

# 华东师范大学软件学院实验报告

实验课程：计算机网络实践	年级：2023 级本科	实验成绩：
实验名称：Lab 3 IPv4	姓名：张梓卫	
实验编号：（3）	学号：10235101526	实验日期：2024/12/06
指导老师：刘献忠	组号：	实验时间：2 课时

目录

一 实验目的	1	1.5 IP 报文的结构	3
二 实验内容与实验步骤	1	1.6 回答问题	4
三 实验环境	2	2 Step 4: Internet Paths	5
四 实验过程与分析	2	3 Step 5: IP Header Checksum	6
1 获取 IP Packets	2	4 课后思考题	7
1.1 启动 Wireshark 捕获数据包	2	4.1 IPV6	7
1.2 使用 Wget 命令	2	4.2 隧道协议	8
1.3 停止捕获	2	4.3 IP 安全	8
1.4 使用 Tracert 追踪路径信息	3	五 实验结果总结	8
		六 附录	9

## 一 实验目的

该实验是课程《计算机网络实践》的第三次实验，全名《IPV4》，目标如下：

- 1. 学会通过 Wireshark 分析 ip 协议
- 2. 了解 IP 数据报的组成
- 3. 了解 IP 各部分的含义

## 二 实验内容与实验步骤

- 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。
- 打开 windows 的命令行，在里面输入 wget www.sina.com
- 打开 Wireshark，停止捕获。

### 三 实验环境

使用 Wireshark v4.2.5, Windows 11 Pro, Wget Tools 进行实验。  
实验报告使用  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  进行撰写, 使用 Vim 编辑器进行文本编辑。

### 四 实验过程与分析

#### 1 获取 IP Packets

##### 1.1 启动 Wireshark 捕获数据包

启动 Wireshark, 按照实验要求将捕获选项设置为: 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。

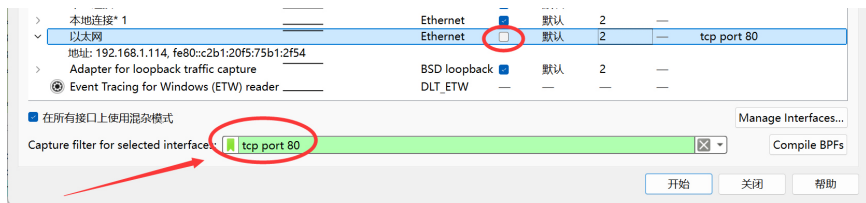


图 1: Wireshark Filter

设置 rename 选项为开启:

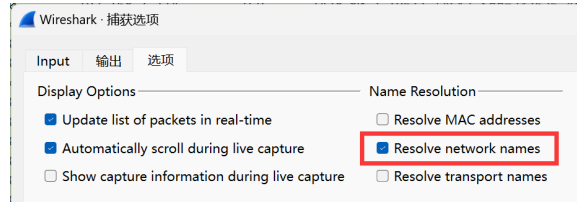


图 2: Wireshark Rename

##### 1.2 使用 Wget 命令

打开命令行, 输入: `wget www.sina.com`



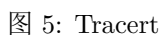
图 3: Wget Command

##### 1.3 停止捕获

根据命令行显示, `www.sina.com` 的主机地址为 61.170.80.226, 在 Wireshark 捕获数据包。

图 4: Get

在 Windows 中，Tracert 命令是直接使用 Tracert www.sina.com 即可进行跃点跟踪路由。使用命令后，等待一段时间，结果如下：



接下来查看 Wireshark 抓包获取的信息，左右两边结合为一张图片，如下图所示：



---

*Lab Report By L<sup>A</sup>T<sub>E</sub>X*

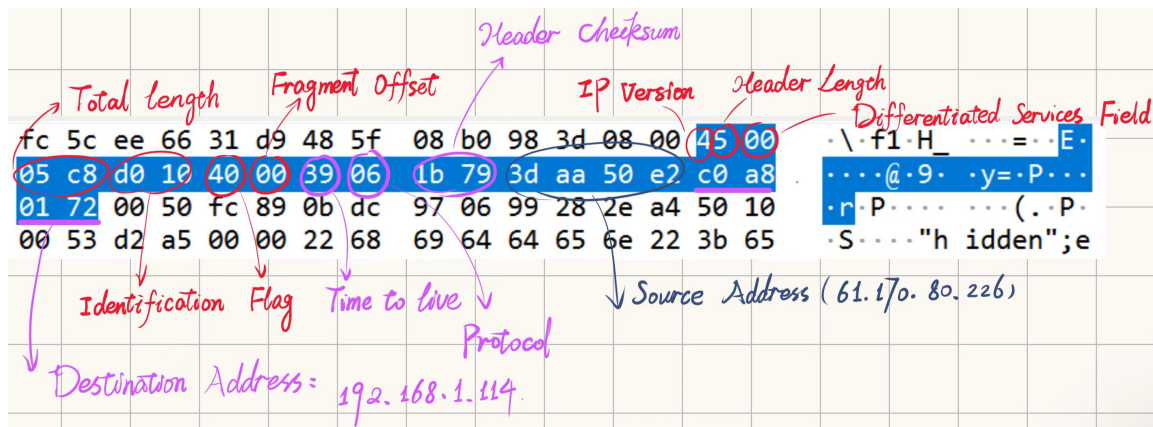


图 7: Bites in Pic

对齐填充后，得到如下的 IP 报文结构，注意这里是从远程计算机发送至本地计算机的抓包：

### 对齐整理：

Version (4 bit)	IHL (4 bit)	Type of Service (8 bit)	Total Length (16 bit)
Identification (16 bit)	Flags (3 bit)		Fragment Offset (13 bit)
Time To Live (8 bit)	Protocol (8 bit)	Header Checksum (16 bit)	
Source IP Address (32 bit)			
Destination IP Address (32 bit)			
Options (variable)		Padding (variable)	

### 1.6 回答问题

由于我选取的抓包示例是从远程计算机发送至本地计算机的，对于实验手册中的问题，我的回答如下：

1.What are the IP addresses of your computer and the remote server?

查看 Source Address 和 Destination Address，分别为：

- 1 Source Address: 61.170.80.226 # 远程服务器的 IP 地址
- 2 Destination Address: 192.168.1.114 # 本地计算机的 IP 地址

2. Does the Total Length field include the IP header plus IP payload, or just the IP payload?

根据 IP 报文结构，我们可以知道，总长度包含了 IP 协议头和 IP 的有效负载，这在 Wireshark 中是显然的。

3. How does the value of the Identification field change or stay the same for different packets? For instance, does it hold the same value for all packets in a TCP connection or does it differ for each packet? Is it the same in both directions? Can you see any pattern if the value does change?

不同的数据包具有不同的标识字段，但若是不分段，则 Identification 为 0x0000。每个报文的 Identification 字段一般来说不相同。IP 软件在存储器中维持一个计数器，每产生一个数据报，计数器就加 1，并将此值赋给标识字段。但这个“标识”不是序号，因为 IP 是无连接服务，数据报不存在按序接收的问题。当数据报由于长度超过网络的 MTU 而必须分片时，这个标识字段的值就被复制到所有的数据报片的标识字段中。相同的标识字段的值使分片后的各数据报片最后能正确地重装成为原来的数据报。

识别字段在双向通信中通常是不同的。每个数据包的识别字段是由发送方独立设置的。由于源主机和目标主机的 IP 栈是独立运行的，从主机 A 到主机 B 的数据包的识别字段通常会有一个序列，而从主机 B 到主机 A 的数据包会有不同的序列。

识别字段的变化是有一定模式的：

顺序递增：大多数系统会为每个发送的数据包顺序递增识别字段的值。例如，如果第一个数据包的 ID 是 1000，那么下一个数据包可能是 1001，以此类推。这在 IPv4 中尤其常见。一些操作系统或配置（例如某些 Linux 设置）会将识别字段设

置为随机值，以避免可预测性，尤其是在设置了不可分片（DF）位时。这种情况下，可能会使得模式不容易观察到。如果设置了不可分片（DF）位，并且数据包没有被分片，则识别字段可能在一些实现中保持恒定。

4. What is the initial value of the TTL field for packets sent from your computer? Is it the maximum possible value, or some lower value?

在我的 Wireshark 抓包中，从个人计算机 (192.168.1.114) 发送的 IP 报文的 TTL 字段初始值为 128，这里应该点开 Source Address 为 61.170.80.226 的 IP 报文，查看 TTL 字段的值。

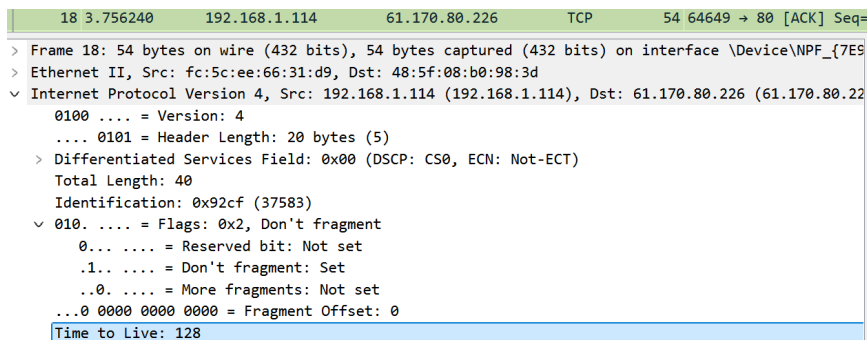


图 8: TTL

5. How can you tell from looking at a packet that it has not been fragmented? Most often IP packets in normal operation are not fragmented. But the receiver must have a way to be sure. Hint: you may need to read your text to confirm a guess.

只需要观察 Wireshark 中的 Don't Fragment 位是否被置为了 1 即可。

```

    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x92cf (37583)
    010. .... = Flags: 0x2, Don't fragment
    0... .... = Reserved bit: Not set
    .1.. .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
  
```

图 9: Don't Fragment

6. What is the length of the IP Header and how is this encoded in the header length field? Hint: notice that only 4 bits are used for this field, as the version takes up the other 4 bits of the byte. You may guess and check your text.

IP 协议头的第 5-8 位 Header Length 即为协议头的长度。这 4 位的十进制值域为 5-15，5 代表没有额外可选的协议头，即长度为 20 字节；为 15 为最大长度，最多为 60 字节。故头部长度范围为 20 字节到 60 字节。

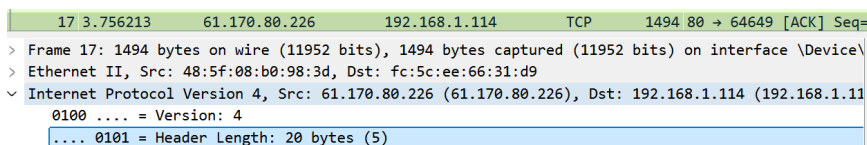


图 10: Header Length

## 2 Step 4: Internet Paths

之前获取到的 tracert 路径如下图所示：

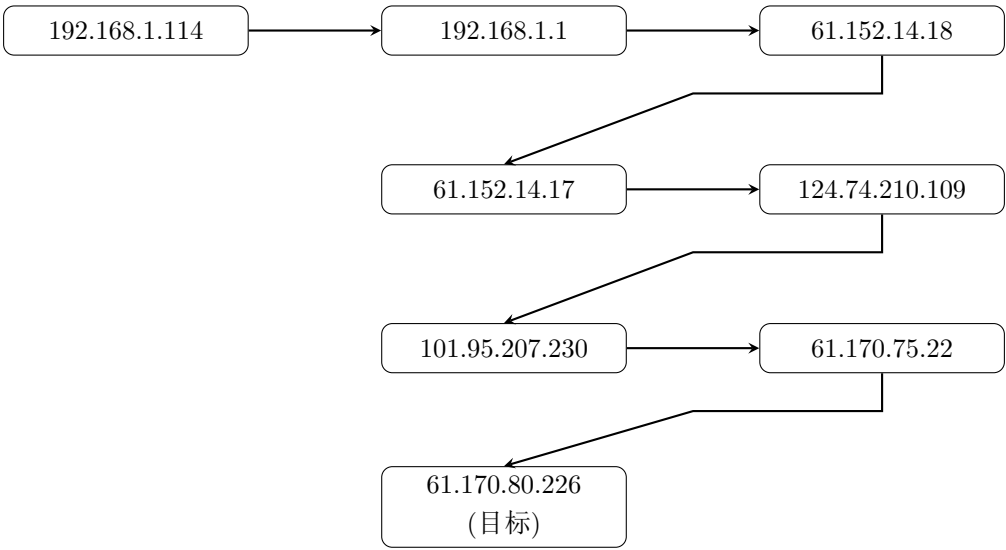
```
26421 master # ?101 x tracert www.sina.com

通过最多 30 个跃点跟踪
到 ww1.sinaimg.cn.w.alikunlun.com [61.170.80.226] 的路由：

 1  <1 毫秒  <1 毫秒  <1 毫秒  192.168.1.1
 2    2 ms    3 ms    2 ms    61.152.14.18
 3    5 ms    3 ms    3 ms    61.152.14.17
 4    8 ms    7 ms    7 ms    124.74.210.109
 5    5 ms    *        *        101.95.207.230
 6    *        *        *        请求超时。
 7    4 ms    4 ms    4 ms    61.170.75.22
 8    4 ms    4 ms    3 ms    61.170.80.226

跟踪完成。
```

图 11: Tracert



3 Step 5: IP Header Checksum

根据 PPT 中的帮助说明，计算对 IP 首部检验和的算法如下：

- (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。

先选择一个 IPV4 协议头，查看其 20 字节的字段。

```
> Frame 17: 1494 bytes on wire (11952 bits), 1494 bytes captured (11952 bits) on interface \Device\NPF{...}
> Ethernet II, Src: 48:5f:08:b0:98:3d, Dst: fc:5c:ee:66:31:d9
  > Internet Protocol Version 4, Src: 61.170.80.226 (61.170.80.226), Dst: 192.168.1.114 (192.168.1.114)
    0100 .... = Version: 4
```

图 12: Checksum

取出这 20 字段，首先把校验和字段置为 0。

1	45 00 -> 4500
2	05 c8 -> 05c8
3	d0 0f -> d00f

```

4  40 00 -> 4000
5  39 06 -> 3906
6  1b 7a -> 0000 # 设置为 0
7  3d aa -> 3daa
8  50 e2 -> 50e2
9  c0 a8 -> c0a8
10 01 72 -> 0172

```

然后进行二进制求和，这一部分我使用计算器来完成：

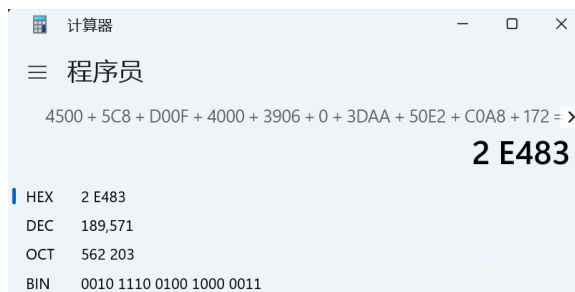


图 13: Checksum

$$E483 + 0002 = E485$$

之后，使用按位取反功能，完成最后一步 Checksum 的计算。



图 14: NOT

这与 Wireshark 中的校验和一致：

```

v Internet Protocol Version 4, Src: 61.170.80.226 (61.170.80.226), Dst: 192.168.1.114 (192.168.1.114)
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1480
    Identification: 0xd00f (53263)
  v 010. .... = Flags: 0x2, Don't fragment
    0... .... = Reserved bit: Not set
    .1... .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 57
    Protocol: TCP (6)
    Header Checksum: 0x1b7a [correct]

```

图 15: The same checksum

## 4 课后思考题

### 4.1 IPV6

现在这个 IPv6 即将普及的时代，它已经不稀有了，在家捣鼓串流时就曾想过用公网 IPv6 来做一些便利性的操作。



在寝室的宿舍网络下，路由器并没有支持 IPv6 的连接，而直接打开 `ipconfig` 看到的 FE80 实际上就和 192.168 没有什么区别，就是一个局域网内分配的地址而已。

我尝试了使用 ISATAP 隧道来建立 IPV4 与 IPV6 的桥梁，可惜失败了。

我使用的代码为：

```
1 netsh interface ipv6 isatap set router isatap.tsinghua.edu.cn
2 netsh interface ipv6 isatap set state endabled
3 netsh interface ipv6 isatap show state
4 >>> enabled
5 ping -6 ipv6.google.com
6 >>> 无法解析域名
```

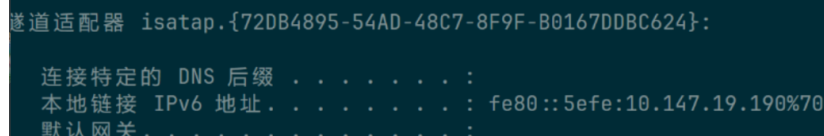


图 16: ISATAP

## 4.2 隧道协议

这是一种基于 tunnels 的技术，即把原始的 IP 包封装在另一个数据包内进行传输。

具体的协议包括 IPsec、SSH、Sock 等，它可以用于跨网络通信（如 IPV4 和 IPV6 之间的过渡），还可以提高安全性，实现私有网络和公共网络的互通。隧道协议的核心工作原理如下：

1. 封装：将原始 IP 包作为有效负载，嵌入到一个新数据包中，新数据包会添加一个外层 IP 头部。
2. 传输：通过隧道协议将封装后的数据包从一个网络节点传输到另一个节点。
3. 解封装：在目标节点处解封装数据包，提取出原始的 IP 包，并将其转发到最终的目的地。

## 4.3 IP 安全

不同的 IP 地址被分配给不同的地区、机构，因此利用公网 IP 能够获取一定的地理信息知识。

IPsec (IP Security) 是 IETF 制定的三层隧道加密协议，它为 Internet 上传输的数据提供了高质量的、可互操作的、基于密码学的安全保证。

1. 数据机密性 (Confidentiality)：IPsec 发送方在通过网络传输包前对包进行加密。
2. 数据完整性 (Data Integrity)：IPsec 接收方对发送方发送来的包进行认证，以确保数据在传输过程中没有被篡改。
3. 数据来源认证 (Data Authentication)：IPsec 在接收端可以认证发送的 IPsec 报文的发送端是否合法。
4. 防重放 (Anti-Replay)：IPsec 接收方可检测并拒绝接收过时或重复的报文。

# 五 实验结果总结

通过本次实验，我对 IP 协议的组成、解析方法及其相关网络协议有了更深入的理解，并熟练掌握了 Wireshark 抓包工具的使用方法。

实验详细研究了 IPv4 协议的头部结构，通过实际抓取 IP 数据包并标注各字段（如版本、头长度、总长度、标识字段等），加深了对 IP 报文组成部分的认知。这种对细节的深入解析，让我在理论知识上有了更扎实的积累。在分析过程中，我结合实际网络行为（如访问新浪网站），通过可视化界面和协议分解信息，验证了 IP 报文中校验和、TTL 等关键字段的计算方法。在实验中，我研究了 IP 协议的头部校验和的计算方法，验证了 Wireshark 中校验和的正确性。同时，我通过 `tracert` 命令追踪数据包的网络路径，深入理解了 IP 分片、跳数限制 (TTL) 的作用，以及数据包在网络中的传输过程。



## 六 附录

### 参考资料

- 隧道协议: <https://blog.csdn.net/wangjianno2/article/details/75208036>
- IPv6: <https://jingyan.baidu.com/article/22fe7ced67c9443002617f94.html>
- IP 地址查询: <https://www.chaipip.com/ip.php>
- 查询本机 IP: <https://ip.cn/>