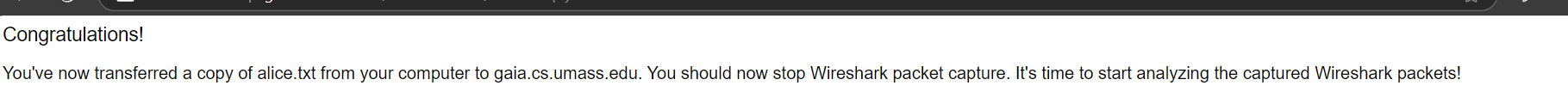
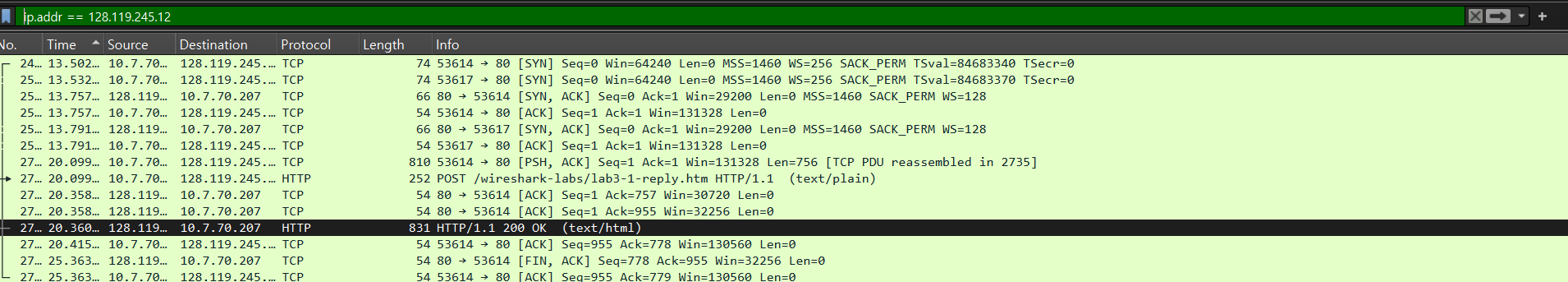
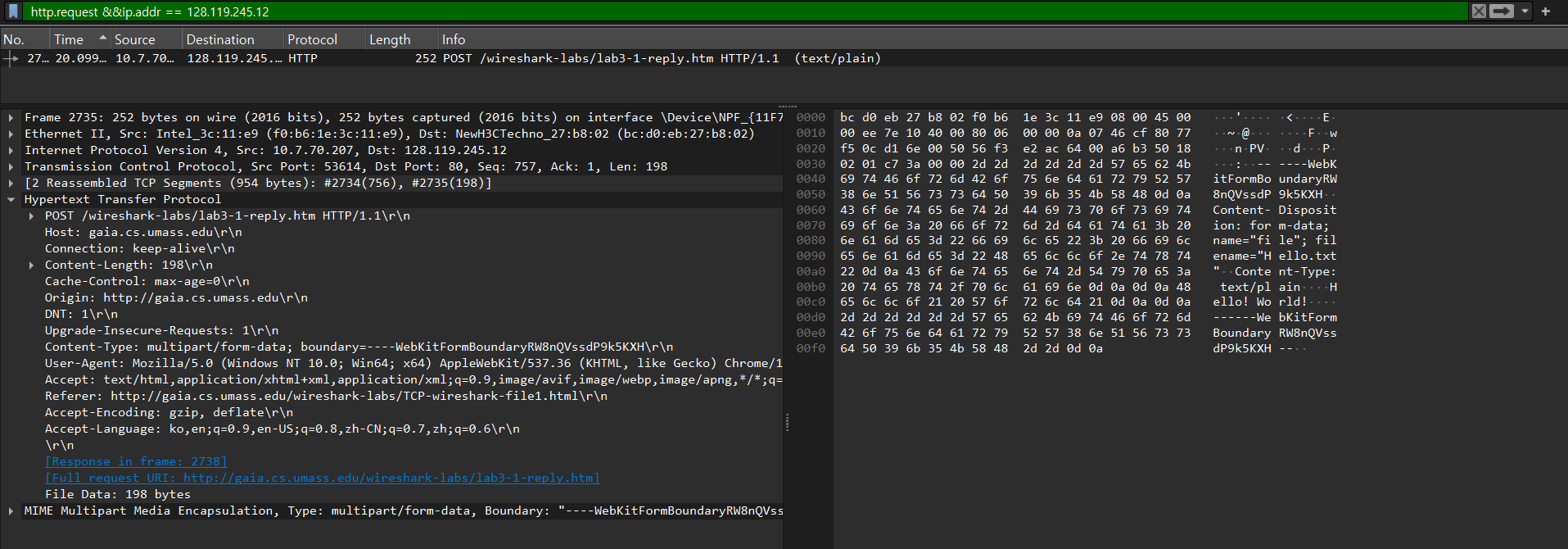
## 1）上传文件



## 2) 过滤出目的地址的协议



## 3) HTTP 请求报文 (HTTP Request)



### 请求行：

请求类型 (Request Method)：POST - 表示客户端向服务器发送数据。

请求路径 (Request Path)：/wireshark-labs/lab3-1-reply.htm - 客户端请求的目标资源。

协议版本 (Protocol Version)：HTTP/1.1 - HTTP协议的版本。

### 请求头：

Host：指定目标服务器的主机名（gaia.cs.umass.edu）。

User-Agent：发送请求的客户端信息（Mozilla/5.0）。

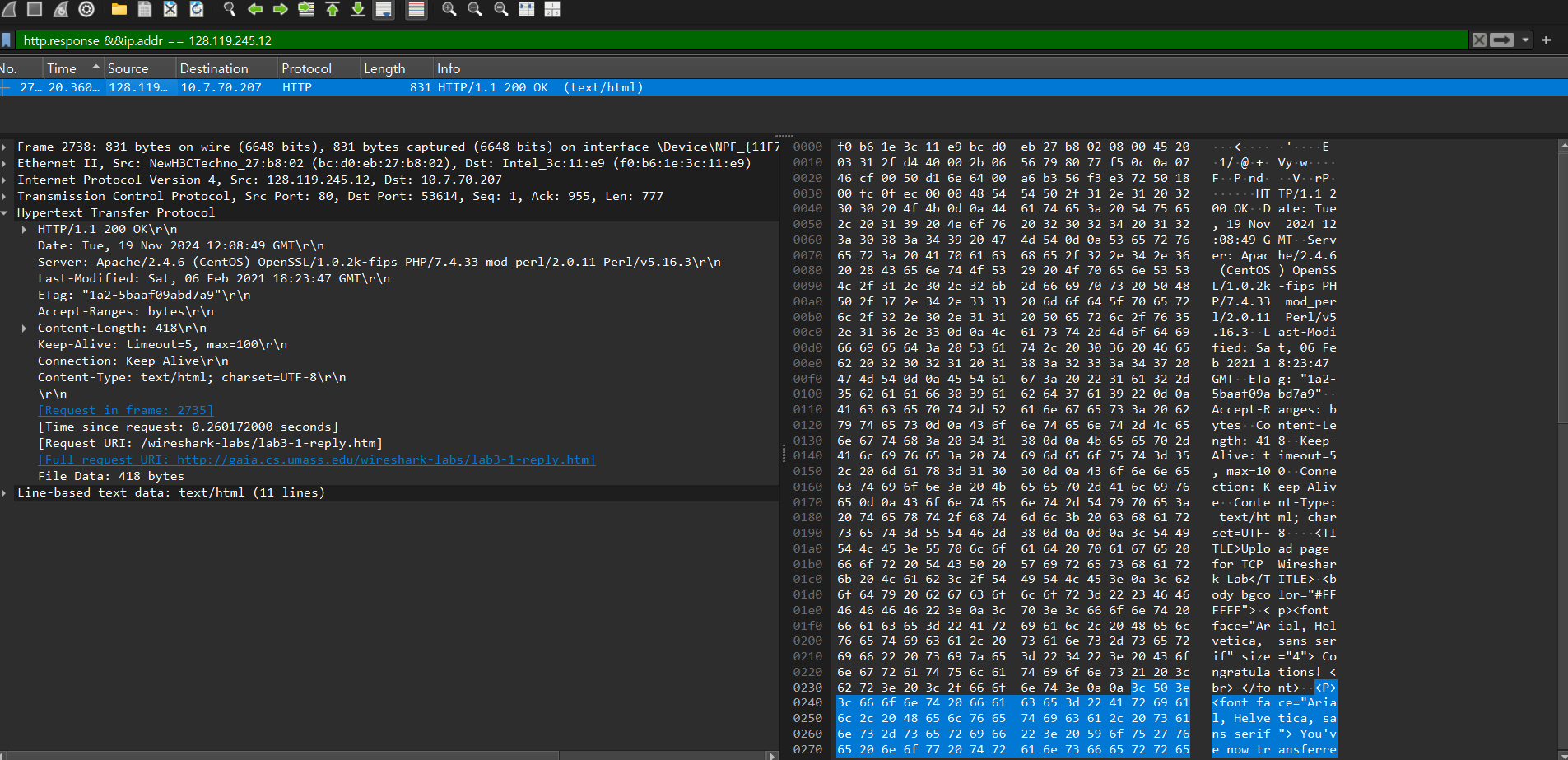
Content-Type：请求数据的媒体类型（multipart/form-data）。

Referer：告诉服务器请求来源的页面地址。

请求体 (Request Body)：

在截图中可以看到实际数据部分，包含传递的数据内容，比如表单信息。

## HTTP 响应报文 (HTTP Response)



### 状态行 (Status Line)：

HTTP/1.1 200 OK

协议版本 (Protocol Version)：HTTP/1.1。

状态码 (Status Code)：200 - 表示请求成功。

状态描述 (Status Text)：OK。

### 响应头 (Response Headers)：

arduino

Server: Apache/2.4.6

Content-Type: text/html ...

**Server**：标明服务器类型和版本（Apache/2.4.6）。

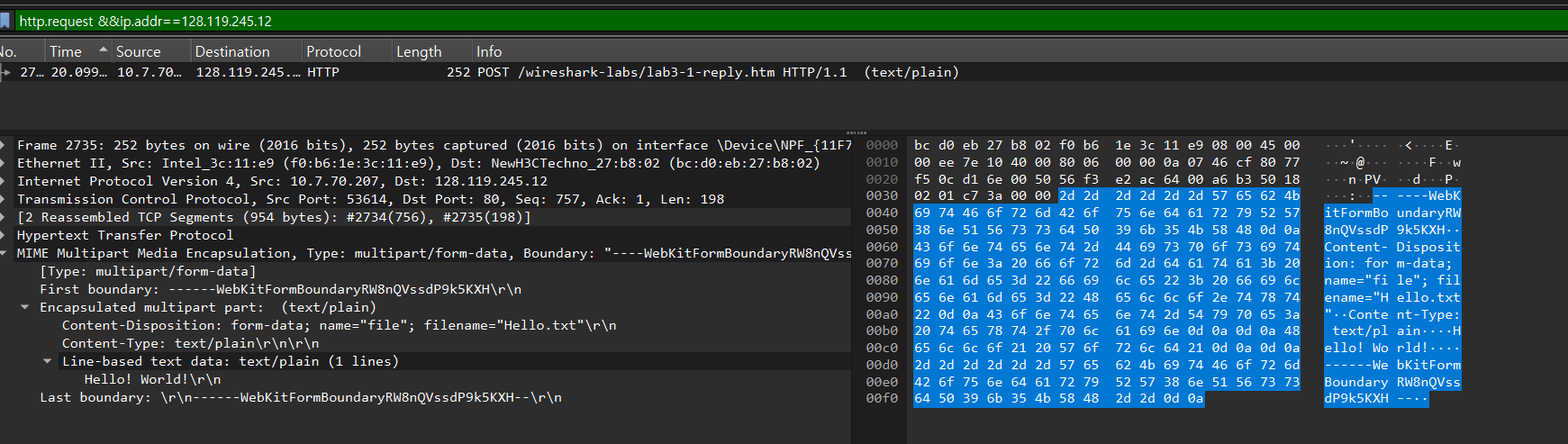
**Content-Type**：响应内容的媒体类型（text/html）。

**Content-Length**：响应内容的字节长度。

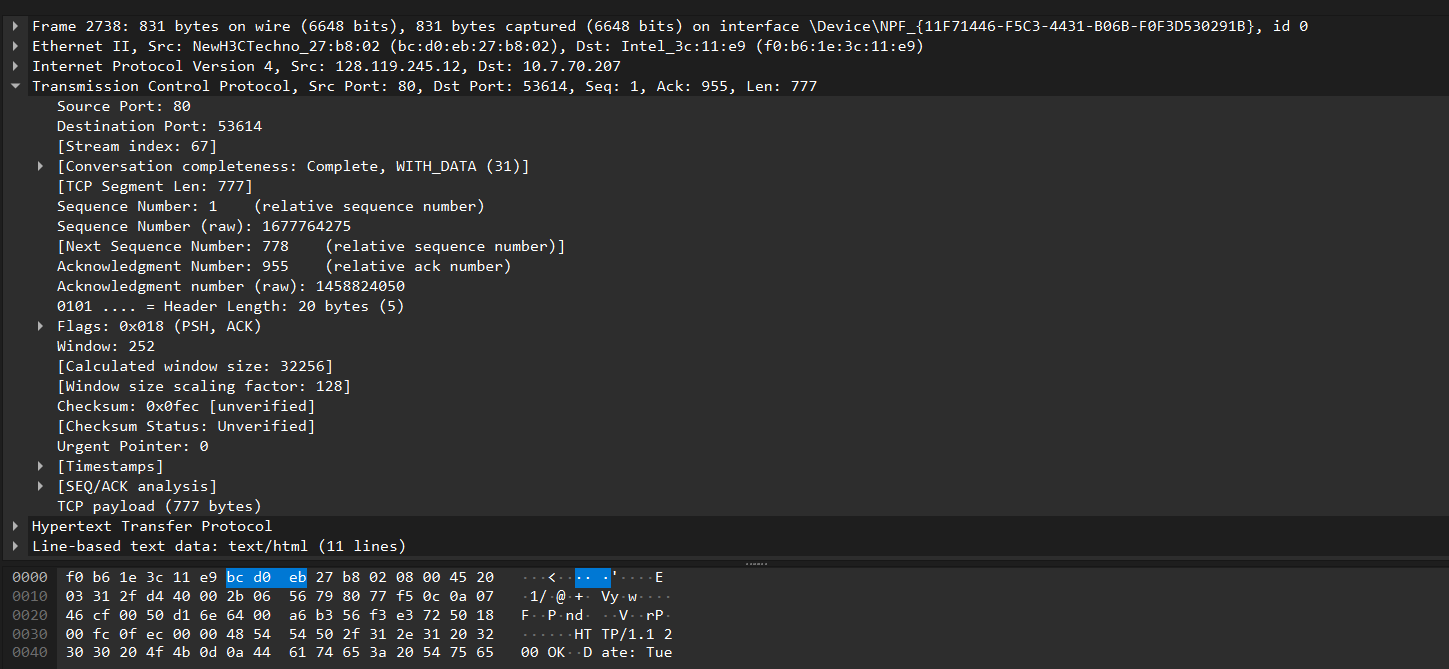
### 响应体 (Response Body)：

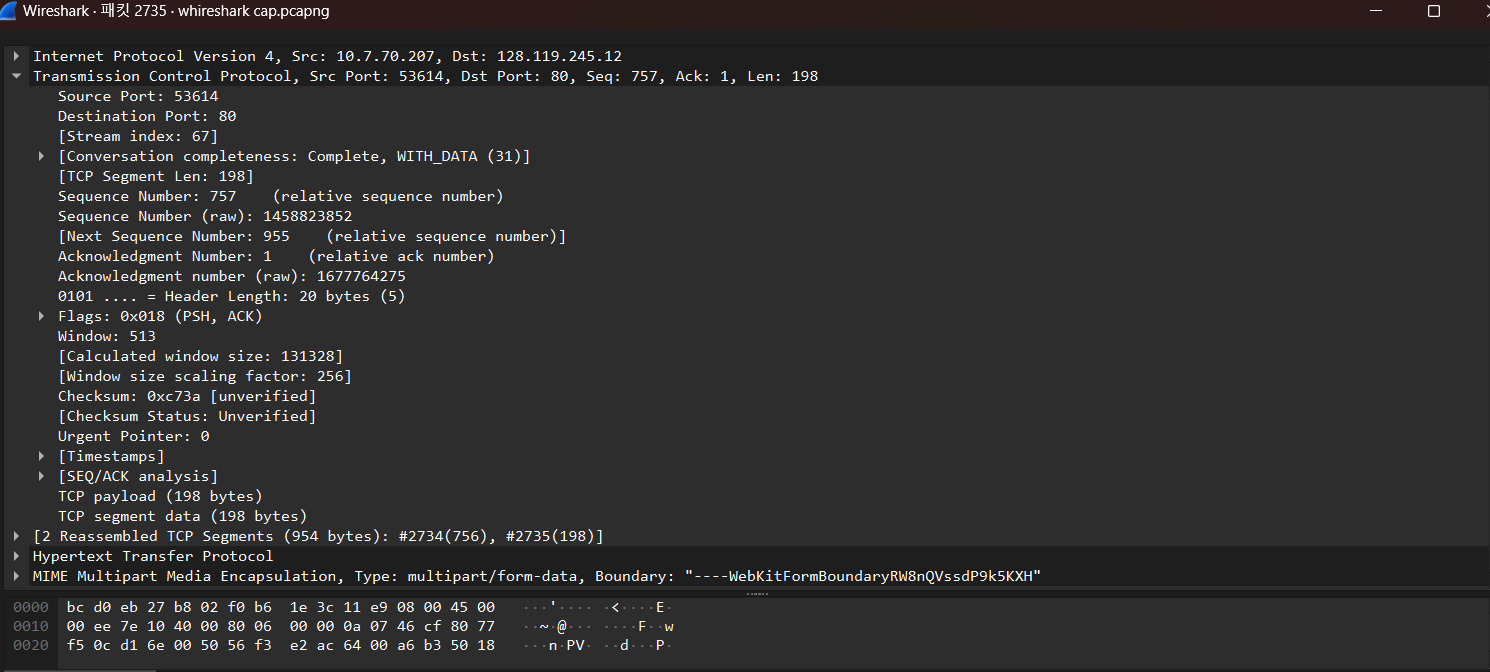
从截图中蓝色高亮部分可以看到返回的HTML内容，包括网页的主要内容。

### 上传内容：



## 4） TCP 协议报文





### TCP 报文字段解释

**Source Port**: 53614  
表示发送端的源端口，用于标识通信的应用程序。

**Destination Port**: 80  
表示接收端的目标端口，通常为服务器的 HTTP 服务端口。

**Sequence Number**:

**Raw**: 表示当前 TCP 报文的序列号，数据从该序号开始。

**Relative Sequence Number**: Wireshark 将初始序列号归零后的值，便于分析。

**Acknowledgment Number**:

**Raw**: 表示接收到的下一个数据段的序列号。

**Relative Acknowledgment Number**: Wireshark 标准化值，用于确认接收。

**Header Length**: 20 bytes，TCP 报头的长度。

**Flags**: 0x018 (PSH, ACK)

**PSH (Push)**: 数据需要立即发送到接收应用。

**ACK (Acknowledgment)**: 确认报文，表示接收方已经收到数据。

**Window Size**: 252，用于流量控制，表示接收方可以接受的字节数。

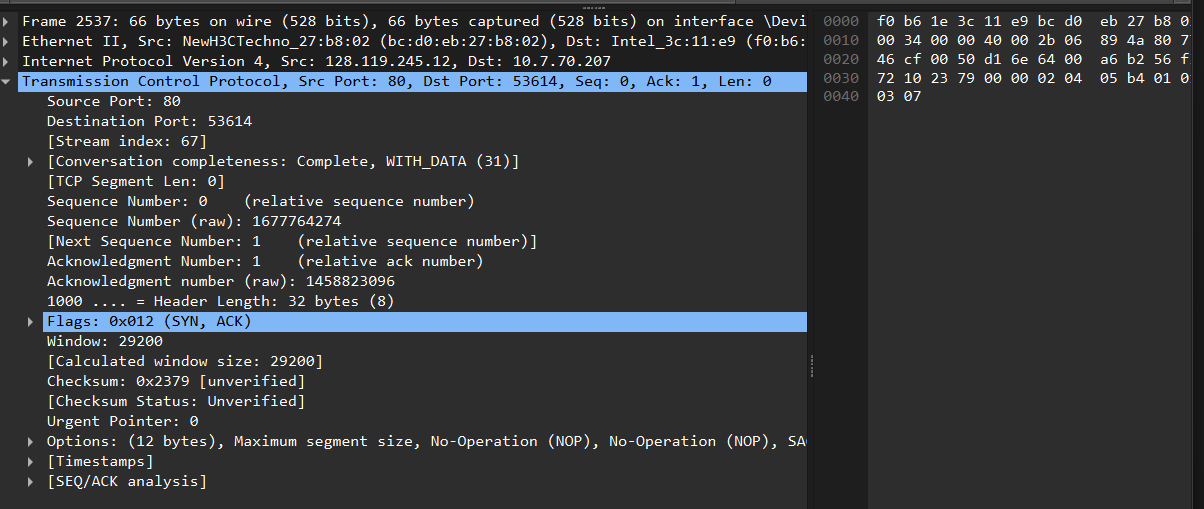
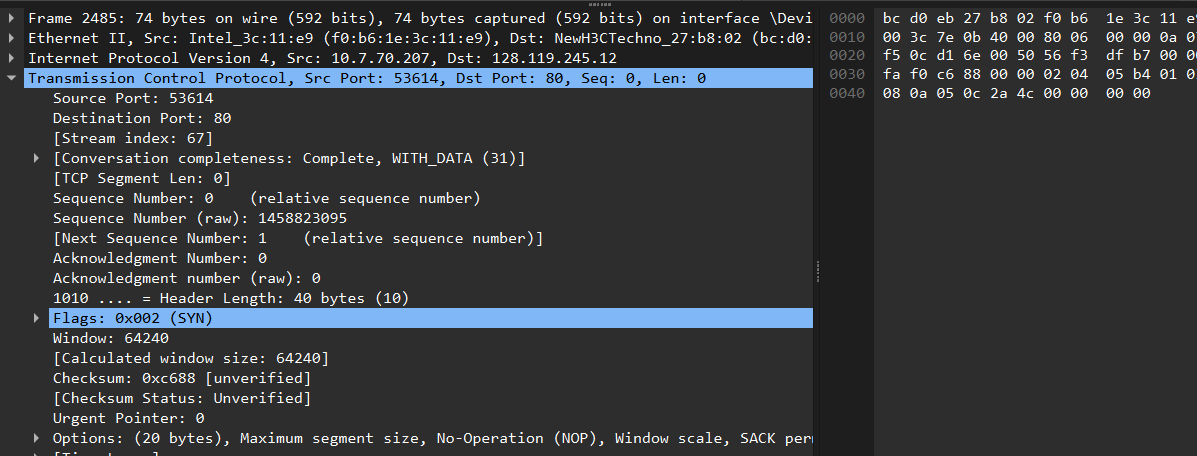
**Checksum**: 0x0fec，报文的校验值，用于确保数据传输的完整性。

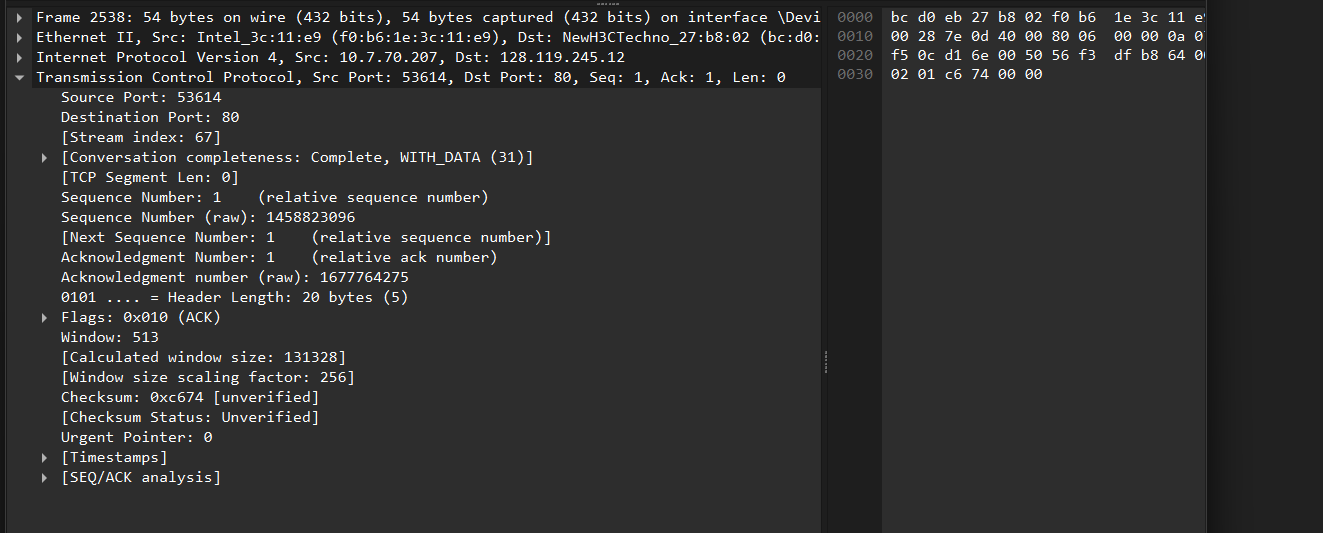
**Urgent Pointer**: 0，表示是否有紧急数据，这里未使用。

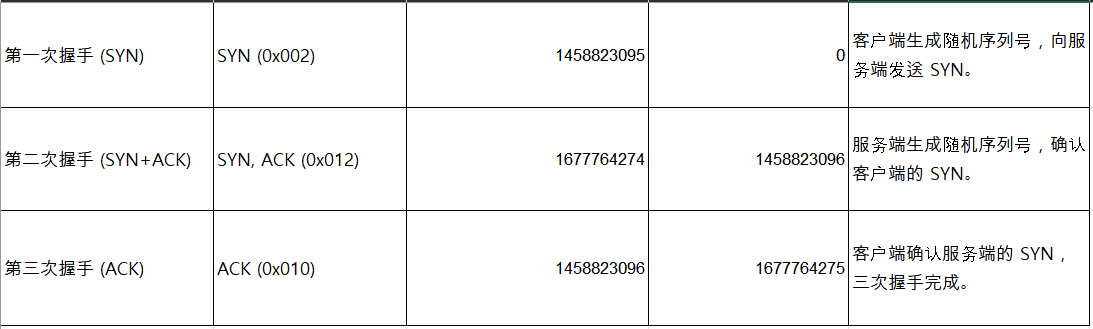
### 分析数据部分：

**TCP Payload**: 包含报文中的实际传输数据，比如 HTTP 请求内容或表单数据。

## **5) TCP 三次握手分析，**







### 第一次握手（SYN）第一个截图:

客户端生成随机的 **Raw Sequence Number** 1458823095，没有 Acknowledgment Number。

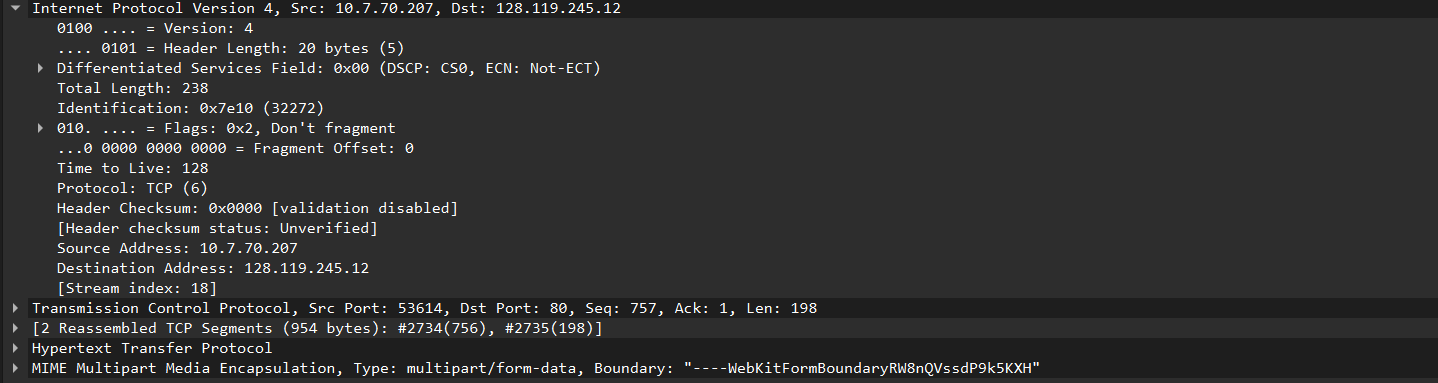
### 第二次握手（SYN+ACK）第二个截图:

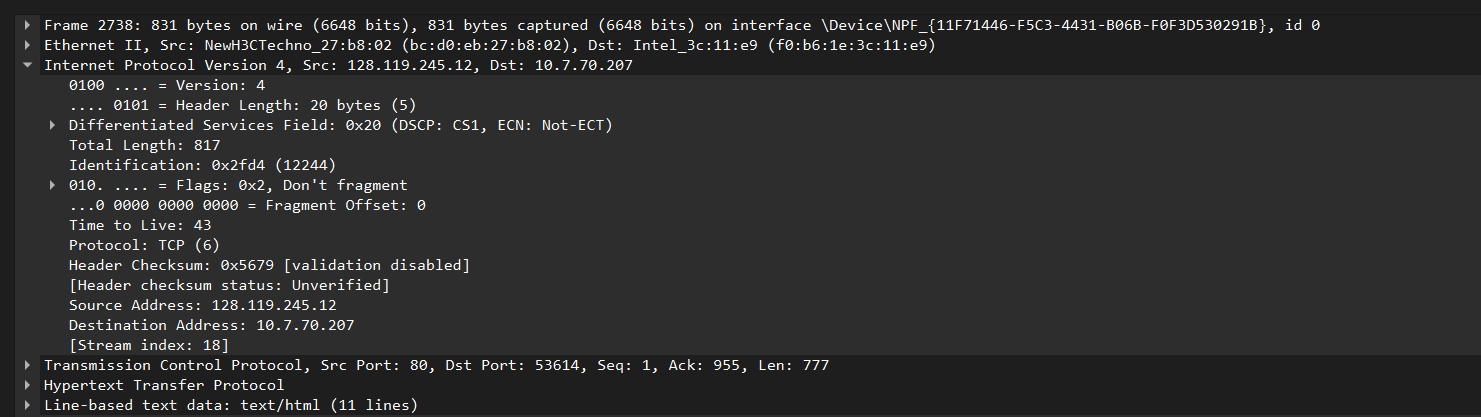
服务端生成随机的 **Raw Sequence Number** 1677764274，并将 Acknowledgment Number 设置为 1458823096（确认客户端的序列号加 1）。

### 第三次握手（ACK）第三个截图:

客户端将 Raw Sequence Number 设置为 1458823096，并将 Acknowledgment Number 设置为 1677764275（确认服务端的序列号加 1）。

## 6)IP 数据包字段解析





**Version（版本）**:

**值:** 4

含义: 表示 IP 协议版本为 IPv4（Internet Protocol version 4）。

**Header Length（头长）**:

**值:** 20 bytes（5 个 32-bit 字）

**含义**: 表示 IP 数据包的头部长度，单位为 4 字节。

**Differentiated Services Field（服务类型 / DS 字段）**:

**值**: 0x00 或 0x20（不同截图中值可能不同）

**含义**: 用于定义服务质量（QoS），例如优先级或延迟。DSCP 表示差分服务代码点，ECN 表示显式拥塞通知。

**Total Length（总长）:**

**值:** 238 或 817 字节

**含义**: 表示 IP 数据包的总长度，包括头部和数据部分。

**Identification（标识）:**

**值**: 0x7e10 或 0x2fd4

**含义:** 数据包的唯一标识符，用于区分哪些分片属于同一个数据包。

**Flags（标志）:**

**值:** 0x2（Don't fragment）

**含义:**

位 0: 保留，始终为 0。

位 1: DF（Don't Fragment，不分片）。

位 2: MF（More Fragments，还有分片）。

此处 0x2 表示不允许分片。

**Fragment Offset（片偏移）:**

**值:** 0

**含义:** 表示分片数据相对于原始数据包的偏移量。此处为 0 表示该数据包没有被分片。

**Time to Live（生存时间 / TTL）**:

**值**: 128 或 43

**含义**: 表示数据包可以经过的最大路由跳数，避免数据包在网络中无限循环。每经过一个路由器，TTL 减 1。

**Protocol（协议）**:

**值**: 6（TCP）

**含义**: 表示 IP 数据包的上层协议类型。6 表示传输控制协议（TCP）。

**Header Checksum（校验和）**:

**值:** 0x0000 或 0x5679（不同截图中值不同）

**含义:** 用于验证 IP 头部的完整性。

**Source Address（源地址）**:

**值**: 10.7.70.207 或 128.119.245.12

**含义:** 表示发送数据包的源 IP 地址。

**Destination Address（目的地址）**:

**值:** 128.119.245.12 或 10.7.70.207

**含义:** 表示接收数据包的目标 IP 地址。

## 实验心得：

通过对 HTTP 请求和响应报文的捕获与解析，我更直观地理解了 HTTP 协议的工作原理，特别是请求类型（如 GET 和 POST）的具体使用场景。TCP 报文的解析则帮助我理解了三次握手的全过程，包括 SYN、ACK 和序列号的变化，这加深了我对可靠传输机制的认知。在实验过程中，我曾遇到过滤条件不正确导致无结果的问题。通过查阅文档并反复尝试，我明确了过滤语法的正确用法，最终成功捕获并分析了目标数据包。