第 3 次课程报告

同济大学电子与信息工程学院

计算机科学与技术系

计算机网络课程报告

林日中

1951112@TONGJI.EDU.CN

2022 年 6 月

目录

在本次课程报告中，我们利用抓包软件 Wireshark 分析以太网 MAC 帧、无线局域网 MAC 帧、IP 协议、TCP 协议和 HTTP 协议。

1 以太网 MAC 帧

在以太网链路上的数据包称作以太帧。

以太帧起始部分由前导码和帧开始符组成；后面紧跟着一个以太网报头，以 MAC 地址说明目的地址和源地址；帧的中部是该帧负载的包含其他协议报头的数据包（例如 IP 协议）；以太帧由一个 32 位冗余校验码结尾。它用于检验数据传输是否出现损坏。

1.1 捕获以太网 MAC 帧实验步骤

1. 打开 Wireshark，进入捕获界面。
2. 将 Wireshark 过滤器设为 `http`。
3. 在浏览器访问 `http://www.baidu.com`。
4. 在 Wireshark 的捕获界面中点击目的地址为 `220.181.38.148` 的帧，可以看到捕获到的以太网 MAC 帧，如图 1.1 所示。

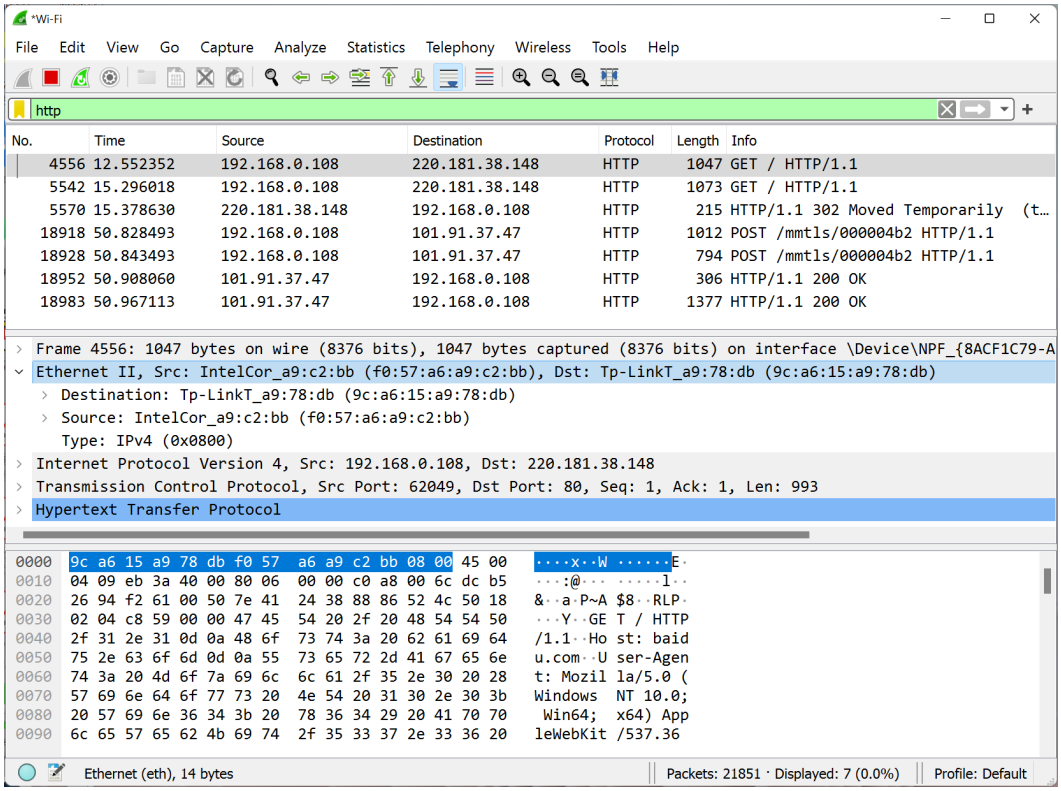

图 1.1: 捕获到的以太网 MAC 帧

1.2 以太网 MAC 帧简述

如图 1.2 所示，以太网 MAC 帧由 6 字节目的 MAC 地址、6 字节源 MAC 地址、2 字节以太网帧数据类型字段组成。

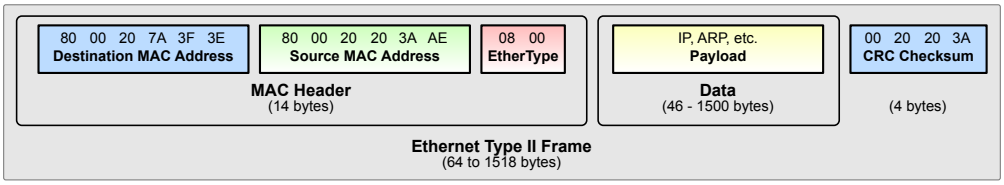

图 1.2: 以太网 II 的帧格式

1.3 以太网 MAC 帧分析

捕获到的以太网 MAC 帧内容为：

```
9c a6 15 a9 78 db f0 57 a6 a9 c2 bb 08 00
```

其中，目的 MAC 地址为 `9c a6 15 a9 78 db`，源 MAC 地址为 `f0 57 a6 a9 c2 bb`，以太网帧数据类型为 `08 00`，即 IP 协议。与图 1.2 所示的格式相吻合。

2 无线局域网 MAC 帧

无线局域网（英语：Wireless LAN，缩写 WLAN）是不使用任何导线或传输电缆连接的局域网，而使用无线电波或电场与磁场作为数据传送的介质，传送距离一般只有几十米。无线局域网的主干网路通常使用有线电缆，无线局域网用户通过一个或多个无线接取器接入无线局域网。无线局域网现在已经广泛的应用在商务区，大学，机场，及其他需要无在线网的公共区域。

无线局域网最通用的标准是 IEEE 定义的 802.11 系列标准，以及中华人民共和国从 2003 年开始陆续颁布的一系列国家标准 WAPI。

2.1 捕获无线局域网 MAC 帧实验步骤

由于捕获 802.11 帧需要设置网卡为监控模式（Monitor Mode），而我的网卡不支持此操作，不具备捕获 802.11 无线协议报文的条件。因此，采用《计算机网络——自顶向下方法》教材提供的爬取结果（`http://www-net.cs.umass.edu/wireshark-labs/wireshark-traces.zip`）。

捕获到的无线局域网 MAC 帧如图 2.1 所示。

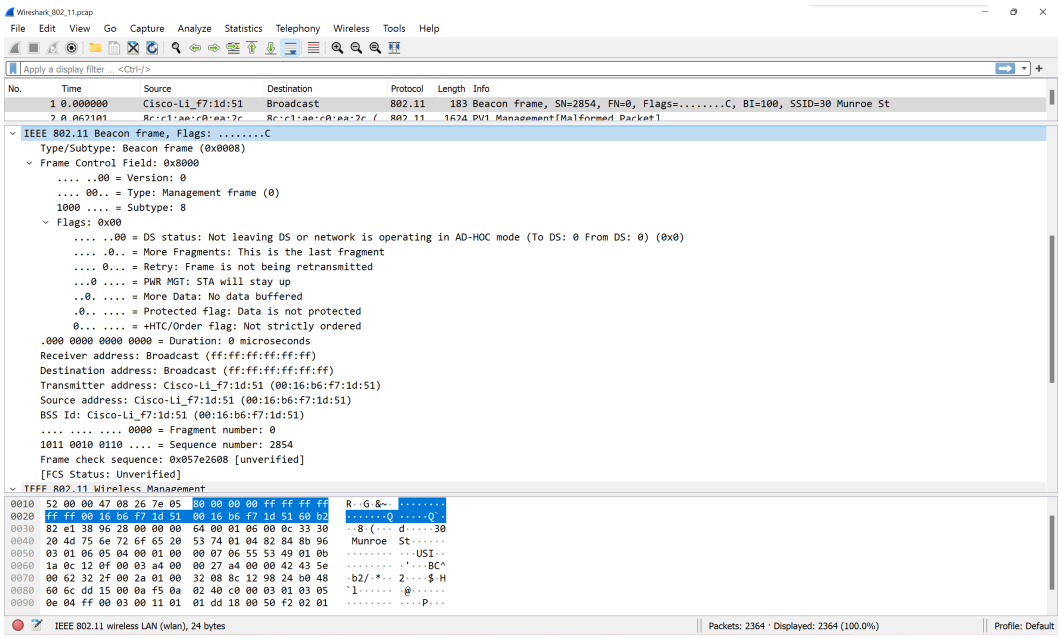

图 2.1: 捕获到的无线局域网 MAC 帧

2.2 无线局域网 MAC 帧简述

IEEE 802.11 MAC 帧格式如图 2.2 所示。

MAC 头的前两个字节形成一个帧控制字段，规定了帧的形式和功能。这个帧控制字段被细分为以下子字段。

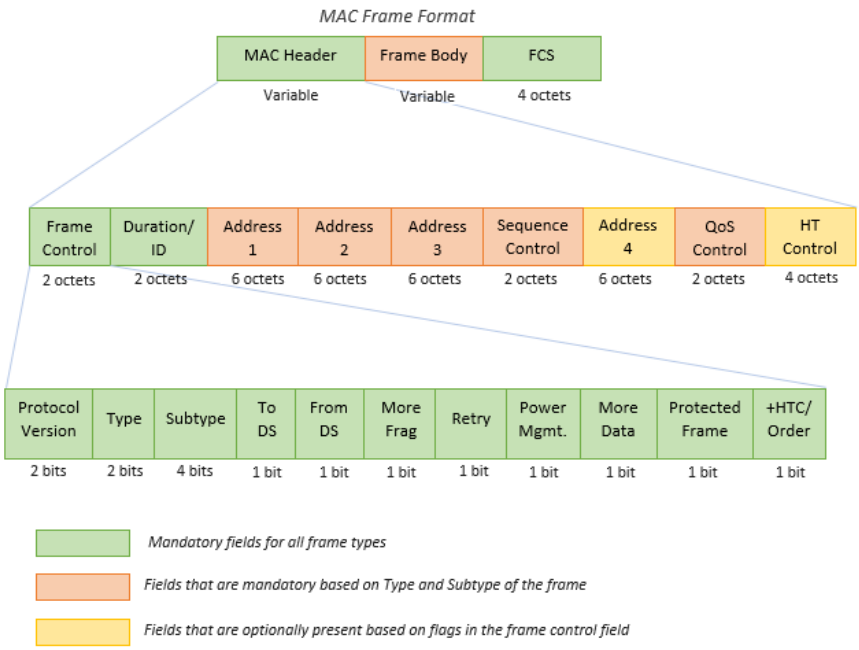

图 2.2: IEEE 802.11 MAC 帧格式

- **协议版本**：代表协议版本的两个比特。当前使用的协议版本为 0，其他值保留给未来使用。
- **类型**：两个比特，识别 WLAN 帧的类型。控制、数据和管理是 IEEE 802.11 中定义的各种帧类型。
- **子类型**：四个比特，提供帧之间的额外区分。类型和子类型一起使用，以确定确切的帧。
- **ToDS 和 FromDS**：每个都是一个比特的大小。它们表示一个数据帧是否前往一个分配系统。控制和管理帧将这些值设置为零。所有的数据帧都将设置这些位中的一个。然而，独立基本服务集（IBSS）网络内的通信总是将这些位设置为零。
- **更多片段**：当一个数据包被分成多个帧进行传输时，更多片段位被设置。除了数据包的最后一帧，每一帧都会设置这个位。
- **重试**：有时帧需要重传，为此有一个重试位，当一个帧被重新发送时被设置为 1。这有助于消除重复的帧。
- **电源管理**：该位表示帧交换完成后发送方的电源管理状态。接入点需要管理连接，绝不会设置省电位。
- **更多数据**：更多数据位用于缓冲分布式系统中收到的帧。接入点使用该位以方便处于省电模式的站点。它表示至少有一个帧是可用的，并针对所有连接的站。
- **受保护帧**：如果帧体被保护机制加密，如有线等效保密（WEP）、Wi-Fi 保护接入（WPA）或 Wi-Fi 保护接入 II（WPA2），则保护帧位被设置为 1 值。

- **顺序**：该位仅在采用“严格排序”传输方法时被设置。帧和片段并不总是按顺序发送，因为这会导致传输性能的下降。

接下来的两个字节是为持续时间 ID 字段保留的，表明该字段的传输需要多长时间，以便其他设备知道该信道何时可以再次使用。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）。

一个 802.11 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。

头部的其余字段为：

- **顺序控制字**段是一个两字节的部分，用于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。
- 一个可选的两字节的**服务质量控制字段**，存在于 QoS 数据帧中；它是在 802.11e 中加入的。

有效载荷或帧体字段大小可变，从 0 到 2304 字节，加上安全封装的任何开销，包含来自高层的信息。

帧检查序列（FCS）是标准 802.11 帧中的最后四个字节。通常被称为循环冗余检查（CRC），它允许对检索的帧进行完整性检查。当帧即将被发送时，FCS 被计算和附加。当一个站收到一个帧时，它可以计算出该帧的 FCS 并与收到的帧进行比较。如果它们相匹配，就可以认为该帧在传输过程中没有失真。

2.3 无线局域网 MAC 帧分析

捕获到的无线局域网 MAC 帧内容为：

- **FC**：
  - **协议版本**：0，占 2bit，
  - **类型**：0，占 2bit，
  - **子类型**：8，占 4bit；
  - **ToDS 和 FromDS**：00，占 2bit，
  - **更多片段**：0，占 1bit，
  - **重试**：0，占 1bit，
  - **电源管理**：0，占 1bit，
  - **更多数据**：0，占 1bit，
  - **受保护帧**：0，占 1bit，
  - **顺序**：0，占 1bit；

- **持续时间**：0ms，占 2 字节；
- **地址 1**：`ff ff ff ff ff ff`，占 6 字节；
- **地址 2**：`00 16 b6 f7 1d 51`，占 6 字节；
- **SC**：
  - **分片号**：0，占 4bit，
  - **序列号**：2854，占 12bit；
- **FCS**：0x057e2608，占 4 字节。

且与图 2.2 所示的无线局域网 MAC 帧结构相同。

3 IP 协议

网际协议（英语：Internet Protocol，缩写：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，任务仅仅是根据源主机和目的主机的地址来传送数据。

网际协议版本 4（英语：Internet Protocol version 4，缩写：IPv4，又称互联网通信协议第四版）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，直到 2011 年，IANA IPv4 位址完全用尽时，IPv6 仍处在部署的初期。

3.1 捕获 IPv4 报文实验步骤

1. 打开 Wireshark，进入捕获界面。
2. 将 Wireshark 过滤器设为 `http`。
3. 在浏览器访问 `http://www.baidu.com`。
4. 在 Wireshark 的捕获界面中点击目的地址为 `220.181.38.148` 的帧，可以看到捕获到的 IPv4 报文，如图 3.1 所示。

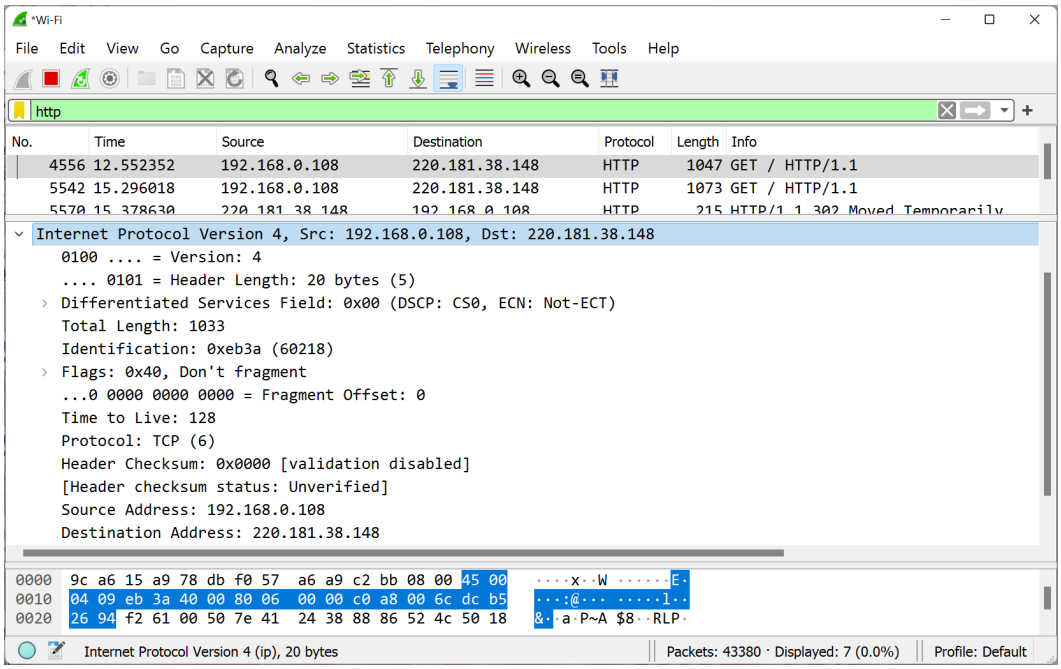

图 3.1: 捕获到的 IPv4 报文

3.2 IPv4 报文简述

IP 报文包含 IP 首部和数据部分，IPv4 报文格式如图 3.2 所示。

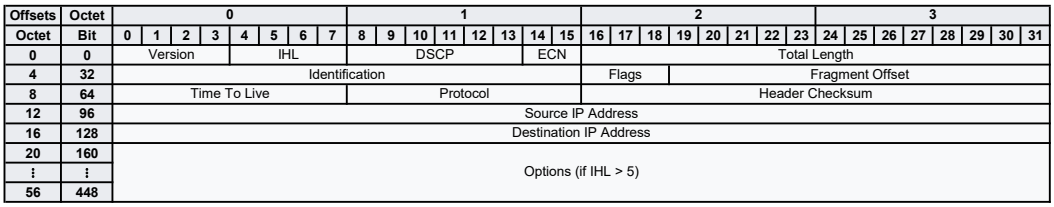

图 3.2: IPv4 报文格式

3.2.1 首部

IPv4 报文的首部包含 14 个字段，其中 13 个是必须的，第 14 个是可选的，并命名为“选项（Options）”字段。首部中的字段均以大端序包装，在以下的图表和讨论中，最高有效位（Most Significant bit）被标记为 0。

版本（Version） 版本字段占 4bit，通信双方使用的版本必须一致。对于 IPv4，字段的值是 4。

首部长度（Internet Header Length，IHL） 占 4bit，首部长度说明首部有多少 32 位字（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。这个字段的最小值是 5（二进制 0101），相当于 $5 * 4 = 20$ 字节，最大十进制值是 15。

差分服务代码点（Differentiated Services Code Point，DSCP） 占 6bit，最初被定义为服务类型字段，实际上并未使用，但 1998 年被 IETF 重定义为区分服务。只有在使用区分服务时，这个字段才起作用，在一般的情况下都不使用这个字段。例如需要实时数据流的技术会应用这个字段，一个例子是 VoIP。

显式拥塞通告（Explicit Congestion Notification，ECN） 允许在不丢弃报文的同时通知对方网络拥塞的发生。ECN 是一种可选的功能，仅当两端都支持并希望使用，且底层网络支持时才被使用。

全长（Total Length） 这个 16 位字段定义了报文总长，包含首部和数据，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 $216 - 1 = 65535$。

IP 规定所有主机都必须支持最小 576 字节的报文，这是假定上层数据长度 512 字节，加上最长 IP 首部 60 字节，加上 4 字节富裕量，得出 576 字节，但大多数现代主机支持更大的报文。当下层的数据链路协议的最大传输单元（MTU）字段的值小于 IP 报文长度时，报文就必须被分片。

标识符（Identification） 占 16 位，这个字段主要被用来唯一地标识一个报文的所有分片，因为分片不一定按序到达，所以在重组时需要知道分片所属的报文。每产生一个数据

报，计数器加 1，并赋值给此字段。一些实验性的工作建议将此字段用于其它目的，例如增加报文跟踪信息以协助探测伪造的源地址。

标志（Flags） 这个 3 位字段用于控制和识别分片，它们是：

- 位 0：保留，必须为 0；
- 位 1：禁止分片（Don't Fragment，DF），当 $DF = 0$ 时才允许分片；
- 位 2：更多分片（More Fragment，MF），$MF = 1$ 代表后面还有分片，$MF = 0$ 代表已经是最后一个分片。

如果 DF 标志被设置为 1，但路由要求必须分片报文，此报文会被丢弃。这个标志可被用于发往没有能力组装分片的主机。当一个报文被分片，除了最后一片外的所有分片都设置 MF 为 1。最后一个片段具有非零片段偏移字段，将其与未分片数据包区分开，未分片的偏移字段为 0。

分片偏移（Fragment Offset） 这个 13 位字段指明了每个分片相对于原始报文开头的偏移量，以 8 字节作单位。

存活时间（Time To Live，TTL） 这个 8 位字段避免报文在互联网中永远存在（例如陷入路由环路）。存活时间以秒为单位，但小于一秒的时间均向上取整到一秒。在现实中，这实际上成了一个跳数计数器：报文经过的每个路由器都将此字段减 1，当此字段等于 0 时，报文不再向下一跳传送并被丢弃，最大值是 255。常规地，一份 ICMP 报文被发回报文发送端说明其发送的报文已被丢弃。这也是 traceroute 的核心原理。

协议（Protocol） 占 8bit，这个字段定义了该报文数据区使用的协议。IANA 维护着一份协议列表，参见第 3.2.2 小节。

首部检验和（Header Checksum） 这个 16 位检验和字段只对首部查错，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，如果不一致，此报文将会被丢弃。重新计算的必要性是因为每一跳的一些首部字段（如 TTL、Flag、Offset 等）都有可能发生变化，不检查数据部分是为了减少工作量。

数据区的错误留待上层协议处理——用户数据报协议（UDP）和传输控制协议（TCP）都有检验和字段。此处的检验计算方法不使用 CRC。

源地址（Source address） 一个 IPv4 地址由四个字节共 32 位构成，此字段的值是将每个字节转为二进制并拼在一起所得到的 32 位值。例如，10.9.8.7 是

00001010000010010000100000000111

但因为 NAT 的存在，这个地址并不总是报文的真实发送端，因此发往此地址的报文会被送往 NAT 设备，并由它被翻译为真实的地址。

目的地址（Destination address） 与源地址格式相同，但指出报文的接收端。

选项（Options） 附加的首部字段可能跟在目的地址之后，但这并不被经常使用，从 1 到 40 个字节不等。请注意首部长度字段必须包括足够的 32 位字来放下所有的选项（包括任何必须的填充以使首部长度能够被 32 位整除）。当选项列表的结尾不是首部的结尾时，EOL（选项列表结束，0x00）选项被插入列表末尾。表 3.1 展示了选项段的字段。

字段	长度（位）	描述
备份	1	当此选项需要被备份到所有分片中时，设为 1。
类	2	常规的选项类别，0 为 “控制”，2 为 “查错和措施”，1 和 3 保留。
数字	5	指明一个选项。
长度	8	指明整个选项的长度，对于简单的选项此字段可能不存在。
数据	可变	选项相关数据，对于简单的选项此字段可能不存在。

表 3.1: 选项（Options）段的字段

3.2.2 数据

数据字段不是首部的一部分，因此并不被包含在首部检验和中。数据的格式在协议首部字段中被指明，并可以是任意的传输层协议。一些常见协议的协议字段值如表 3.2 所示。

协议字段值	协议名	缩写
1	互联网控制消息协议	ICMP
2	互联网组管理协议	IGMP
6	传输控制协议	TCP
17	用户数据报协议	UDP
41	IPv6 封装	ENCAP
89	开放式最短路径优先	OSPF
132	流控制传输协议	SCTP

表 3.2: 一些常见协议的协议字段值

3.3 IPv4 报文分析

捕获到的 IPv4 报文为：

```
45 00 04 09 eb 3a 40 00 80 06 00 00 c0 a8 00 6c dc b5 26 94
```

其中，

- **版本**为 4，占 4bit；
- **首部长度**为 $4 * 5 = 20$ 字节，占 4bit；
- **差分服务代码点**为 CS0，占 6bit；
- **显式拥塞通告**为 Not-ECT，占 2bit；
- **总长度**为 1033 字节，占 16bit；
- **标识符**为 `eb 3a`，占 16bit；
- **标志**为 Don't Fragment，即位 1 为 1，占 3bit；
- **分片偏移**为 0，占 13bit；
- **存活时间**为 $8 * 16 = 128$ 跳，占 8bit；
- **协议**为 `TCP`，对应表 3.2 中的字段值 6，占 8bit；
- **首部检验和**为 `00 00`，占 16bit；
- **源地址**为 `c0 a8 00 6c`，即 `192.168.0.108`，占 32bit；
- **目的地址**为 `dc b5 26 94`，即 `220.181.38.148`，占 32bit。

且与图 3.2 所示的 IPv4 报文结构相同。

4 TCP 协议

传输控制协议（英语：Transmission Control Protocol，缩写：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，由 IETF 的 RFC 793 定义。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。用户数据报协议（UDP）是同一层内另一个重要的传输协议。

在因特网协议族（Internet protocol suite）中，TCP 层是位于 IP 层之上，应用层之下的中间层。不同主机的应用层之间经常需要可靠的、像管道一样的连接，但是 IP 层不提供这样的流机制，而是提供不可靠的包交换。

应用层向 TCP 层发送用于网间传输的、用 8 位字节表示的数据流，然后 TCP 把数据流分割成适当长度的报文段（通常受该计算机连接的网络的数据链路层的最大传输单元（MTU）的限制）。之后 TCP 把结果包传给 IP 层，由它来透过网络将包传送给接收端实体的 TCP 层。TCP 为了保证不发生丢包，就给每个包一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的包发回一个相应的确认信息（ACK）；如果发送端实体在合理的往返时延（RTT）内未收到确认，那么对应的数据包就被假设为已丢失并进行重传。TCP 用一个校验和函数来检验数据是否有错误，在发送和接收时都要计算校验和。

4.1 捕获 TCP 报文实验步骤

1. 打开 Wireshark，进入捕获界面。
2. 将 Wireshark 过滤器设为 `http`。
3. 在浏览器访问 `http://www.baidu.com`。
4. 在 Wireshark 的捕获界面中点击目的地址为 `220.181.38.148` 的帧，可以看到捕获到的 TCP 报文，如图 4.1 所示。

4.2 TCP 报文简述

TCP 报文格式如图 4.2 所示。

源端口（16 位） 标明发送端口。

目的地端口（16 位） 识别接收端口。

序列号（32 位） 具有双重作用。

- 如果 SYN 标志被设置（1），那么这就是初始序列号。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。

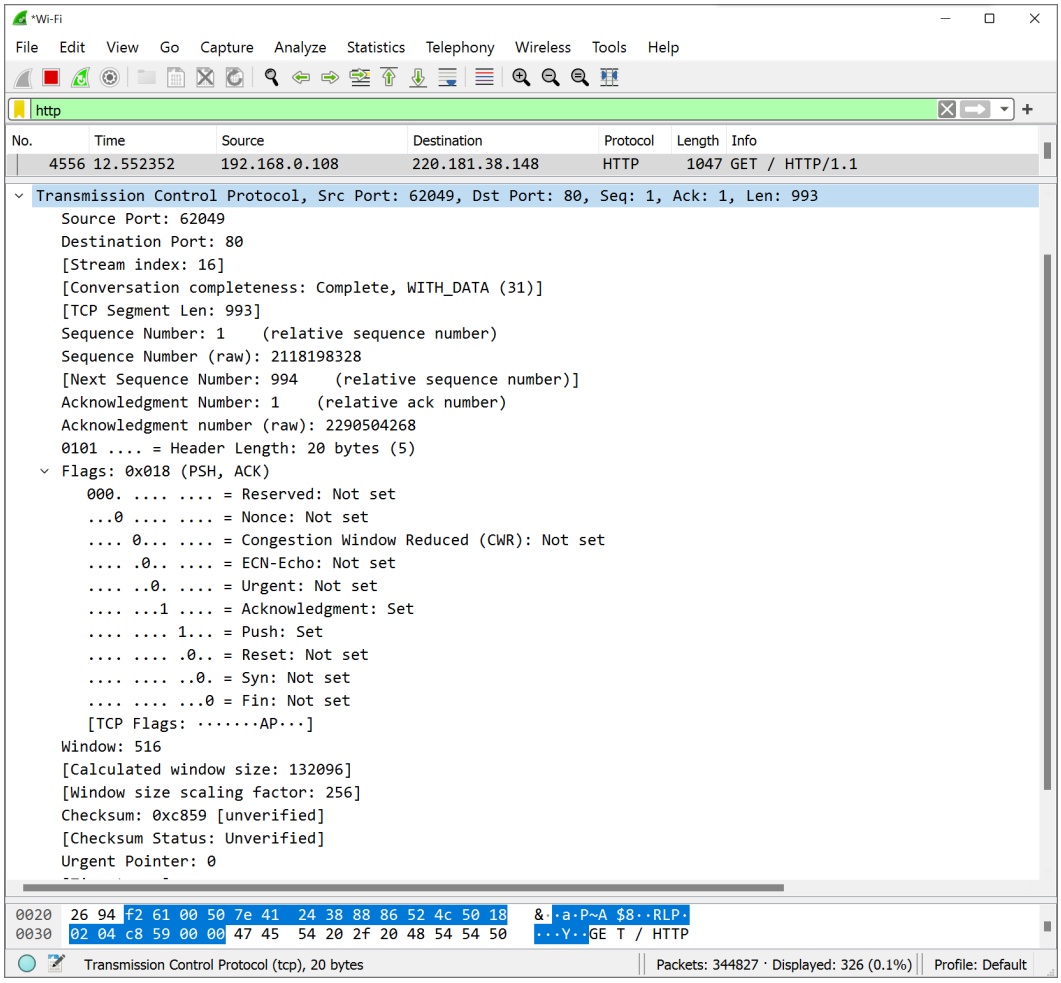

图 4.1: 捕获到的 TCP 报文

Offsets	Octet	0								1								2								3							
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20	160	Options (if data offset > 5. Padded at the end with "0" bits if necessary.)																															
60	480																																

图 4.2: TCP 报文格式

- 如果 SYN 标志被清除（0），那么这就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 位） 如果 ACK 标志被设置，那么这个字段的值就是 ACK 的发送者所期望的下一个序列号。这确认收到了所有先前的字节（如果有的话）。两端发送的第一个 ACK 确认另一端的初始序列号本身，但没有数据。

数据偏移（4 位） 指定 TCP 头的大小，以 32 位字为单位。头部的最小尺寸为 5 个字，最大尺寸为 15 个字，因此最小尺寸为 20 字节，最大尺寸为 60 字节，允许在头部有多达 40 字节的选项。这个字段的名称来自于它也是从 TCP 段的开始到实际数据的偏移量。

保留（3 位） 供将来使用，应设置为零。

标志（9 位） 包含 9 个 1 位标志（控制位）：

- NS（1 位）：ECN-nonce——隐蔽性保护。
- CWR（1 位）：拥塞窗口减少（CWR）标志由发送主机设置，以表明它收到了一个设置了 ECE 标志的 TCP 段，并已在拥塞控制机制中作出反应 [b] 。
- ECE（1 位）：ECN-Echo 有双重作用，取决于 SYN 标志的值。它表示：
  - 如果 SYN 标志被设置（1），说明 TCP 对等体有 ECN 能力。
  - 如果 SYN 标志清零（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（1 位）：表明紧急指针字段是重要的
- ACK（1 位）：表示确认字段是重要的。客户端发送的初始 SYN 数据包之后的所有数据包都应该设置这个标志。
- PSH（1 位）：推送功能。要求将缓冲的数据推送给接收的应用程序。
- RST（1 位）：重置连接。
- SYN（1 位）：同步序列号。只有从两端发送的第一个数据包应该设置这个标志。其他一些标志和字段根据这个标志改变含义，有些只有在它被设置时才有效，有些则在它被清除时有效。
- FIN（1 位）：来自发送方的最后一个数据包

窗口大小（16 位） 接收窗口的大小，指定该段的发送方当前愿意接收的窗口大小单位的数量。

校验和（16 位） 16 位校验和字段用于 TCP 头、有效载荷和 IP 伪头的错误检查。伪头包括源 IP 地址、目的 IP 地址、TCP 协议的协议号（6）以及 TCP 头和有效载荷的长度

(字节)。

紧急指针（16 位） 如果 URG 标志被设置，那么这个 16 位字段是序列号的一个偏移，表示最后一个紧急数据字节。

选项（0-320 位，32 的倍数） 这个字段的长度由数据偏移字段决定。选项有最多三个字段。Option-Kind（1 字节），Option-Length（1 字节），Option-Data（可变）。

Option-Kind 字段表示选项的类型，是唯一一个非可选的字段。根据 Option-Kind 的值，接下来的两个字段可能被设置。Option-Length 表示该选项的总长度，Option-Data 包含与该选项相关的数据（如果适用）。例如，一个 Option-Kind 的字节为 1 表示这是一个无操作选项，只用于填充，后面没有 Option-Length 和 Option-Data 字段。一个 Option-Kind 字节为 0，标志着选项的结束，也只有一个字节。Option-Kind 字节为 2 是用来表示最大片段大小的选项，后面有一个 Option-Length 字节，指定 MSS 字段的长度。Option-Length 是给定选项字段的总长度，包括 Option-Kind 和 Option-Length 字段。因此，虽然 MSS 值通常用两个字节表示，但 Option-Length 将是 4。举个例子，一个值为 0x05B4 的 MSS 选项域在 TCP 选项部分被编码为（0x02 0x04 0x05B4）。

4.3 TCP 报文分析

捕获到的 TCP 报文为：

```
f2 61 00 50 7e 41 24 38 88 86 52 4c 50 18 02 04 c8 59 00 00
```

其中，

- **源端口**为 62049，占 16bit；
- **目的端口**为 80，占 16bit；
- **序列号**为 2118198328，占 32bit；
- **确认号**为 2290504268，占 32bit；
- **数据偏移** 20，占 4bit；
- **保留**占 3bit；
- **标志**占 9bit：
  - $NS = 0$，占 1bit，
  - $CWR = 0$，占 1bit，
  - $ECE = 0$，占 1bit，
  - $URG = 0$，占 1bit，

  - $ACK = 1$，占 1bit，
  - $PSH = 1$，占 1bit，
  - $RST = 0$，占 1bit，
  - $SYN = 0$，占 1bit，
  - $FIN = 0$，占 1bit；

- **窗口大小**为 516，占 16bit；
- **校验和**为 0xc859，占 16bit；
- **紧急指针**为 0x0000，占 16bit；
- **选项**占 0bit；
- 此后为 **TCP 有效载荷**，即携带的数据。

且与图 4.2 所示的 TCP 报文结构相同。

5 HTTP 协议

超文本传输协议（英语：HyperText Transfer Protocol，缩写：HTTP）是一种用于分布式、协作式和超媒体信息系统的应用层协议。HTTP 是万维网的数据通信的基础。

HTTP 是一个客户端（用户）和服务端（网站）之间请求和应答的标准，通常使用 TCP 协议。通过使用网页浏览器、网络爬虫或者其它的工具，客户端发起一个 HTTP 请求到服务器上指定端口（默认端口为 80）。我们称这个客户端为用户代理程序（user agent）。应答的服务器上存储着一些资源，比如 HTML 文件和图像。我们称这个应答服务器为源服务器（origin server）。在用户代理和源服务器中间可能存在多个“中间层”，比如代理服务器、网关或者隧道（tunnel）。

尽管 TCP/IP 协议是互联网上最流行的应用，但是在 HTTP 协议中并没有规定它必须使用或它支持的层。事实上 HTTP 可以在任何互联网协议或其他网络上实现。HTTP 假定其下层协议提供可靠的传输。因此，任何能够提供这种保证的协议都可以被其使用，所以其在 TCP/IP 协议族使用 TCP 作为其传输层。

通常，由 HTTP 客户端发起一个请求，创建一个到服务器指定端口（默认是 80 端口）的 TCP 连接。HTTP 服务器则在那个端口监听客户端的请求。一旦收到请求，服务器会向客户端返回一个状态，比如”HTTP/1.1 200 OK”，以及返回的内容，如请求的文件、错误消息、或者其它信息。

5.1 捕获 HTTP 报文实验步骤

1. 打开 Wireshark，进入捕获界面。
2. 将 Wireshark 过滤器设为 `http`。
3. 在浏览器访问 `http://1.15.151.43:3456`（我的服务器地址）。
4. 在 Wireshark 的捕获界面中点击目的地址为 `1.15.151.43` 的帧，可以看到捕获到的客户端请求 HTTP 报文，如图 5.1 所示。
5. 捕获到的服务端应答 HTTP 报文如图 5.2 所示。

5.2 HTTP 报文简述

HTTP/1.1 协议中共定义了八种方法（也叫“动作”）来以不同方式操作指定的资源。方法名称是区分大小写的。当某个请求所针对的资源不支持对应的请求方法的时候，服务器应当返回状态码 405（Method Not Allowed），当服务器不认识或者不支持对应的请求方法的时候，应当返回状态码 501（Not Implemented）。

发出的请求信息（message request）包括以下几个：

- 请求行（例如 `GET /images/logo.gif HTTP/1.1`，表示从/images 目录下请求 logo.gif 这个文件）

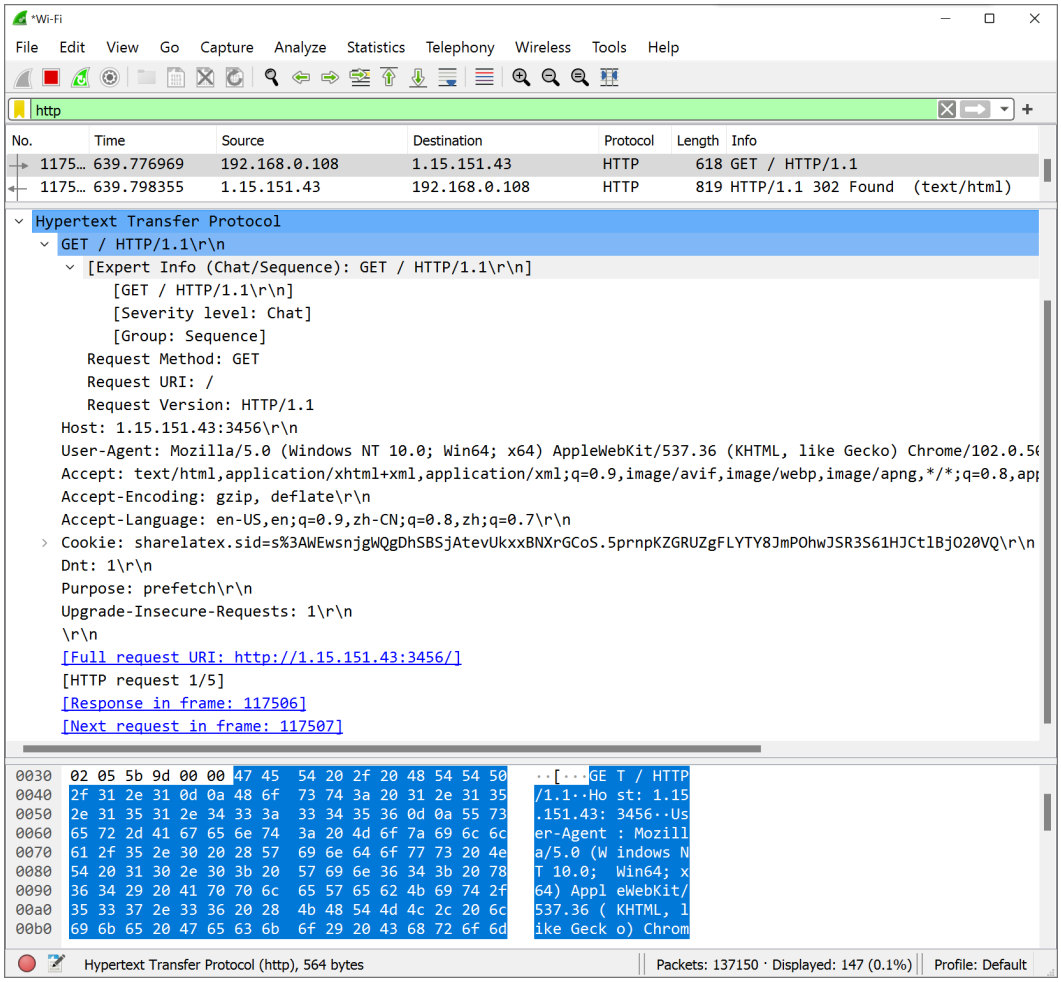

图 5.1: 捕获到的客户端请求 HTTP 报文

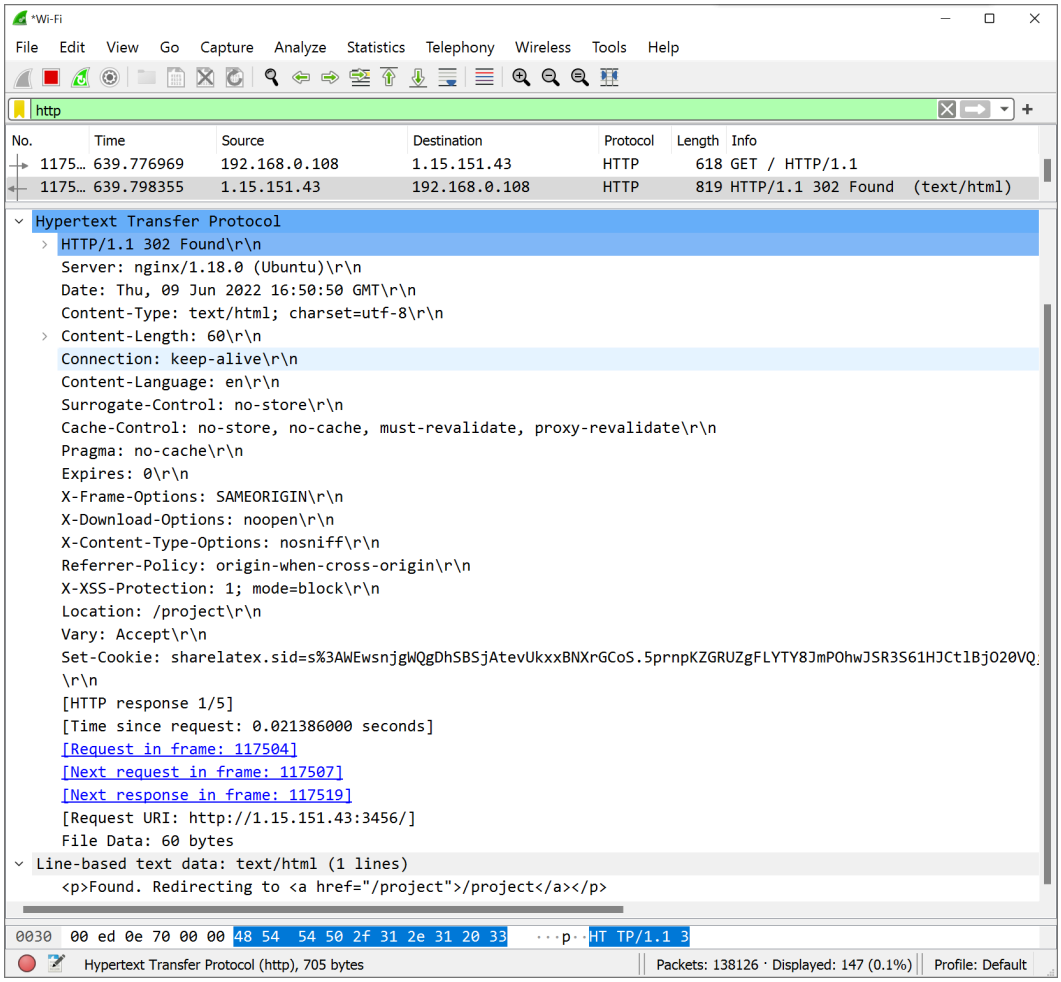

图 5.2: 捕获到的服务端应答 HTTP 报文

- 请求头（例如 `Accept-Language: en`）
- 空行
- 其他消息体

请求行和标题必须以 `<CR><LF>` 作为结尾。空行内必须只有 `<CR><LF>` 而无其他空格。在 HTTP/1.1 协议中，所有的请求头，除 Host 外，都是可选的。

HTTP 服务器至少应该实现 GET 和 HEAD 方法，其他方法都是可选的。

5.2.1 GET

向指定的资源发出“显示”请求。使用 GET 方法应该只用在读取资料，而不应当被用于产生“副作用”的操作中，例如在网络应用程序中。其中一个原因是 GET 可能会被网络爬虫等随意访问。参见安全方法。浏览器直接发出的 GET 只能由一个 url 触发。GET 上要在 url 之外带一些参数就只能依靠 url 上附带 querystring。

5.2.2 HEAD

与 GET 方法一样，都是向服务器发出指定资源的请求。只不过服务器将不传回资源的本文部分。它的好处在于，使用这个方法可以在不必传输全部内容的情况下，就可以获取其中“关于该资源的信息”（元信息或称元数据）。

5.2.3 POST

向指定资源提交数据，请求服务器进行处理（例如提交表单或者上传文件）。数据被包含在请求本文中。这个请求可能会创建新的资源或修改现有资源，或二者皆有。每次提交，表单的数据被浏览器用编码到 HTTP 请求的 body 里。浏览器发出的 POST 请求的 body 主要有两种格式，一种是 `application/x-www-form-urlencoded` 用来传输简单的数据，大概就是 `key1=value1&key2=value2` 这样的格式。另外一种是传文件，会采用 `multipart/form-data` 格式。采用后者是因为 `application/x-www-form-urlencoded` 的编码方式对于文件这种二进制的数据非常低效。

5.2.4 PUT

向指定资源位置上传其最新内容。

5.2.5 DELETE

请求服务器删除 Request-URI 所标识的资源。

5.2.6 TRACE

回显服务器收到的请求，主要用于测试或诊断。

5.2.7 OPTIONS

这个方法可使服务器传回该资源所支持的所有 HTTP 请求方法。用 “*” 来代替资源名称，向 Web 服务器发送 OPTIONS 请求，可以测试服务器功能是否正常运作。

5.2.8 CONNECT

HTTP/1.1 协议中预留给能够将连接改为隧道方式的代理服务器。通常用于 SSL 加密服务器的链接（经由非加密的 HTTP 代理服务器）。

5.3 HTTP 报文分析

捕获到的 HTTP 报文如图 5.1 和 5.2 所示。

5.3.1 客户端请求

客户端请求的请求行、请求头分别在下方代码的第 1 行和第 2 至 10 行。其中，请求行由请求方法（GET）、请求路径（/）和 HTTP 版本（HTTP/1.1）组成；请求头由请求头字段名称和请求头值组成，请求头字段名称和请求头值之间用 : 分隔。

```
1  GET / HTTP/1.1
2  Host: 1.15.151.43:3456
3  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   ↪ (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
4  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=0.9
5  Accept-Encoding: gzip, deflate
6  Accept-Language: en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7
7  Cookie: sharelatex.sid=s%3AWEwsnjgWQgDhSBSjAtevUkxxBNXrGCoS.5prnpKZGRUZgF」
   ↪ LYTY8JmPOhwJSR3S61HJCtlBjO20VQ
8  Dnt: 1
9  Purpose: prefetch
10 Upgrade-Insecure-Requests: 1
```

请求头后跟一个空行，表示请求头结束。后面的是请求体（可选）。

5.3.2 服务器响应

服务器响应的响应行、响应行分别在下方代码的第 1 行、第 2 至 19 行。其中，响应行由协议版本（HTTP/1.1）、状态码（302）和状态码描述（Found）组成；响应头由响应头字段名称和响应头值组成，响应头字段名称和响应头值之间用 : 分隔。

```
HTTP/1.1 302 Found
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 09 Jun 2022 16:50:50 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 60
Connection: keep-alive
Content-Language: en
Surrogate-Control: no-store
Cache-Control: no-store, no-cache, must-revalidate, proxy-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: SAMEORIGIN
X-Download-Options: noopen
X-Content-Type-Options: nosniff
Referrer-Policy: origin-when-cross-origin
X-XSS-Protection: 1; mode=block
Location: /project
Vary: Accept
Set-Cookie: sharelatex.sid=s%3AWEwsnjgWQgDhSBSjAtevUkxxBNXrGCoS.5prnpKZGR」
↪ UZgFLYTY8JmPOhwJSR3S61HJCtlBjO20VQ; Path=/; Expires=Tue, 14 Jun 2022
↪ 16:50:50 GMT; HttpOnly; SameSite=Lax
```

响应头后跟一个空行，表示响应头结束。后面的是响应体正文，此处为 1 行以行为单位的文本数据，为

```
<p>Found. Redirecting to <a href="/project">/project</a></p>
```