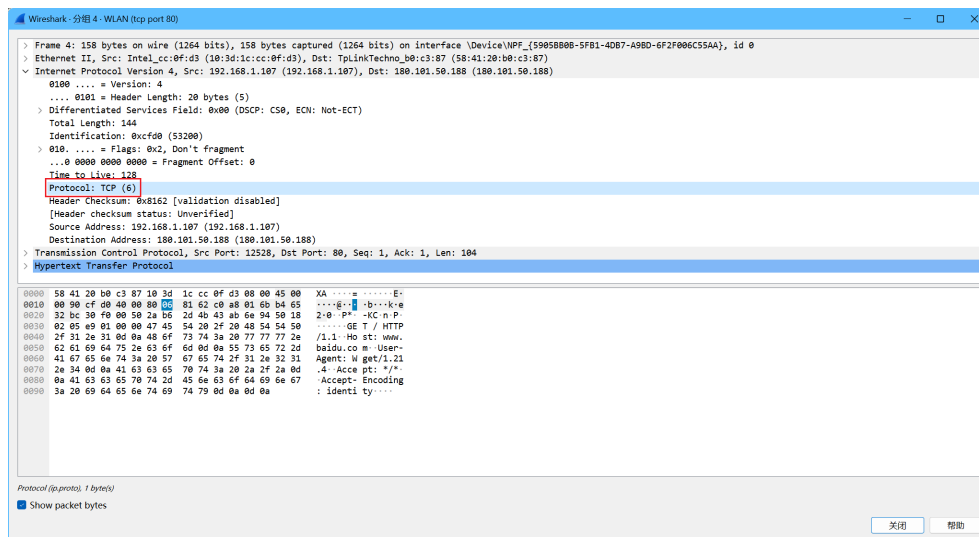


图 13: IP 头部

## 4.6 问题与思考

1. 查看不包含高层数据的短 TCP 数据包，查看它发往哪？不携带高层数据的数据包有用吗？

答：这个包的目的地是我们要访问的网站或者我们自己，通常是确认和控制信息类报文，用于建立 TCP 连接，对 HTTP 协议是有用的。这是 TCP 三次握手的一部分，用于建立 TCP 连接。

2. 在经典的分层模型中，低层字段包装到高层数据包外面，成为一条新消息。但这并非总是如此，Web 响应（一个包含 HTTP 标头和 HTTP 有效负载的 HTTP 消息）可能被转换为多个较低层的消息（即多个 TCP 数据包）。假设你为 Web 响应的第一个和最后一个 TCP 数据包绘制了数据包结构，那么该结构与

经典分层模型有什么不同？

答：第一个 TCP 数据包的负载是 HTTP 协议的头部，而其余 TCP 数据包的负载是 HTTP 的有效载荷。

3. 在上述经典分层模型中，低层字段包装到高层数据包外面，如果较低层添加加密，此模型将如何更改？

答：双方需要协商加密算法，然后在传输数据时，接收方需要解密数据，否则无法解析数据包。

4. 在上述经典分层模型中，低层字段包装到高层数据包外面，如果较低的层添加压缩，此模型将如何更改？

答：发送方可以将解压缩方法附在头部，接收方在接收到数据包后，可以解压缩数据包，然后解析数据包。

## 5 实验结果总结

通过本次实验，我学会了通过 Wireshark 获取各协议层的数据包，掌握了协议层数据包结构，分析了协议开销，分析了解复用键。对计算机网络中的协议层有了更深的理解。

## 6 附录

无