《网络与通信》课程实验报告

实验三：数据包结构分析

姓名	张泽毅	院系	计算机学院	学号	21120971
任课教师	刘通		指导教师	刘通	
实验地点	计 706		实验时间	周三 7-8	
实验课表现	出勤、表现得分(10)		实验报告得分(40)		实验总分	
	操作结果得分(50)					

实验目的：

1. 了解 Sniffer 的工作原理，掌握 Sniffer 抓包、记录和分析数据包的方法；
2. 在这个实验中，你将使用抓包软件捕获数据包，并通过数据包分析每一层协议。

实验内容：

使用抓包软件捕获数据包，并通过数据包分析每一层协议。

实验要求：（学生对预习要求的回答）（10 分）　　得分：

- 常用的抓包工具

1. Wireshark：

Wireshark是一款开源的网络分析工具，用于捕获、分析和解释网络数据包。它能够监控计算机网络上的数据流量，显示各种协议的详细信息，并提供强大的过滤和搜索功能，帮助用户深入了解网络通信、故障排除以及安全性分析。Wireshark用户友好，支持多平台操作系统，是网络专业人士和安全研究人员常用的工具之一。

2. tcpdump：

tcpdump是一个命令行网络抓包工具，主要用于Unix和Linux系统。它能够实时捕获网络数据包，并以文本形式显示或保存到文件中。tcpdump具有灵活的过滤和显示功能，使得用户能够针对特定的网络流量进行监视和分析，是系统管理员和网络工程师常用的命令行工具。

3. Fiddler：

Fiddler 是一个用于抓取HTTP和HTTPS数据包的工具，主要用于网络调试和性能优化。它允许用户监视 Web 应用程序和浏览器之间的HTTP/HTTPS通信，捕获请求和响应数据，帮助开发人员诊断问题、调试网页应用程序，优化性能，检查网络安全性，并进行各种网络分析。

4. Charles Proxy：

Charles Proxy是一个跨平台的网络抓包工具，能够拦截和分析HTTP和HTTPS通信。它具有友好的用户界面，支持Windows、macOS和Linux系统。Charles Proxy通常用于开发人员调试Web应用程序，可以查看网络请求和响应的详细信息，帮助开发人员快速定位问题并进行修复。

5. Burp Suite：

Burp Suite是一套专业的网络安全测试工具，主要用于渗透测试和漏洞扫描。它包含了代理、扫描、爬虫、渗透测试等多个模块，其中代理模块具有强大的抓包和修改功能。Burp Suite能够拦截和修改HTTP请求和响应，帮助安全专家发现Web应用程序中的漏洞和安全风险，是一款广泛用于渗透测试领域的工具。

6. Sniffer：

Sniffer是一个通用的抓包工具，可用于捕获和分析网络流量。它可以在多种操作系统上运行，并具备实时监视网络数据包、解析各种协议、检测网络问题、发现潜在安全威胁的功能。Sniffer的灵活性和可扩展性使其成为网络管理员和安全专家常用的工具，用于定位网络故障、检测恶意活动等任务。

实验过程中遇到的问题如何解决的？（10 分）	得分：

问题 1：发现在抓包的过程中抓到了很多源地址和目标地址都不是本机的数据包。

解决：这是由于在 WireShark 中打开了混杂模式，在混杂模式下，Wireshark 会捕获经过网络中的所有数据包，而不仅仅是与当前机器相关的数据包。这意味着 Wireshark 能够监控整个网络段的流量，包括发往其他设备的数据包。而在普通模式下，Wireshark 只会捕获发送给当前机器的数据包，或者由当前机器生成的数据包。这意味着它只会监控和显示与本机相关的网络通信，而不会显示传输到其他设备或网络节点的数据包。

问题 2：打开 Wireshark 没看到数据包抓取界面。面对众多的数据包不知道如何进行筛选、分析。不了解各个数据包背景色的含义。

解决：需要先选择对应的接口来监控数据包。通过自学课程内容、查阅资料、与同学讨论实践，结合抓到的数据包进行学习。关于数据包的背景色，黑色背景代表报文的各类错误，红色背景代表各类异常情景，其它颜色代表正常。

问题 3：在学习 tcp 数据包时，尝试使用实验二设计的 tcp 传输系统进行实践，但是在 WLAN 中无法获得相关数据包。

解决：由于实验二设计的是本机同时作为服务端与客户端，所以需要捕获的是本地计算机上产生的环回流量，这些流量不会通过物理网络适配器传输，而是在计算机内部回环。因此 Wireshark 提供了一种特殊的适配器，用于捕获这种环回流量：在 Wireshark 的接口列表中有一个特殊的适配器，命名为“Loopback”，选择这个适配器可以捕获本地计算机上的环回流量。

问题 4：筛选数据包时发现没有百度服务器地址与本机 ip 地址 tcp 三次握手的数据包，却有百度 ip 地址与本机 ip 地址对应的 icmp 类型的数据包。

解决：发生这一情况可能是由于以下原因：防火墙或路由器可能过滤了 TCP 连接请求，网络配置问题，百度服务器端问题或者正在尝试访问的网站使用了 HTTPS 协议，导致 Wireshark 无法解析加密的数据包内容。需要检查网络设置、防火墙配置、访问的网站是否使用了 HTTPS，或者进一步分析网络设备和服务器端配置来解决。**（本问题解决内容均由上网查阅得到）**

本次实验的体会（结论）（10 分）	得分：

本次实验在自学完之后操作起来是相对比较简单的，但是其中对数据包进行分析需要对课堂上学到的知识以及自学的相关内容有很好地实际运用。

对于实验本身来说，经过本次学习，我掌握了如何利用 WireShark 软件进行抓包，了解了数据包的结构，并学会了如何对数据包进行分析，对各类协议也有了更加深刻的了解。通过本次实验，我对计算机的信息运输以及数据包在各个层之间的传输方式有了一定的认识。借助于本次实验以及实验二的成果，我对 TCP 的三次握手与四次挥手也有了更加深入的理解与直观的感受。实验中遇到的小错误也引导着我，让我对相关知识有更全面的认识。总的来说，这次实验让我收获了很多。

思考题：（10 分）	
思考题 1：（4 分）	得分：

写出捕获的数据包格式。

- ARP 数据包（数据包分析以第一个为例）

```
45 22.322265     62:4b:1a:61:e9:bb    Broadcast           ARP         42 Who has 192.168.3.55? (ARP Probe)
66 31.333108     82:a0:7d:b0:5d:6d    Broadcast           ARP         60 Who has 192.168.3.59? Tell 192.168.3.1
80 47.818857     52:57:13:3f:34:ce    Broadcast           ARP         42 Who has 192.168.3.1? Tell 192.168.3.16
```

ARP 数据包有三层结构：

```
> Frame 45: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{9AC86AF2-6D8B-47BC-8310-F006282A9C1B}, id 0
> Ethernet II, Src: 62:4b:1a:61:e9:bb (62:4b:1a:61:e9:bb), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Address Resolution Protocol (ARP Probe)
```

Frame（物理层）

```
v Frame 45: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{9AC86AF2-6D8B-47BC-8310-F006282A9C1B}, id 0
    Section number: 1
  > Interface id: 0 (\Device\NPF_{9AC86AF2-6D8B-47BC-8310-F006282A9C1B})
    Encapsulation type: Ethernet (1)
    Arrival Time: Oct  7, 2023 20:40:25.029716000 中国标准时间
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1696682425.029716000 seconds
    [Time delta from previous captured frame: 0.102673000 seconds]
    [Time delta from previous displayed frame: 0.102673000 seconds]
    [Time since reference or first frame: 22.322265000 seconds]
    Frame Number: 45
    Frame Length: 42 bytes (336 bits)
    Capture Length: 42 bytes (336 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:arp]
    [Coloring Rule Name: ARP]
    [Coloring Rule String: arp]
```

1. **Frame 45**：这是 Wireshark 捕获的第 45 个数据包。
2. **42 bytes on wire (336 bits)**：数据包总大小为 42 字节（336 比特）。
3. **\Device\NPF_{9AC86AF2-6D8B-47BC-8310-F006282A9C1B}**：这是捕获接口的标识符。
4. **Interface id: 0:** 数据包被捕获的接口信息。
5. **Encapsulation type: Ethernet (1)**：数据包使用 Ethernet 以太网封装。
6. **Arrival Time**：数据包抵达的时间是 2023 年 10 月 7 日，20 时 40 分 25.029 秒。
7. **[Time shift for this packet: 0.000000000 seconds]**： 时间偏移。
8. **Epoch Time: 1696682425.029716000 seconds:** 从计算机元年开始计算的时间。
9. **[Time delta from previous captured frame:]:** 当前数据包与前一个捕获的数据包之间的时间差。
10. **[Time delta from previous displayed frame:]**：当前数据包与前一个显示的数据包之间的时间差。
11. **[Time since reference or first frame:]:** 第一个数据包或参考数据包到这个数据包的时间差
12. **Frame Number: 45**：数据包在 Wireshark 中的序号为 45。
13. **Frame Length: 42 bytes (336 bits)**：数据包的实际长度为 42 字节（336 比特）。
14. **Capture Length: 42 bytes (336 bits)**：Wireshark 实际捕获并显示的数据包长度为 42 字节（336 比特）。
15. **[Frame is marked: False]、[Frame is ignored: False]:** 这个数据包是否被标记或忽略。
16. **[Protocols in frame: eth:ethertype:arp]**：数据包中包含的协议是以太网和 ARP。
17. **[Coloring Rule Name: ARP]、[Coloring Rule String: arp]:** 颜色规则的名称和匹配规则字符串

Ethernet Ⅱ（数据链路层）：

```
Ethernet II, Src: 62:4b:1a:61:e9:bb (62:4b:1a:61:e9:bb), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  > Source: 62:4b:1a:61:e9:bb (62:4b:1a:61:e9:bb)
    Type: ARP (0x0806)
```

1. **Ethernet II**：数据包使用 Ethernet II 帧格式。
   - **Source (源 MAC 地址)**: 62:4b:1a:61:e9:bb
   - **Destination (目标 MAC 地址)**: Broadcast (ff:ff:ff:ff:ff:ff)，这表示数据包是广播到网络中的所有设备。
2. **Type: ARP (0x0806)**：Ethernet II 帧中的类型字段指示这是一个 ARP 协议的数据包（以十六进制表示为 0x0806）。

Address Resolution Protocol（地址解析协议）：

```
Address Resolution Protocol (ARP Probe)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    [Is probe: True]
    Sender MAC address: 62:4b:1a:61:e9:bb (62:4b:1a:61:e9:bb)
    Sender IP address: 0.0.0.0
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.3.55
```

1. **Hardware type: Ethernet (1)**：表示硬件类型是以太网，它用于指示 ARP 包中包含的硬件地址（MAC 地址）的类型。
2. **Protocol type: IPv4 (0x0800)**：表示协议类型是 IPv4，指示 ARP 包中包含的协议地址（IPv4 地址）的类型。
3. **Hardware size: 6**：指示硬件地址的大小为 6 字节。
4. **Protocol size: 4**：指示协议地址的大小为 4 字节。
5. **Opcode: request (1)**：指示这是一个 ARP 请求包，用于询问某个 IP 地址对应的 MAC 地址。
6. **[Is probe: True]**：这个 ARP 请求被标记为“探测包”，表示它是用于探测网络中是否已经有设备在使用指定的 IP 地址。
7. **Sender MAC address: 62:4b:1a:61:e9:bb**：ARP 请求的发送者 MAC 地址。
8. **Sender IP address: 0.0.0.0**：ARP 请求的发送者 IP 地址。这里是 0.0.0.0，表示请求者并未知道自己的 IP 地址，因此在 ARP 请求中留空。
9. **Target MAC address: 00:00:00:00:00:00**：ARP 请求的目标 MAC 地址。在 ARP 请求中，通常会将目标 MAC 地址设置为全零，因为请求者并不知道目标设备的 MAC 地址。
10. **Target IP address: 192.168.3.55**：ARP 请求的目标 IP 地址。

- ICMP 数据包

91416	117.046794	192.168.3.6	180.101.50.242	ICMP	74	Echo (ping) request id=0x0001, seq=52/13312, ttl=128 (reply in 91426)
91426	117.056519	180.101.50.242	192.168.3.6	ICMP	74	Echo (ping) reply id=0x0001, seq=52/13312, ttl=52 (request in 91416)
92028	118.056871	192.168.3.6	180.101.50.242	ICMP	74	Echo (ping) request id=0x0001, seq=53/13568, ttl=128 (reply in 92035)
92035	118.069597	180.101.50.242	192.168.3.6	ICMP	74	Echo (ping) reply id=0x0001, seq=53/13568, ttl=52 (request in 92028)
93271	119.063069	192.168.3.6	180.101.50.242	ICMP	74	Echo (ping) request id=0x0001, seq=54/13824, ttl=128 (reply in 93301)
93301	119.074650	180.101.50.242	192.168.3.6	ICMP	74	Echo (ping) reply id=0x0001, seq=54/13824, ttl=52 (request in 93271)
94046	120.074552	192.168.3.6	180.101.50.242	ICMP	74	Echo (ping) request id=0x0001, seq=55/14080, ttl=128 (reply in 94053)
94053	120.083318	180.101.50.242	192.168.3.6	ICMP	74	Echo (ping) reply id=0x0001, seq=55/14080, ttl=52 (request in 94046)

ICMP 数据包有四层结构：

```
> Frame 91416: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{9AC86AF2-6D8B-47BC-8310-F006282A9C1B}, id 0
> Ethernet II, Src: IntelCor_f0:89:3a (cc:d9:ac:f0:89:3a), Dst: 82:a0:7d:b0:5d:6d (82:a0:7d:b0:5d:6d)
> Internet Protocol Version 4, Src: 192.168.3.6, Dst: 180.101.50.242
> Internet Control Message Protocol
```

Frame（物理层）：

```
v Frame 91416: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{9AC86AF2-6D8B-47BC-8310-F006282A9C1B}, id 0
    Section number: 1
  > Interface id: 0 (\Device\NPF_{9AC86AF2-6D8B-47BC-8310-F006282A9C1B})
    Encapsulation type: Ethernet (1)
    Arrival Time: Oct  7, 2023 21:10:06.291906000 中国标准时间
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1696684206.291906000 seconds
    [Time delta from previous captured frame: 0.004531000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 117.046794000 seconds]
    Frame Number: 91416
    Frame Length: 74 bytes (592 bits)
    Capture Length: 74 bytes (592 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:icmp:data]
    [Coloring Rule Name: ICMP]
    [Coloring Rule String: icmp || icmpv6]
```

1. **Frame 91416**：这是 Wireshark 捕获的第 91416 个数据包。
2. **74 bytes on wire (592 bits)**：数据包总大小为 74 字节（592 比特）。
3. **\Device\NPF_{9AC86AF2-6D8B-47BC-8310-F006282A9C1B}**：这是捕获接口的标识符。
4. **Encapsulation type: Ethernet (1)**：数据包使用 Ethernet 封装。
5. **Arrival Time**：数据包抵达的时间是 2023 年 10 月 7 日，21 时 10 分 06.291 秒。
6. **Frame Number: 91416**：数据包在 Wireshark 中的序号为 91416。
7. **Frame Length: 74 bytes (592 bits)**：数据包的实际长度为 74 字节（592 比特）。
8. **Capture Length: 74 bytes (592 bits)**：Wireshark 实际捕获并显示的数据包长度为 74 字节（592 比特）。
9. **Protocols in frame: eth:ethertype:ip:icmp:data**：数据包中包含的协议是以太网、IP、ICMP 和数据。
10. **[Coloring Rule Name: ICMP]**：Wireshark 使用颜色规则对不同类型的数据包进行标记。在这里，该数据包被标记为 ICMP 数据包。

Ethernet Ⅱ（数据链路层）：

```
v Ethernet II, Src: IntelCor_f0:89:3a (cc:d9:ac:f0:89:3a), Dst: 82:a0:7d:b0:5d:6d (82:a0:7d:b0:5d:6d)
  > Destination: 82:a0:7d:b0:5d:6d (82:a0:7d:b0:5d:6d)
  > Source: IntelCor_f0:89:3a (cc:d9:ac:f0:89:3a)
    Type: IPv4 (0x0800)
```

1. **Ethernet II**：表示这是一个 Ethernet II 数据帧，即以太网帧格式。
2. **Src: IntelCor_f0:89:3a (cc:d9:ac:f0:89:3a)**：源 MAC 地址，表示数据帧的发送者的物理硬件地址。源 MAC 地址是 **cc:d9:ac:f0:89:3a**，这通常是网络接口卡的唯一标识符。
3. **Dst: 82:a0:7d:b0:5d:6d**：目标 MAC 地址，表示数据帧的接收者的物理硬件地址。目标 MAC 地址是 **82:a0:7d:b0:5d:6d**，表示数据帧将被发送到这个 MAC 地址所对应的设备。
4. **Type: IPv4 (0x0800)**：以太网帧中的类型字段，表示帧的上层协议类型。在这里，类型字段的值为 **0x0800**，表示上层协议是 IPv4。

Internet Protocol Version 4（网络层）：

```
Internet Protocol Version 4, Src: 192.168.3.6, Dst: 180.101.50.242
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0xfeef (65263)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: ICMP (1)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.3.6
    Destination Address: 180.101.50.242
```

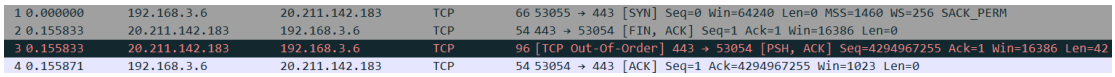
1. **Internet Protocol Version 4**：表示这是一个 IPv4 数据包。
2. **Src: 192.168.3.6**：源 IP 地址，表示数据包的发送者的 IP 地址。
3. **Dst: 180.101.50.242**：目标 IP 地址，表示数据包的接收者的 IP 地址。
4. **0100 .... = Version: 4**：版本号，指示这是 IPv4 协议的数据包。
5. **.... 0101 = Header Length: 20 bytes (5)**：IP 头部的长度为 20 字节。
6. **Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)**：指示服务质量和流量分类信息，这里 DSCP 和 ECN 的值均为 0。
7. **Total Length: 60**：整个 IP 数据包的长度是 60 字节。
8. **Identification: 0xfeef (65263)**：用于标识数据包的唯一性，通常由发送方设置。
9. **000. .... = Flags: 0x0**：用于分片控制的标志位。标志位为 0，表示不分片。
10. **...0 0000 0000 0000 = Fragment Offset: 0**：分片偏移，指示分片在原始数据包中的位置。偏移为 0，表示这个数据包没有分片。
11. **Time to Live: 128**：生存时间，表示数据包在网络中可以经过的最大路由器跳数。每经过一个路由器，该值就会减 1。当该值为 0 时，数据包会被丢弃。
12. **Protocol: ICMP (1)**：指示 IP 数据包的上层协议类型。这里的值为 1，表示上层协议是 ICMP。
13. **Header Checksum: 0x0000 [validation disabled]**：IP 头部的校验和，用于验证 IP 头部的完整性。
14. **[Header checksum status: Unverified]**：指示 IP 头部校验和的验证状态为“未验证”。

Internet Control Message Protocol（ICMP 协议分析请求包）：

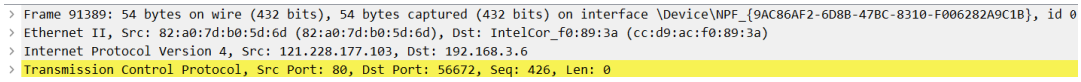
```
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4d27 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 52 (0x0034)
    Sequence Number (LE): 13312 (0x3400)
    [Response frame: 91426]
  > Data (32 bytes)
```

1. **Type: 8 (Echo (ping) request)**：指示 ICMP 消息类型为“Echo 请求”，就是常用的“ping 请求”。
2. **Code: 0**：指示具体的消息代码，对于 Echo 请求，代码通常为 0。
3. **Checksum: 0x4d27 [correct]**：ICMP 头部的校验和，用于验证 ICMP 消息的完整性。校验和的值为 0x4d27，表示校验和正确。
4. **Checksum Status: Good**：指示 ICMP 头部校验和的验证状态为“正确”。
5. **Identifier (BE): 1 (0x0001)**：用于唯一标识 ICMP 请求的标识符，这个字段的值为 1。
6. **Identifier (LE): 256 (0x0100)**：同一个标识符字段，但以小端格式表示，值为 256。
7. **Sequence Number (BE): 52 (0x0034)**：序列号，用于标识每个 ICMP 请求的顺序。这个字段的值为 52。
8. **Sequence Number (LE): 13312 (0x3400)**：同一个序列号字段，但以小端格式表示，值为 13312。
9. **[Response frame: 91426]**：指示响应将在 Wireshark 的帧号 91426 中找到。
10. **Data (32 bytes)**：ICMP 请求的数据部分，长度为 32 字节。

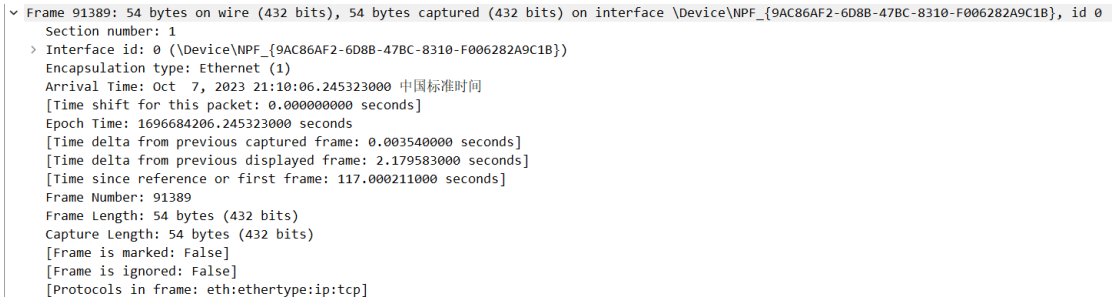
- **TCP 协议数据包**



TCP 协议数据包有四层结构：

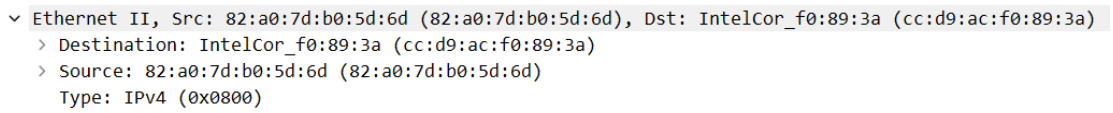


Frame（物理层）：



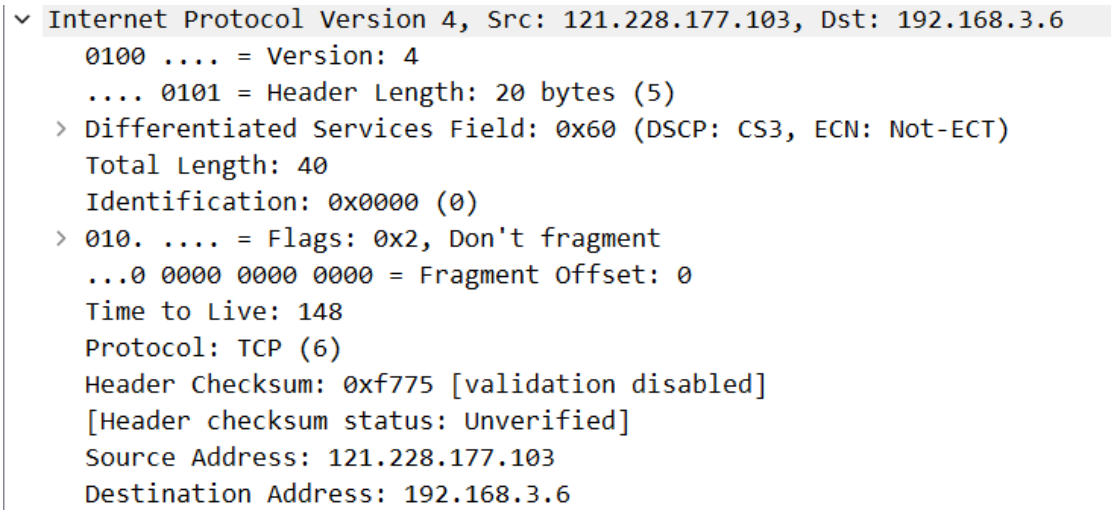
1. **Frame 91389**：这是 Wireshark 捕获的第 91389 个数据包。
2. **54 bytes on wire (432 bits)**：数据包总大小为 54 字节（432 比特）。
3. **\Device\NPF_{9AC86AF2-6D8B-47BC-8310-F006282A9C1B}**：捕获接口的标识符。
4. **Encapsulation type: Ethernet (1)**：数据包使用 Ethernet 封装。
5. **Arrival Time**：数据包抵达的时间是 2023 年 10 月 7 日，21 时 10 分 06.245 秒。
6. **Frame Number: 91389**：数据包在 Wireshark 中的序号为 91389。
7. **Frame Length: 54 bytes (432 bits)**：数据包的实际长度为 54 字节（432 比特）。
8. **Capture Length: 54 bytes (432 bits)**：Wireshark 实际捕获并显示的数据包长度为 54 字节（432 比特）。
9. **Protocols in frame: eth:ethertype:ip:tcp**：数据包中包含的协议是以太网、IP 和 TCP。

Ethernet Ⅱ（数据链路层）：



1. **Ethernet II**：表示这是一个 Ethernet II 数据帧，即以太网帧格式。
2. **Src: 82:a0:7d:b0:5d:6d**：源 MAC 地址，表示数据帧的发送者的物理硬件地址。源 MAC 地址是 **82:a0:7d:b0:5d:6d**，通常是网络接口卡的唯一标识符。
3. **Dst: IntelCor_f0:89:3a (cc:d9:ac:f0:89:3a)**：目标 MAC 地址，表示数据帧的接收者的物理硬件地址。目标 MAC 地址是 **cc:d9:ac:f0:89:3a**，表示数据帧将被发送到这个 MAC 地址所对应的设备。
4. **Type: IPv4 (0x0800)**：以太网帧中的类型字段，表示帧的上层协议类型。类型字段的值为 **0x0800**，表示上层协议是 IPv4。

Internet Protocol Version 4（网络层）：



1. **Internet Protocol Version 4**：这是一个 IPv4 数据包。
2. **Src: 121.228.177.103**：源 IP 地址，表示数据包的发送者的公网 IP 地址。
3. **Dst: 192.168.3.6**：目标 IP 地址，表示数据包的接收者的私有 IP 地址。
4. **0100 .... = Version: 4**：版本号，表示这是 IPv4 协议的数据包。
5. **.... 0101 = Header Length: 20 bytes (5)**：IP 头部的长度为 20 字节（因为 IPv4 头部长度字段表示的是 32 位字（4 字节）的数量）。
6. **Differentiated Services Field: 0x60 (DSCP: CS3, ECN: Not-ECT)**：指示服务质量和流量分类信息。DSCP 的值为 60，对应服务类别 CS3，ECN 标志位为 Not-ECT（未设置）。
7. **Total Length: 40**：整个 IP 数据包的长度，包括 IP 头部和数据部分。这个数据包的总长度是 40 字节。
8. **Identification: 0x0000 (0)**：用于标识数据包的唯一性，通常由发送方设置。这个数据包的标识字段为 0。
9. **010. .... = Flags: 0x2, Don't fragment**：用于分片控制的标志位。这里的标志位为 10，表示“不分片”。
10. **...0 0000 0000 0000 = Fragment Offset: 0**：分片偏移，指示分片在原始数据包中的位置。这里的偏移为 0，表示这个数据包没有分片。
11. **Time to Live: 148**：生存时间，表示数据包在网络中可以经过的最大路由器跳数。每经过一个路由器，该值就会减 1。当该值为 0 时，数据包会被丢弃。

12. **Protocol: TCP (6)**：指示 IP 数据包的上层协议类型。这里的值为 6，表示上层协议是 TCP。
13. **Header Checksum: 0xf775 [validation disabled]**：IP 头部的校验和，用于验证 IP 头部的完整性。
14. **[Header checksum status: Unverified]**：指示 IP 头部校验和的验证状态为“未验证”。

Transmission Control Protocol（传输层）：

```
Transmission Control Protocol, Src Port: 80, Dst Port: 56672, Seq: 426, Len: 0
  Source Port: 80
  Destination Port: 56672
  [Stream index: 168]
  [Conversation completeness: Complete, WITH_DATA (63)]
  [TCP Segment Len: 0]
  Sequence Number: 426    (relative sequence number)
  Sequence Number (raw): 1259307406
  [Next Sequence Number: 426    (relative sequence number)]
  Acknowledgment Number: 0
  Acknowledgment number (raw): 0
  0101 .... = Header Length: 20 bytes (5)
> Flags: 0x004 (RST)
  Window: 0
  [Calculated window size: 0]
  [Window size scaling factor: 128]
  Checksum: 0x1698 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
> [Timestamps]
```

1. **Source Port: 80**：源端口号，表示发送者应用程序的端口号。
2. **Destination Port: 56672**：目标端口号，表示接收者应用程序的端口号。
3. **[Stream index: 168]**：表示该 TCP 流的索引号。
4. **[Conversation completeness: Complete, WITH_DATA (63)]**：表示 TCP 会话的完整性，包括数据。在这个会话中，63 个数据包被完整地交换。
5. **[TCP Segment Len: 0]**：TCP 段的长度，这个段没有携带数据。
6. **Sequence Number: 426**：TCP 序列号。
7. **Sequence Number (raw): 1259307406**：序列号的原始值，表示相对于整个网络连接的绝对序列号。
8. **[Next Sequence Number: 426 (relative sequence number)]**：下一个期望的序列号，表示接收者期望接收的下一个数据包的序列号。
9. **Acknowledgment Number: 0**：确认号，表示接收者期望接收的下一个数据包的序列号。
10. **Acknowledgment number (raw): 0**：确认号的原始值，表示相对于整个网络连接的绝对序列号。
11. **0101 .... = Header Length: 20 bytes (5)**：TCP 头部的长度为 20 字节
12. **Flags: 0x004 (RST)**：标志位，指示 TCP 包的状态。标志位的值为 0x004，表示连接被重置。
13. **Window: 0**：窗口大小，表示发送者的可用缓冲区大小。
14. **Checksum: 0x1698 [unverified]**：TCP 头部的校验和，用于验证 TCP 头部的完整性。在这个数据包中，校验和是 0x1698，但是未经验证。
15. **[Checksum Status: Unverified]**：表示 TCP 头部校验和的验证状态为“未验证”。

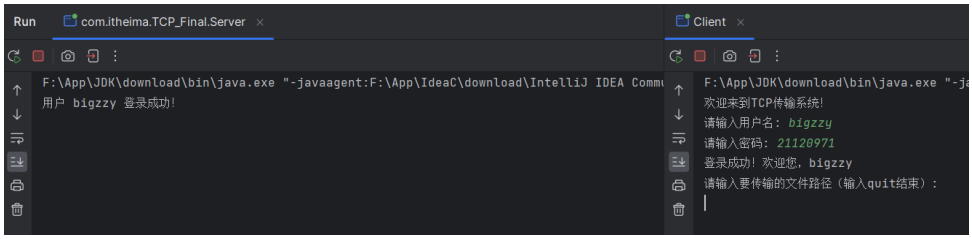
16. **Urgent Pointer: 0**：紧急指针，用于指示紧急数据的位置。紧急指针为 0，表示没有紧急数据。
17. **[Timestamps]**：表示 TCP 数据包中包含了时间戳信息。

思考题2：（6分）　　得分：

写出实验过程并分析实验结果。

1. **实验设计：**以TCP三次握手为例分析捕获到的TCP协议内容。

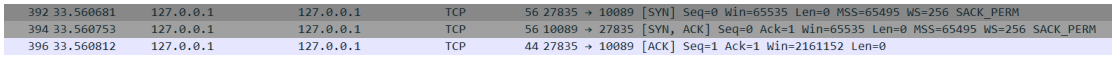
2. **实验过程：**
   - 打开 wireshark 软件选择 adapter for loopback traffic capture 开始捕获。
   - 将端口号 10089 作为捕获过滤条件进行过滤。
   - 打开实验二设计的 TCP 文件传输系统并运行。



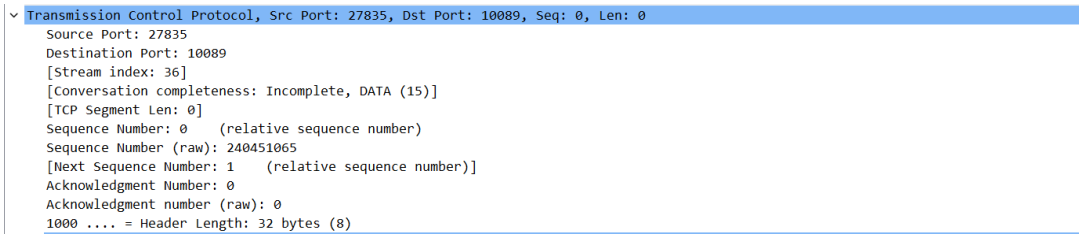
   - 开始分析捕获的 TCP 数据包。

3. **分析数据包：**

捕获三次握手数据包如图所示：



- **第一次握手：客户端发送一个 TCP，标志位为 SYN=1，序号 seq 为 0，27835 → 10089，代表客户端请求建立连接；**



```
Flags: 0x002 (SYN)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...0 .... = Acknowledgment: Not set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  > .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ··········S·]
```

- **第二次握手：服务器向客户端返回一个数据包，SYN=1，ACK=1，10089 → 27835，将确认序号(Acknowledgement Number)设置为客户的序号 seq+1，即 0+1=1；**

```
Transmission Control Protocol, Src Port: 10089, Dst Port: 27835, Seq: 0, Ack: 1, Len: 0
  Source Port: 10089
  Destination Port: 27835
  [Stream index: 36]
  [Conversation completeness: Incomplete, DATA (15)]
  [TCP Segment Len: 0]
  Sequence Number: 0    (relative sequence number)
  Sequence Number (raw): 3571151040
  [Next Sequence Number: 1    (relative sequence number)]
  Acknowledgment Number: 1    (relative ack number)
  Acknowledgment number (raw): 240451066
  1000 .... = Header Length: 32 bytes (8)
  Flags: 0x012 (SYN, ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  > .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ·····A··S·]
```

- **第三次握手：客户端收到服务器发来的数据包后检查确认序号是否正确，以及标志位 ACK 是否为 1。如果正确，客户端会再向服务器端发送一个数据包，SYN=0，ACK=1，确认序号=1，并且把服务器发来 ACK 的序号 seq 加 1 发送给对方。客户端收到后，确认序号值与 ACK=1，27835 → 10089，至此，一次 TCP 连接就此建立，可以进行数据的传输了。**

```
Transmission Control Protocol, Src Port: 27835, Dst Port: 10089, Seq: 1, Ack: 1, Len: 0
    Source Port: 27835
    Destination Port: 10089
    [Stream index: 36]
    [Conversation completeness: Incomplete, DATA (15)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 240451066
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3571151041
    0101 .... = Header Length: 20 bytes (5)
  Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Accurate ECN: Not set
      .... 0... .... = Congestion Window Reduced: Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
```

4. **实验分析：**通过这一实验，既涵盖了wireshark包括捕获数据包、过滤数据包、分析数据包的操作，同时也结合了实验二的成果进一步理解了TCP协议三次握手的内容。本实验以分析实验二TCP文件传输系统中服务端与客户端产生的三次握手为例实现了抓包与分析的内容。通过wireshark直观明了地展示了一次三次握手

额外（四次挥手）：

9804	763.713527	127.0.0.1	127.0.0.1	TCP	44 27835 → 10089 [FIN, ACK] Seq=25 Ack=36 Win=2161152 Len=0
9805	763.713548	127.0.0.1	127.0.0.1	TCP	44 10089 → 27835 [ACK] Seq=36 Ack=26 Win=2161152 Len=0
9806	763.715471	127.0.0.1	127.0.0.1	TCP	44 10089 → 27835 [FIN, ACK] Seq=36 Ack=26 Win=2161152 Len=0
9807	763.715504	127.0.0.1	127.0.0.1	TCP	44 27835 → 10089 [ACK] Seq=26 Ack=37 Win=2161152 Len=0

四次挥手是同理的，当用户quit之后断开连接，在wireshark中也会有直观的显示。

指导教师评语：

日期：