华东师范大学数据学院上机实践报告

Computer Network & Coding Lab 4

课程名称： 计算机网络	年级：2023	上机实践成绩：
指导老师：张召	姓名：陈子谦	
上机实践名称：DNS、TCP、UDP协议分析实验	学号：10235501454	上机实践日期：

一、题目要求

具体题目在每个小点下方展示

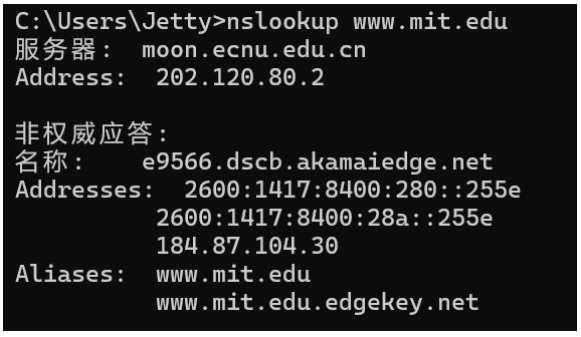
二、功能实现情况

1.DNS 协议分析

Task1

> 运行 nslookup 来确定一个国外大学 (www.mit.edu)的IP地址以及其权威 DNS服务器，请在实验报告中附上操作截图并详细分析返回信息内容

1.运行`nslookup www.mit.edu`后得到如下图所示输出：



- 图片显示查询请求发送到本地DNS服务器`moon.ecnu.edu.cn` (IP: `202.120.80.2`)
- 返回的是一个 `非权威应答`，表明该信息来自本地 DNS 服务器的缓存或其他非直接负责 `mit.edu` 的服务器，
- `名称` 和 `Aliases` 部分清楚地显示，`www.mit.edu` 是一个别名 (Alias)，它指向了 `e9566.dscb.akamaiedge.net` (通过中间别名 `www.mit.edu.edgekey.net`)。这确认 `www.mit.edu` 使用了 **Akamai** 的 **内容分发网络 (CDN)** 服务
- `Addresses` 部分列出了与 `e9566.dscb.akamaiedge.net` 关联的实际 IP 地址，可以得到mit的IP地址为`184.87.104.30(IPV4)`

2.随后运行`nslookup -type=NS mit.edu`查询mit的权威DNS服务器，得到下图结果：

```
C:\Users\Jetty>nslookup -type=NS mit.edu
服务器:  moon.ecnu.edu.cn
Address:  202.120.80.2

非权威应答:
mit.edu nameserver = use5.akam.net
mit.edu nameserver = usw2.akam.net
mit.edu nameserver = ns1-173.akam.net
mit.edu nameserver = asia1.akam.net
mit.edu nameserver = asia2.akam.net
mit.edu nameserver = ns1-37.akam.net
mit.edu nameserver = eur5.akam.net
mit.edu nameserver = use2.akam.net
```

查询了 `mit.edu` 域名的 NS (Name Server) 记录。NS 记录列出了负责该域名的所有权威 DNS 服务器，根据图片信息可以得到mit的权威服务器为`Akamai.net`所提供

Task2

> **运行 nslookup，使用task1中一个已获得的 DNS 服务器，来查询google服务器 (www.google.com)的 IP 地址(可直接查询)，请在实验报告中附上操作截图并详细分析返回信息内容。**

因为Google服务器在`moon.ecnu.edu.cn`的DNS服务器上完全没有记录，因此使用114公共DNS服务器查询，输入`nslookup www.google.com 114.114.114.114`得到如下图所示的结果：

```
C:\Users\Jetty>nslookup www.google.com 114.114.114.114
服务器:  public1.114dns.com
Address:  114.114.114.114

非权威应答:
名称:    www.google.com
Addresses:  2001::1
          199.16.158.12
```

图片显示IP地址`114.114.114.114`所对应的DNS服务器域名为`public1.114dns.com`，而显然114公共DNS服务器并不是Google的权威服务器，它随后通过向Google的权威服务器查询得到`199.16.158.12`的IP地址，不过这并不是Google的主要服务IP段，猜测这是Google的CDN

Task3

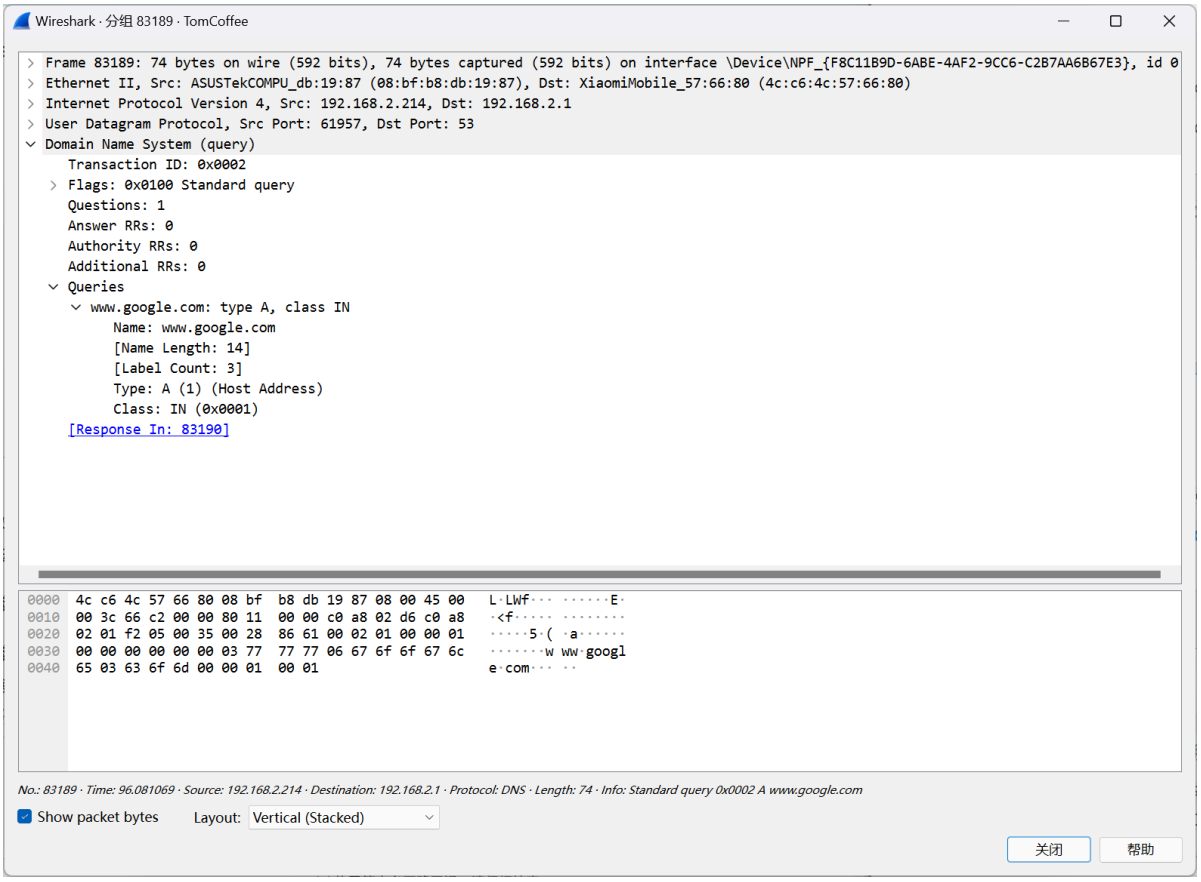
> **根据Wireshark抓取的报文信息，分别分析DNS查询报文和响应报文的组成结构，参考上面的报文格式指出报文的每个部分（如，头部区域等），请将实验结果附在实验报告中。**

- **报文格式**

DNS只有两种报文：查询报文、响应报文，两者有着相同格式，如下：

DNS查询报文

Wireshark 抓取的对 `www.google.com` 的 DNS 查询报文详细信息截图：



1.头部（Header Section）

- **Transaction ID (标识):** `Transaction ID: 0x8082`。这是一个 16 位的标识符，用于匹配查询报文和响应报文。客户端发送查询时设置，服务器在响应时会带回相同的 ID。
- **Flags (标记):** `Flags: 0x0100 Standard query`。这些标志位指示了报文的类型（查询或响应）、操作码、以及其他控制信息。`0x0100` 表示这是一个标准的查询报文 (Standard query)。
- **Questions (查询记录数):** `Questions: 1`。这表明查询部分的记录数为 1，即本次报文中包含一个待查询的问题。

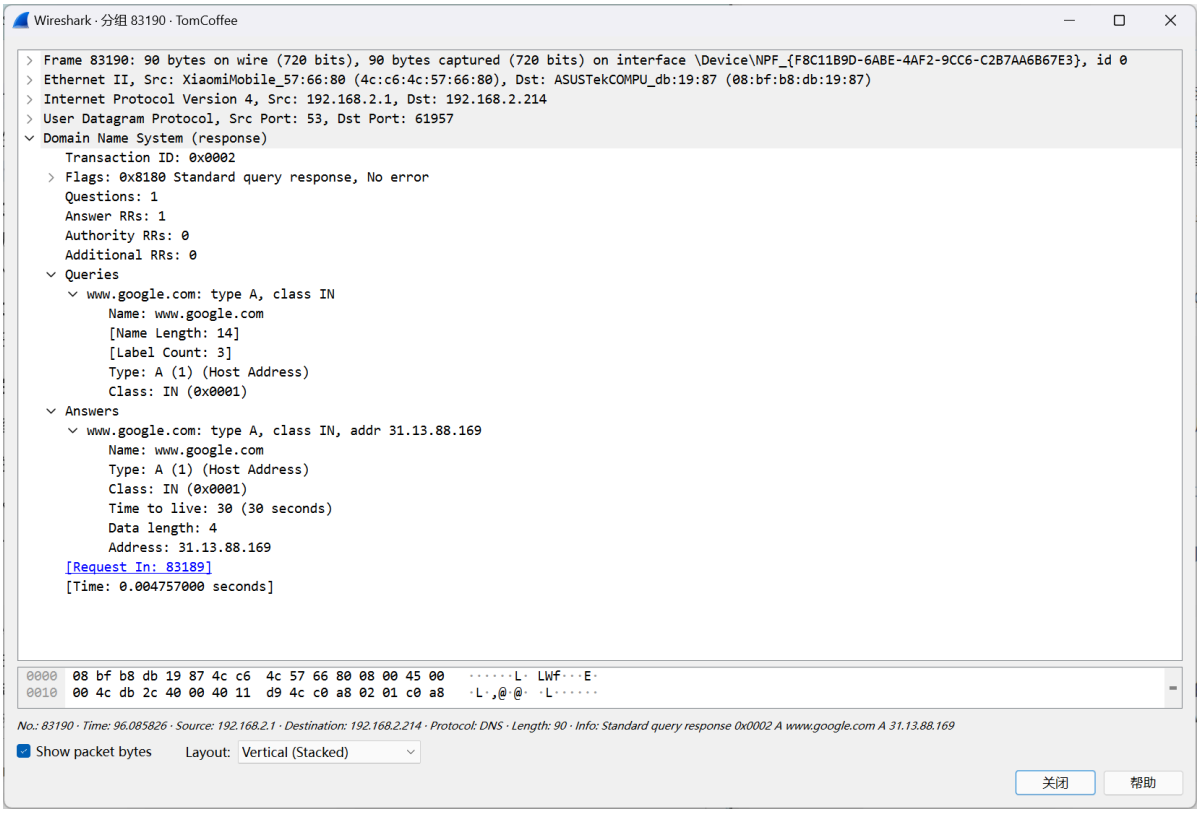
- **Answer RRs (应答记录数):** `Answer RRs: 0`。在查询报文中，应答记录数为 0
- **Authority RRs (授权记录数):** `Authority RRs: 0`。在查询报文中，授权记录数为 0
- **Additional RRs (附加记录数):** `Additional RRs: 0`。在查询报文中，附加记录数为 0

2.查询部分 (Question Section)

在 Wireshark 截图的 `Queries` 部分显示了查询的详细内容，从图中可以知道QS部分氛围三个字段

- **Name (查询名):** `Name: www.google.com`。这是客户端想要查询的域名。
- **Type (查询类型):** `Type: A (1) (Host Address)`。这是客户端请求的资源记录类型。`A` 类型表示查询域名的 IPv4 地址。其他常见类型还有 AAAA (IPv6 地址)、NS (名称服务器)、SOA (起始授权记录)、CNAME (规范名称) 等。
- **Class (查询类):** `Class: IN (0x0001)`。表示互联网 (IN) 数据。几乎所有的 DNS 查询都使用这个类。

DNS响应报文



1.头部（Header Section）

- **Transaction ID (标识):** `Transaction ID: 0x8082`。这个 ID 与之前的查询报文中的 ID 相同，用于匹配本次响应是针对哪一个查询。
- **Flags (标记):** `Flags: 0x8180 Standard query response, no error`。
  - `Response` (响应) 位被设置 (图中 `0x8180` 的最高位是 1)，表示这是一个响应报文，而非查询报文
  - `Standard query response` (标准查询响应) 指示了响应的类型
  - `no error` (无错误) 指示本次查询成功，没有发生 DNS 错误
- **Questions (查询记录数):** `Questions: 1`。重复了原始查询中的问题数量。
- **Answer RRs (应答记录数):** `Answer RRs: 1`。**这是与查询报文的主要区别之一。它表示响应报文的响应部分**中包含了多少个资源记录。这里是 1，说明找到了一个应答记录。

- **Authority RRs (授权记录数):** `Authority RRs: 0`。表示授权部分没有记录。
- **Additional RRs (附加记录数):** `Additional RRs: 0`。表示额外部分没有记录。

2.查询部分 (Question Section)

响应报文中通常会包含与原始查询报文完全相同的查询问题部分，以便客户端核对本次响应是对应哪个查询。因此此处与查询报文一样

3.响应部分 (Answer Section)

- **Name (查询名):** `Name: www.google.com`。此记录所属的域名。
- **Type (查询类型):** `Type: A (1) (Host Address)`。此记录的类型，这里是 A 记录，即 IPv4 地址记录。
- **Class (查询类):** `Class: IN (0x0001)`。表示互联网 (IN) 数据。几乎所有的 DNS 查询都使用这个类。
- **Time to live (TTL):** `Time to live: 30 seconds`。该记录可以被缓存的时间（单位：秒）。TTL 到期后，缓存该记录的 DNS 服务器需要重新查询获取最新值。
- **Data length (数据长度):** `Data length: 4`。后续资源数据字段的长度，对于 IPv4 地址 (A 记录)，长度是 4 字节。
- **Address (地址 / 资源数据):** `Address: 31.13.88.169`。这是 A 记录的资源数据字段，即 `www.google.com` 解析到的 IPv4 地址。这是客户端最终需要的信息。

4.授权部分 (Authority Section)

`Authority RRs: 0`，所以授权部分为空。这个部分通常在响应非权威时出现，列出负责被查询域名的权威 DNS 服务器的 NS 记录。

5.额外部分 (Additional Section)

`Additional RRs: 0`，所以额外部分为空。这个部分通常包含一些可能对客户端有帮助的附加信息，例如，如果在授权部分列出了权威服务器的域名，这里可能会包含这些服务器的 A 记录（IP 地址），以便客户端可以直接联系它们。

Task4

> **基于task3中得到的查询和响应报文进行分析，试问这里的查询是什么“Type"的，查询消息是否包含任何“answers"？试问这里的响应消息提供了多少个“answers”，这些“answers”具体包含什么？请将实验结果附在实验报告中。**

1.本次查询是什么 "Type" 的?

查询的类型是 **A 记录查询**。

2.查询消息是否包含任何 "answers"?

查询消息本身**不包含**任何“answers”

3.这里的响应消息提供了多少个 "answers"?

`Answer RRs: 1`表明响应报文的响应部分包含 **1 个**资源记录。

4.这些 "answers" 具体包含什么?

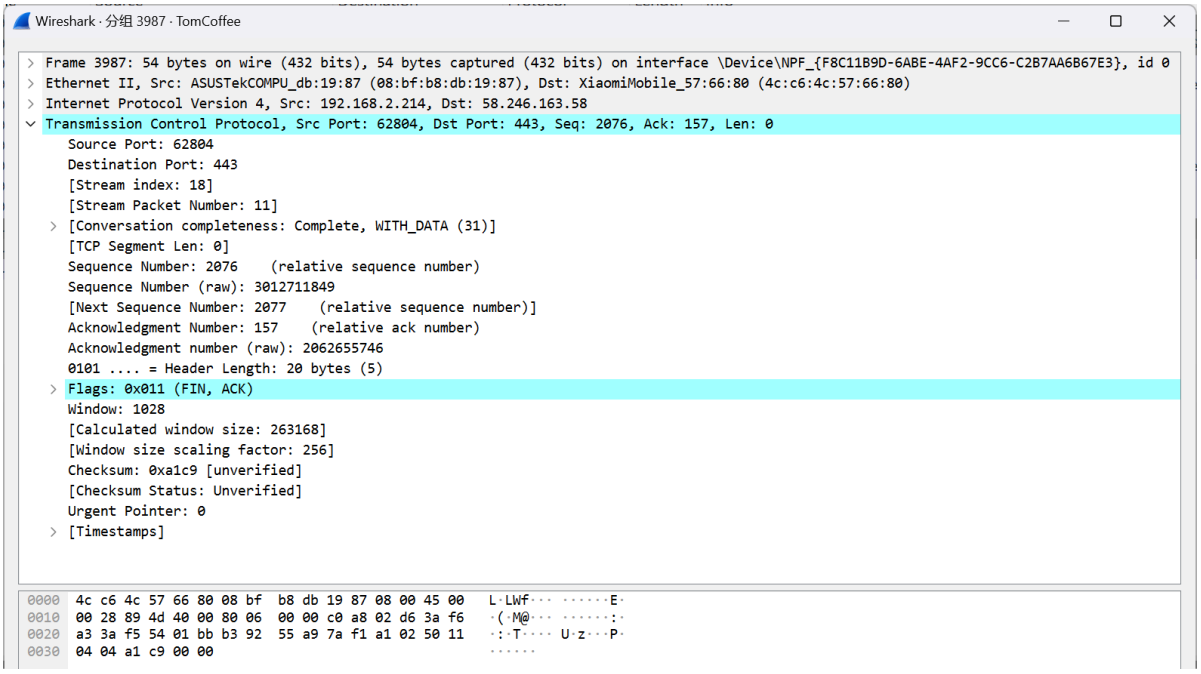
这 1 个“answer”是一个完整的 A 记录资源记录。它具体包含了 `www.google.com` 这个域名对应的 **IPv4 地址** (`31.13.88.169`)，以及该记录的类型 (A)、类 (IN)、生存时间 (TTL 为 30 秒) 和数据长度 (4 字节)。这个资源记录就是对客户端“查询 `www.google.com` 的 A 记录”这个问题的回答。

2.TCP 协议分析

Task1

> **利用Wireshark抓取一个TCP数据包，查看其具体数据结构和实际的数据（要求根据报文结构正确标识每个部分），请将实验结果附在实验报告中。**

图 3-29　TCP 报文段结构



头部 (Header)

- **源端口号 (Source Port):** `Source Port: 62884`。这是发送该 TCP 报文的应用程序使用的端口号
- **目的端口号 (Destination Port):** `Destination Port: 443`。这是该 TCP 报文要发送到的目标应用程序端口号
- **序号 (Sequence Number):** `Sequence Number: 2076 (relative sequence number)`。这是该报文段携带的数据的第一个字节在整个字节流中的序号（Wireshark 默认显示相对序号）

- **确认号 (Acknowledgment Number):** `Acknowledgment Number: 157 (relative ack number)`。这是发送方期望收到对方下一个报文段的第一个字节的序号，用于确认已收到对方的数据
- **首部长度 (Header Length):** `Header Length: 20 bytes (5)`。表示 TCP 头部的长度，单位是 4 字节（32 比特字）。`5` 表示头部有 5 个 4 字节的字段，总长 20 字节。TCP 头部最小为 20 字节
- **标志 (Flags):** `Flags: 0x011 (FIN, ACK)` 这些标志位控制 TCP 连接的状态和行为。`0x011` 的二进制是 `000010001`，其中倒数第二个位 (ACK) 和倒数第一个位 (FIN) 被设置为 1。
  - `FIN`: Finish Flag，表示发送方已完成数据发送，请求终止连接。
  - `SYN`: Synchronize Flag，用于建立连接时同步序列号。
  - `RST`: Reset Flag，用于异常终止连接。
  - `PSH`: Push Flag，提示接收端尽快将数据交给应用层。
  - `ACK`: Acknowledgment Flag，表示确认号字段有效。
  - `URG`: Urgent Flag，表示紧急数据指针有效。
  - `ECE`, `CWR`：用于拥塞控制。
  - 截图中的报文带有 FIN 和 ACK 标志，表明这是一个在连接终止阶段发送的报文。
- **接收窗口 (Window):** `Window: 263168`。这是发送方告知对方自己还有多少缓冲区空间可以接收数据，用于流量控制。
- **因特网检验和 (Checksum):** `Checksum: 0x0c9c [Unverified]`。用于校验 TCP 头部和数据的完整性。
- **紧急数据指针 (Urgent Pointer):** `Urgent Pointer: 0`。在 URG 标志设置时有效，指向紧急数据的最后一个字节的相对偏移量。这里为 0，表示无紧急数据。对应格式图中的"紧急数据指针"。
- **保留/未用 (Reserved/Unused):** Flags 部分下方显示了 0x011，其中包含一些保留位，必须为 0。

选项 (Options)

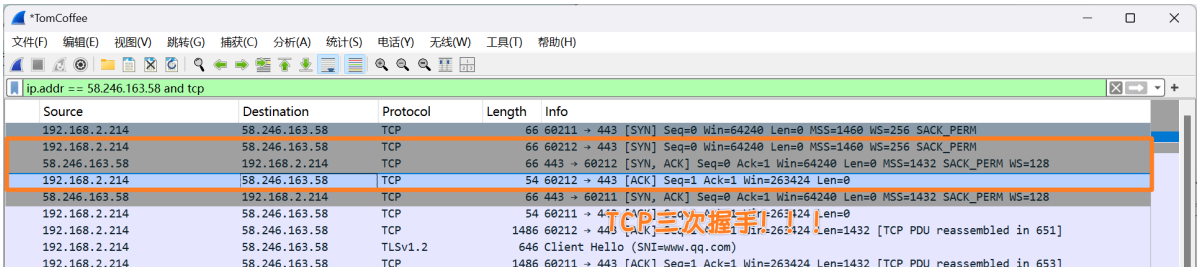
- 位于固定头部后面，长度可变，但必须是 4 字节的整数倍。如果头部长度大于 20 字节，则包含选项。
- `Header Length: 20 bytes (5)` 表明该报文的头部长度正好是最小长度 20 字节，因此**不包含选项**。

数据 (Data)

- 这是 TCP 报文段实际承载的应用层数据载荷（如 HTTP 请求/响应体、加密后的 TLS 数据等）。它紧跟在 TCP 头部（包括选项，如果存在的话）之后。
- `TCP Segment Len: 0` 以及 Info 字段中的 `Len: 0` 表明，虽然这是一个 TCP 报文，但它**不包含任何应用层数据载荷**。这是一个纯控制报文 (FIN/ACK)。
- **在包含数据的 TCP 报文中，数据部分会在 Wireshark 截图的 TCP 协议详情下方显示为原始的应用层数据，或者如果 Wireshark 能够解析，会显示为应用层协议（如 HTTP, TLSv1.2 Application Data）的详细内容。** 截图底部十六进制数据区域，前 20 个字节是 TCP 头部的内容，如果报文包含数据，数据会紧随其后显示。

Task2

> **根据TCP三次握手的交互图和抓到的TCP报文详细分析三次握手过程，请将实验结果附在实验报告中。**



这连续三个报文构成了标准的 TCP 三次握手过程：

第一次握手 (SYN):

客户端 (`192.168.1.214`) 发送一个 SYN 报文到服务器 (`58.248.163.58`) 的 443 端口 (HTTPS 常用端口)，请求建立连接。报文中包含客户端的初始序列号 (Seq=0)。

第二次握手 (SYN-ACK):

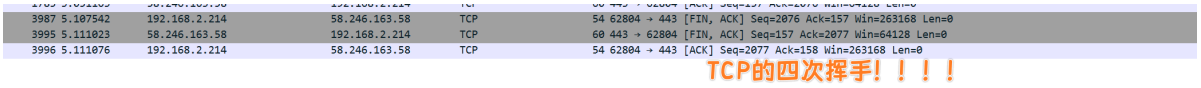
服务器接收到 SYN 后，发送一个 SYN-ACK 报文作为回应。报文中包含服务器的初始序列号 (Seq=0) 并确认收到客户端的 SYN (Ack=1，表示期望收到客户端序列号 1)。

第三次握手 (ACK):

客户端接收到 SYN-ACK 后，发送 ACK 报文作为最终确认。报文中确认收到服务器的 SYN (Ack=1，表示期望收到服务器序列号 1)

Task3

> **根据TCP四次挥手的交互图和抓到的TCP报文详细分析四次挥手过程，请将实验结果附在实验报告中。**



在这个截图里，四次挥手体现为以下序列：

1. 客户端发送 FIN-ACK (报文 3987)
2. 服务器发送 FIN-ACK (报文 3995) （这里服务器的 ACK 和 FIN 合并了）
3. 客户端发送 ACK (报文 3996)

这个序列是 TCP 四次挥手在实际抓包中常见的一种形式。

报文 3987:

- **源:** `192.168.1.214` (客户端)
- **目的:** `58.248.163.58` (服务器)
- **信息:** `58443 > 443 [FIN, ACK] ...`
- **分析:** 这是客户端发出的第一个挥手报文。`[FIN, ACK]` 标志表示客户端已经没有数据要发送了，希望关闭连接，同时确认收到了服务器之前发送的数据。

报文 3995：

- **源:** `58.248.163.58` (服务器)
- **目的:** `192.168.1.214` (客户端)
- **信息:** `443 > 58443 [FIN, ACK] ...`
- **分析:** 这是服务器发出的挥手报文。`[FIN, ACK]` 标志表示服务器也已经没有数据要发送了，同意关闭连接，同时确认收到了客户端的 FIN。在这个特定的抓包中，服务器将其对客户端 FIN 的 ACK 和自己的 FIN 合并到了一个报文发送。

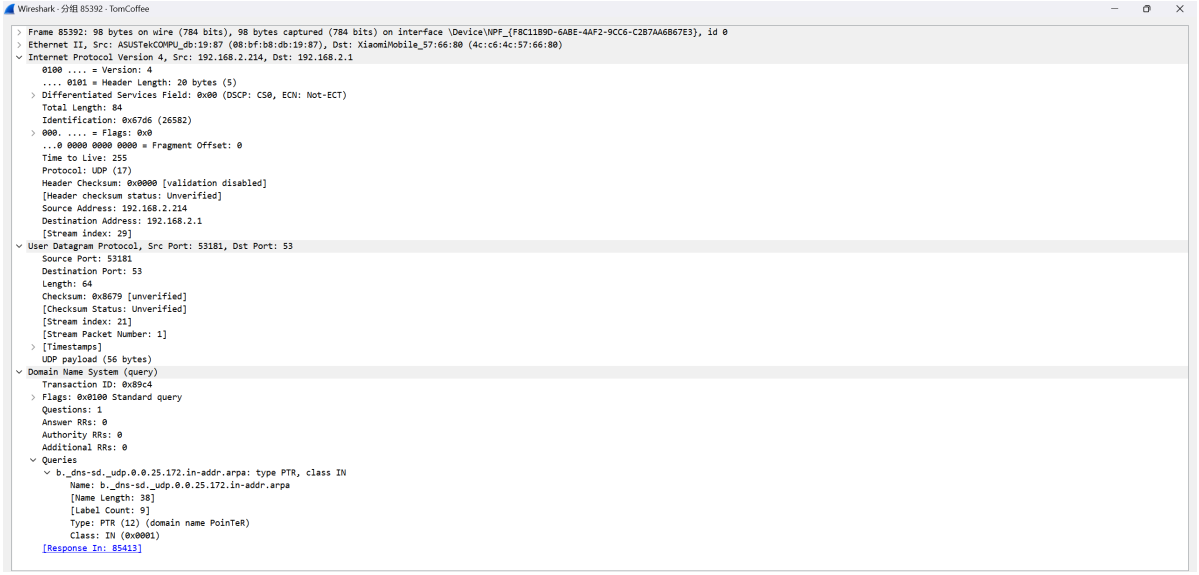
报文 3996：

- **源:** `192.168.1.214` (客户端)
- **目的:** `58.248.163.58` (服务器)
- **信息:** `58443 > 443 [ACK] ...`
- **分析:** 这是客户端发出的最后一个挥手报文。`[ACK]` 标志表示客户端确认收到了服务器的 FIN 报文。发送这个报文后，客户端会进入 TIME_WAIT 状态，等待一段时间以确保服务器收到这个 ACK。

3.UDP 协议分析

Task1

> **从跟踪中选择一个 UDP 数据包。从此数据包中，识别并确定 UDP 首部字段，请为这些字段命名并将实验结果附在实验报告中。**



1.源端口号 (Source Port)

- **Wireshark 显示：** `Source Port: 53181`
- **长度：** 16 比特 (2 字节)
- **作用：** 标识发送该 UDP 报文的应用程序的端口号。

2.目的端口号 (Destination Port)

- **Wireshark 显示：** `Destination Port: 53`
- **长度：** 16 比特 (2 字节)
- **作用：** 标识接收该 UDP 报文的应用程序的端口号。对于 DNS 服务，标准端口是 53。

3.长度 (Length)

- **Wireshark 显示：** 在 `User Datagram Protocol` 层的摘要行显示 `Length: 64`。
- **长度：** 16 比特 (2 字节)
- **作用：** 表示整个 UDP 数据报的长度，包括 8 字节的 UDP 头部和 UDP 数据（有效载荷）的长度。这里的 64 字节表示 UDP 头部 (8 字节) + DNS 数据 (56 字节) = 64 字节。

4.检验和 (Checksum)

- **Wireshark 显示：** `Checksum: 0x0b88 [validation disabled]` 或 `[Unverified]`。
- **长度：** 16 比特 (2 字节)
- **作用：** 用于校验 UDP 头部和数据的完整性。在 IPv4 中是可选的，但在 IPv6 中通常是强制的。`[validation disabled]` 可能表示 Wireshark 配置了不校验 UDP 检验和，或者发送方没有计算检验和（如果它是可选且未计算的话）。

Task2

> **UDP首部中的长度字段指的是什么，以及为什么需要这样设计？使用捕获的 UDP 数据包进行验证，请将实验结果附在实验报告中。**

1. UDP 头部中的长度字段指的是什么?

- UDP 头部中的长度字段是一个 16 比特的字段，它表示**整个 UDP 数据报的长度。**
- 这个长度包括 **UDP 头部（固定为 8 字节）的长度** 和 **UDP 数据（有效载荷）的长度。**
- UDP 数据报的最小长度是 8 字节（只有头部没有数据）。

2. 为什么需要这样设计（为什么需要长度字段）？

- **接收端处理需要：** 接收端的 UDP 模块需要知道从 IP 层接收到的数据包中，UDP 头部在哪里结束以及 UDP 数据在哪里结束，以便正确地提取数据并将其交给正确的应用程序（通过目的端口号）。长度字段明确指明了整个 UDP 数据报的大小，接收端可以根据这个长度准确地分割报文头和数据。
- 与 IP 层长度字段配合或独立使用：

  尽管 IP 头部（在 IPv4 中是"总长度"，在 IPv6 中是"载荷长度"）也包含了整个 IP 数据包（或载荷）的长度信息，但这并不能直接替代 UDP 的长度字段。

  - UDP 的长度字段是 UDP 协议本身的属性，使得 UDP 层可以独立处理其数据报的长度。
  - 在计算 UDP 检验和（如果启用的话）时，需要包含伪头部、UDP 头部以及 UDP 数据，其中就用到了 UDP 的长度字段，这有助于校验数据报的完整性，包括长度信息本身的正确性。
  - 允许 UDP 报文的长度小于 IP 数据报的总长度（虽然不常见，但理论上 IP 层可以在 UDP 数据报后填充额外的数据，尽管规范通常要求两者长度一致）。UDP 长度字段确保接收端只处理属于 UDP 的部分。
- **数据部分长度可变：** UDP 是一种无连接协议，其数据部分的长度不是固定的。长度字段是唯一明确告诉接收端这个特定 UDP 数据报有多长的字段。

3. **使用捕获的 UDP 数据包进行验证**

```
User Datagram Protocol, Src Port: 53181, Dst Port: 53
    Source Port: 53181
    Destination Port: 53
    Length: 64
    Checksum: 0x8679 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 21]
    [Stream Packet Number: 1]
  > [Timestamps]
    UDP payload (56 bytes)
```

UDP 头部中的长度字段显示为： `Length: 64` 字节。根据定义：UDP 数据报总长度 = UDP 头部长度 + UDP 数据（有效载荷）长度将获取的值代入：64 字节 = 8 字节 + 56 字节，计算结果：64 字节 = 64 字节与 Wireshark 显示的 UDP 长度字段的值一致。这验证了 UDP 头部中的长度字段确实指代的是 UDP 头部加上 UDP 数据部分的总长度。

Task3

> **UDP 有效负载中可包含的最大字节数是多少？请将实验结果附在实验报告中。**

UDP 头部包含一个 16 比特的“长度”字段，这个字段表示 **整个 UDP 数据报的长度**，包括 8 字节的 UDP 头部和其后所有数据的长度。因此，UDP 数据报的总长度（头部 + 数据）的最大值是 **65535 字节**

理论上，根据 UDP 头部中长度字段的定义，UDP 有效负载中可包含的最大字节数是 **65535 字节 - 8 字节 = 65527 字节**

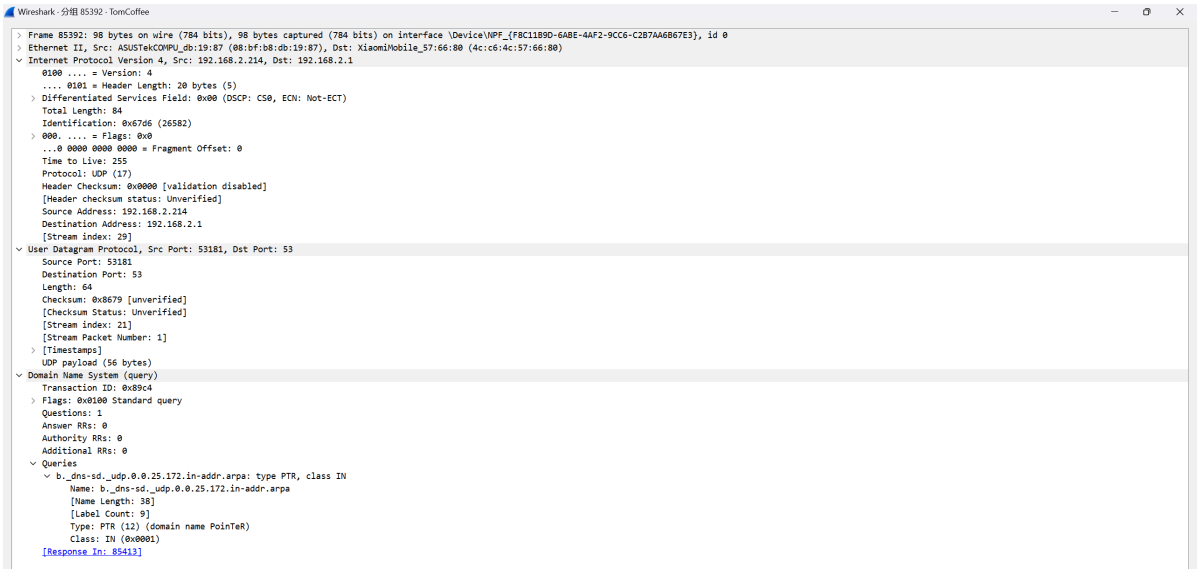
使用task3中的报文进行验证。UDP Length = 64 字节，UDP Payload = 56 字节，UDP Header 固定为 8 字节

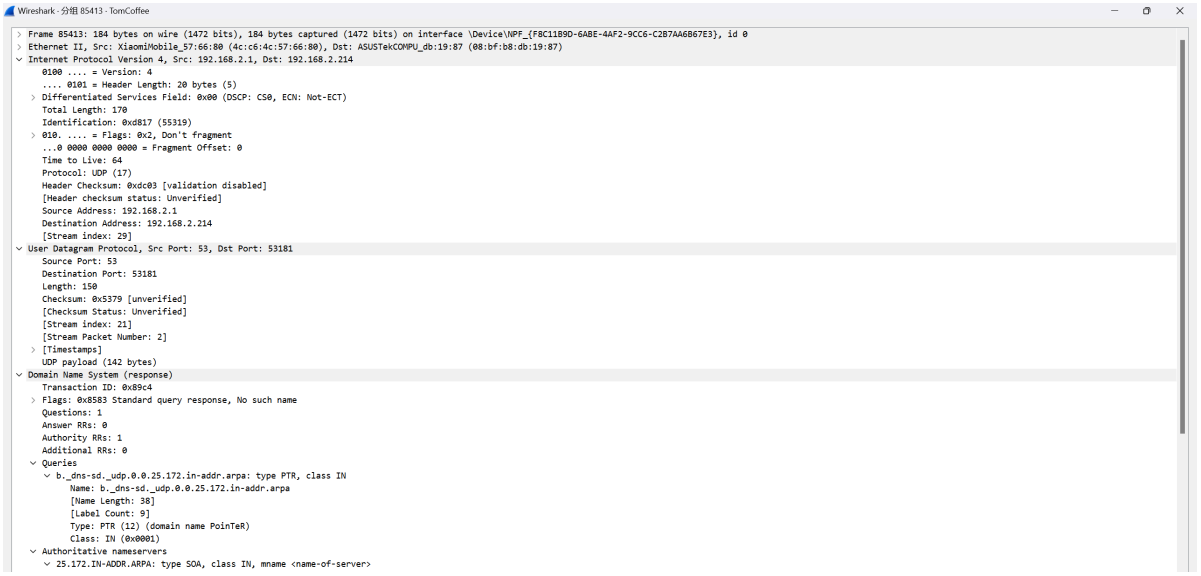
64 = 8 + 56

关系成立~

Task4

> **观察发送 UDP 数据包后接收响应的 UDP 数据包，这是对发送的 UDP 数据包的回复，请描述两个数据包中端口号之间的关系。(提示：对于响应 UDP 目的地应该为发送 UDP 包的地址。）请将实验结果附在实验报告中。**





上图为发送UDP包，下图为接受UDP包

发送 UDP 数据包 (报文 85302)

- **源 IP 地址 (Source IP):** `192.168.1.214` (客户端)
- **目的 IP 地址 (Destination IP):** `192.168.1.1` (服务器，本地 DNS 解析器)
- **源端口号 (Source Port):** `53181` (客户端使用的临时端口)
- **目的端口号 (Destination Port):** `53` (服务器的 DNS 服务端口)

接收响应 UDP 数据包 (报文 85413)

- **源 IP 地址 (Source IP):** `192.168.1.1` (服务器，本地 DNS 解析器)
- **目的 IP 地址 (Destination IP):** `192.168.1.214` (客户端)
- **源端口号 (Source Port):** `53` (服务器的 DNS 服务端口)
- **目的端口号 (Destination Port):** `53181` (客户端发送查询时使用的源端口)

结果分析：

通过对比发送（查询）UDP 报文和接收（响应）UDP 报文的端口号，可以清楚地看到它们之间的对应关系：

1. 发送报文的源端口号 (`53181`) 变为了接收报文的目的端口号 (`53181`)。
2. 发送报文的目的端口号 (`53`) 变为了接收报文的源端口号 (`53`)。

这种关系与 IP 地址的关系是一致的：发送报文的源 IP (`192.168.1.214`) 变为了接收报文的目的 IP (`192.168.1.214`)；发送报文的目的 IP (`192.168.1.1`) 变为了接收报文的源 IP (`192.168.1.1`)。

三、总结

本次计算机网络上机实践，围绕 DNS、TCP、UDP 这三个核心网络协议，综合运用 `nslookup` 命令行工具和 Wireshark 抓包分析手段，深入探究了它们的工作原理和报文结构。

在 DNS 协议分析部分，通过 `nslookup` 成功查询了国外大学和常用网站的 IP 地址，并学会了如何识别域名背后的 CDN 服务（如 Akamai）。通过查询 SOA 和 NS 记录，明确了域名的权威 DNS 服务器所在，理解了权威与非权威应答的区别以及 DNS 委托的概念。进一步，利用 Wireshark 详细解剖了 DNS 查询和响应报文的头部和查询/响应部分，识别了各字段的含义，并明确了查询类型（如 A 记录）以及响应报文携带的 answers 内容。

TCP 协议分析部分是本次实践的重点之一。通过 Wireshark 抓取并分析实际的 TCP 数据包，我们学习了 TCP 报文段头部的详细结构和各字段（包括源/目的端口、序号、确认号、各种标志位、窗口大小等）的作用。更重要的是，在抓包数据中成功识别并分析了 TCP 连接建立过程中的“三次握手”（SYN, SYN-ACK, ACK 序列）和连接终止过程中的“四次挥手”（FIN/ACK 序列），加深了对 TCP 可靠性、面向连接特性的理解。

UDP 协议分析部分则着重于理解这一无连接协议的特点。通过 Wireshark 分析 UDP 报文，掌握了其简洁的头部结构，识别了源/目的端口号、长度和检验和字段。对 UDP 头部长度字段的深入分析，明确了其表示的是 UDP 头部加有效载荷的总长度，并通过捕获的报文验证了这一关系，理解了其在接收端处理数据报时的必要性。最后，通过对比一对 UDP 查询和响应报文，清晰地展示了端口号之间（以及 IP 地址之间）的对调关系，理解了这是无连接协议实现请求-响应模式、将响应正确交付给源应用程序的关键所在。

总而言之，本次实践活动将理论知识与实际抓包分析相结合，使我们能够直观地看到网络协议在真实通信中的具体表现形式，加深了对 DNS 解析流程、TCP 连接管理机制以及 UDP 数据传输特点的理解。通过亲手操作和分析报文细节，不仅掌握了 `nslookup` 和 Wireshark 这两个重要的网络分析工具，也巩固了计算机网络课程中关于应用层、传输层和网络层核心协议的知识，是一次富有成效的学习经历。