

实验报告

计算机网络 Computer Networking

报告标题: 协议分析——IP 协议

学号: 19240212

姓名: 华博文

日期: 2025 年 10 月 5 日

一 实验目的

本实验旨在掌握 Wireshark 的启动、网卡选择、IP 相关数据包捕获与筛选操作, 熟悉其主窗口三部分结构及功能; 理解 IPv4 协议头部的完整结构(字段组成、含义及格式), 能准确提取 Version、Header Length、TTL、Protocol、源 / 目的 IP 等关键字段信息; 掌握 IP 分片的触发条件、分片原理(标志位、分片偏移)及接收端重组过程, 理解 ICMP 协议与 IP 协议的关联; 学会按 IP 协议格式解析指定十六进制 IP 数据报, 提取 IPv4 头部、ICMP 头部及载荷内容并规范输出; 最终加深对计算机网络分层模型(数据链路层、网络层)的理解, 验证 IP 协议在数据传输中的封装与分片机制.

二 实验内容简要描述

1. Wireshark 与 IP 头部及分片分析: 通过终端启动 Wireshark, 执行 `ping www.nnu.edu.cn -s 4500 -c 1` 命令, 启动抓包后等待命令执行完成, 停止抓包并筛选 IP 报文, 分析 IP 头部关键字段及 IP 分片的数量、标志位、重组逻辑, 同时提取 ICMP 头部字段;
2. IP 数据报解析: 基于给定 IP 十六进制数据, 用 C++ 编写解析逻辑, 按指定格式输出 IPv4 头部、ICMP 头部及载荷内容, 并将结果保存至指定文件.

三 实验步骤与结果分析

1 Wireshark 与 IP 头部及分片分析

1.1 启动 Wireshark

打开 Linux 终端, 输入命令 `wireshark` 并回车, 启动 Wireshark 网络分析工具.

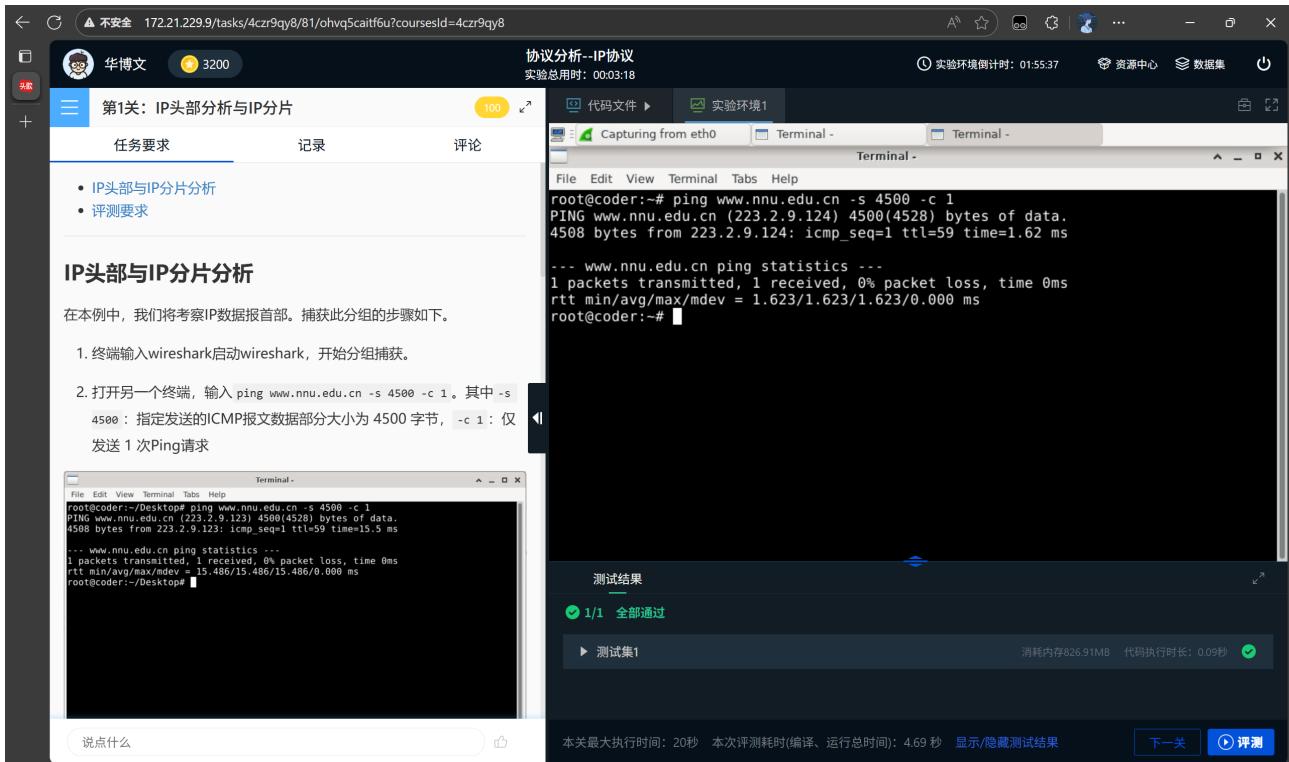
1.2 选择捕获网卡

在 Wireshark 顶部的网卡列表中, 选择当前设备使用的网卡 `eth0` (而非回环网卡 `lo`), 确保能捕获外网通信数据.

1.3 执行 `ping` 命令

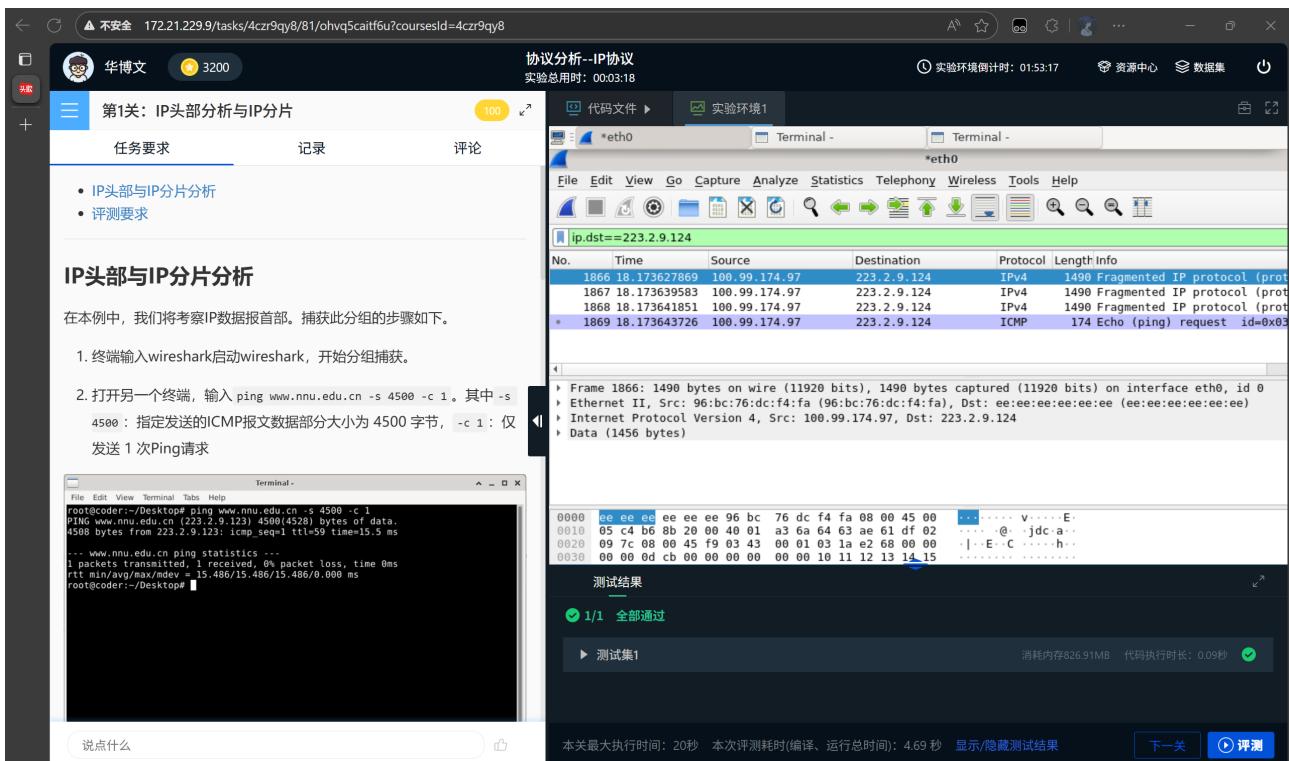
打开另一个 Linux 终端, 输入命令 `ping www.nnu.edu.cn -s 4500 -c 1` 并回车: 其中 `-s 4500` 指定 ICMP 报文数据部分大小为 4500 字节, `-c 1` 表示仅发送 1 次请求; 终端反

反馈显示 4508 bytes from 223.2.9.124: icmp_seq=1 ttl=59 time=1.62 ms, 表明 ping 请求已发送并收到响应。



1.4 停止捕获与筛选 IP 报文

待 ping 命令执行完成, 点击 Wireshark 的“停止捕获”按钮 (灰色方形图标), 结束抓包。在 Wireshark 顶部的 Apply a display filter 输入框中输入 ip.dst==223.2.9.124 并回车, 报文列表仅显示目的 IP 为该地址的 IP 相关报文。



1.5 IP 头部字段分析

以 No.1866 报文 (IPv4 分片) 为例, 展开 “Internet Protocol Version 4” 字段, 各关键字段如下:

The screenshot shows the NetworkMiner interface with the following details:

- Top Bar:** 不安全 172.21.229.9/tasks/4czr9qy8/81/ohvq5cait6u?coursesId=4czr9qy8
- Left Panel:** 华博文 3200, 第1关: IP头部分析与IP分片, 任务要求, 记录, 评论, IP头部与IP分片分析 (selected), 评估要求.
- Middle Panel:** 协议分析--IP协议, 实验总用时: 00:03:18, 实验环境倒计时: 01:51:15, 资源中心, 数据集.
- Right Panels:**
 - Code Editor:** *eth0, Terminal - *eth0, File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help.
 - Terminal:** ip.dst==223.2.9.124, showing a list of captured frames (No. 1866, 1867, 1868, 1869).

No.	Time	Source	Destination	Protocol	Length	Info
1866	18.173627869	100.99.174.97	223.2.9.124	IPv4	1490	Fragmented IP protocol (proto)
1867	18.173639583	100.99.174.97	223.2.9.124	IPv4	1490	Fragmented IP protocol (proto)
1868	18.173641851	100.99.174.97	223.2.9.124	IPv4	1490	Fragmented IP protocol (proto)
1869	18.173643726	100.99.174.97	223.2.9.124	ICMP	174	Echo (ping) request id=0x02
 - Details View:** Frame 1866: 1490 bytes on wire (11920 bits), 1490 bytes captured (11920 bits) on interface eth0, id 0. Ethernet II, Src: 96:bc:76:dc:f4:fa (96:bc:76:dc:f4:fa), Dst: ee:ee:ee:ee:ee:ee (ee:ee:ee:ee:ee:ee). Internet Protocol Version 4, Src: 100.99.174.97, Dst: 223.2.9.124. Version: 4, Header Length: 20 bytes (5). Differentiated Services Field: 0x00 (DS0, ECN: Not-ECT). Total Length: 1476. Identification: 0xb68b (46731). Flags: 0x2000, More fragments. Fragment offset: 0. Time to live: 64. Protocol: ICMP (1). Header checksum: 0xa36a [validation disabled]. [Header checksum status: Unverified]. Source: 100.99.174.97. Destination: 223.2.9.124. Reassembled IPv4 in frame: 1869. Data (1456 bytes): 0000: d8 ee ee ee ee ee 96 bc 76 dc f4 fa 08 00 45 00 |.....v.....E| 0010: 05 c4 b6 8b 20 00 40 01 a3 6a 64 63 ae 61 df 02 |@ jdc a..| 0020: 09 7c 08 00 45 f9 03 43 00 01 03 1a e2 68 00 00 |- E C - h| 0030: 00 00 0d cb 00 00 00 00 00 10 11 12 13 14 15

- **Version (版本):** 4, 标识为 IPv4 协议;
- **Header Length (头部长度):** 20 bytes (IPv4 头部长度以 4 字节为单位);
- **Total Length (总长度):** 1476 bytes (标识整个 IP 数据报的长度, 含头部和数据);
- **Identification (标识):** 0xb68b, 十进制为 46731, 用于标识同一原始 IP 数据报的所有分片;
- **Flags (标志):** 0x2000, 其中 “More fragments” 位设为 1, 标识当前为分片且后续仍有分片, “Don't fragment” 位设为 0, 允许分片;
- **Fragment offset (分片偏移):** 0, 标识当前分片在原始数据报数据部分的起始位置 (以 8 字节为单位);
- **TTL (生存时间):** 64, 表示数据报最多可经过 64 个路由器, 每经过一个路由器值减 1;
- **Protocol (上层协议):** 1, 标识上层为 ICMP 协议 (标准协议号中 1 对应 ICMP);
- **Source IP (源 IP 地址):** 100.99.174.97, 为发送 ping 请求的本地 IP;
- **Destination IP (目的 IP 地址):** 223.2.9.124, 为南京师范大学官网相关 IP.

1.6 IP 分片与重组分析

1. **分片数量与特征:** 筛选后共显示 4 个报文, 其中 3 个为 IPv4 分片 (No.1866、1867、1868), 1 个为完整 ICMP 报文 (No.1869). 3 个分片的 Protocol 均为 IPv4, Info 均包含 “Fragmented IP protocol”, 且 Flags 字段的 “More fragments” 均设为 1; 分片偏移依次为 0、1456、2912 (十进制), 对应数据部分起始位置, 每个分片的 Length 均为 1490 bytes (含数据链路层头部).

协议分析--IP协议

实验总用时: 00:03:18

① 实验环境倒计时: 01:48:42

资源中心 数据集

华博文 3200

第1关: IP头部分析与IP分片

任务要求 记录 评论

- IP头部与IP分片分析
- 评测要求

IP头部与IP分片分析

在本例中, 我们将考察IP数据报首部。捕获此组分的步骤如下。

- 终端输入wireshark启动wireshark, 开始分组捕获。
- 打开另一个终端, 输入 ping www.nnu.edu.cn -s 4500 -c 1。其中 -s 4500 : 指定发送的ICMP报文数据部分大小为 4500 字节, -c 1 : 仅发送 1 次Ping请求

说点什么

Terminal -

```
root@coder:~/Desktop# ping www.nnu.edu.cn -s 4500 -c 1
PING www.nnu.edu.cn (223.2.9.123) 4500(4528) bytes of data.
4508 bytes from 223.2.9.123: icmp_seq=1 ttl=59 time=15.5 ms
-- www.nnu.edu.cn ping statistics --
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 15.486/15.486/15.486/0.000 ms
root@coder:~/Desktop#
```

说点什么

协议分析--IP协议

实验总用时: 00:03:18

① 实验环境倒计时: 01:48:42

资源中心 数据集

华博文 3200

*eth0 Terminal - Terminal -

ip.dst==223.2.9.124

No.	Time	Source	Destination	Protocol	Length Info
1866	18.173627869	100.99.174.97	223.2.9.124	IPv4	1490 Fragmented IP protocol (proto)
1867	18.173639583	100.99.174.97	223.2.9.124	IPv4	1490 Fragmented IP protocol (proto)
1868	18.173641851	100.99.174.97	223.2.9.124	IPv4	1490 Fragmented IP protocol (proto)
*	18.173643726	100.99.174.97	223.2.9.124	ICMP	174 Echo (ping) request id=0x03

Frame 1867: 1490 bytes on wire (11920 bits), 1490 bytes captured (11920 bits) on interface eth0, id 0
Ethernet II, Src: (96:bc:76:dc:f4:fa) (96:bc:76:dc:f4:fa), Dst: (ee:ee:ee:ee:ee:ee) (ee:ee:ee:ee:ee:ee)
Internet Protocol Version 4, Src: 100.99.174.97, Dst: 223.2.9.124
0100 ... Version: 4
.... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1476
Identification: 0xb6b8 (46731)
Flags: 0x20b6, More fragments
Fragment offset: 1456
Time to live: 64
Protocol: ICMP (1)
Header checksum: 0xa2b4 [validation disabled]
[Header checksum status: Unverified]
Source: 100.99.174.97
Destination: 223.2.9.124
Reassembled IPv4 in frame: 1869

Data (1456 bytes)

0010 05 c4 b6 8b 20 b6 40 01 a2 b4 64 63 ae 61 df 02 ... @.dc.a...
0020 09 7c a8 a9 aa ab ac ad ae af b6 b1 b2 b3 b4 b5 ...
0030 b6 b7 b8 b9 ba bb bc bd be c8 c1 c2 c3 c4 c5 ...
0040 c6 c7 c8 c9 ca cb cc cd ce cf d0 d1 d2 d3 d4 d5 ...

本关最大执行时间: 20秒 本次评测耗时(编译、运行总时间): 4.69 秒 显示/隐藏测试结果

下一关 评估

协议分析--IP协议

实验总用时: 00:03:18

① 实验环境倒计时: 01:48:42

资源中心 数据集

华博文 3200

第1关: IP头部分析与IP分片

任务要求 记录 评论

- IP头部与IP分片分析
- 评测要求

IP头部与IP分片分析

在本例中, 我们将考察IP数据报首部。捕获此组分的步骤如下。

- 终端输入wireshark启动wireshark, 开始分组捕获。
- 打开另一个终端, 输入 ping www.nnu.edu.cn -s 4500 -c 1。其中 -s 4500 : 指定发送的ICMP报文数据部分大小为 4500 字节, -c 1 : 仅发送 1 次Ping请求

说点什么

Terminal -

```
root@coder:~/Desktop# ping www.nnu.edu.cn -s 4500 -c 1
PING www.nnu.edu.cn (223.2.9.123) 4500(4528) bytes of data.
4508 bytes from 223.2.9.123: icmp_seq=1 ttl=59 time=15.5 ms
-- www.nnu.edu.cn ping statistics --
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 15.486/15.486/15.486/0.000 ms
root@coder:~/Desktop#
```

说点什么

协议分析--IP协议

实验总用时: 00:03:18

① 实验环境倒计时: 01:48:42

资源中心 数据集

华博文 3200

*eth0 Terminal - Terminal -

ip.dst==223.2.9.124

No.	Time	Source	Destination	Protocol	Length Info
1866	18.173627869	100.99.174.97	223.2.9.124	IPv4	1490 Fragmented IP protocol (proto)
1867	18.173639583	100.99.174.97	223.2.9.124	IPv4	1490 Fragmented IP protocol (proto)
1868	18.173641851	100.99.174.97	223.2.9.124	IPv4	1490 Fragmented IP protocol (proto)
*	18.173643726	100.99.174.97	223.2.9.124	ICMP	174 Echo (ping) request id=0x03

Frame 1868: 1490 bytes on wire (11920 bits), 1490 bytes captured (11920 bits) on interface eth0, id 0
Ethernet II, Src: (96:bc:76:dc:f4:fa) (96:bc:76:dc:f4:fa), Dst: (ee:ee:ee:ee:ee:ee) (ee:ee:ee:ee:ee:ee)
Internet Protocol Version 4, Src: 100.99.174.97, Dst: 223.2.9.124
0100 ... Version: 4
.... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1476
Identification: 0xb6b8 (46731)
Flags: 0x216c, More fragments
Fragment offset: 2912
Time to live: 64
Protocol: ICMP (1)
Header checksum: 0xalfe [validation disabled]
[Header checksum status: Unverified]
Source: 100.99.174.97
Destination: 223.2.9.124
Reassembled IPv4 in frame: 1869

Data (1456 bytes)

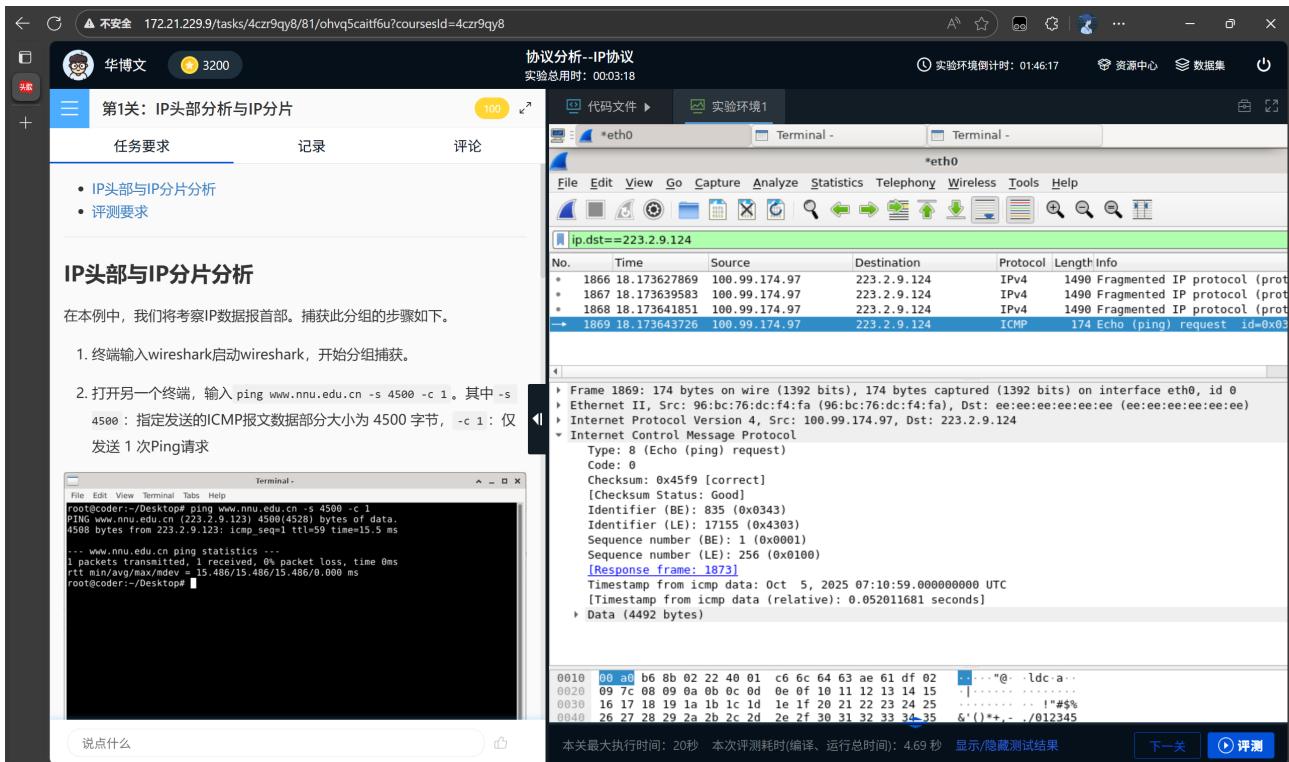
0010 05 c4 b6 8b 21 6c 40 01 a1 fe 64 63 ae 61 df 02 ... @.dc.a...
0020 09 7c 58 59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65 .XYZ\^_ abcde
0030 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 fghijklm nopqrstuvwxyz
0040 76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85 vwxxyz{}~

本关最大执行时间: 20秒 本次评测耗时(编译、运行总时间): 4.69 秒 显示/隐藏测试结果

下一关 评估

- 分片重组逻辑:** 在 3 个 IPv4 分片的详细信息中, 均显示 Reassembled IPv4 in frame: 1869, 表明 Wireshark 已将这 3 个分片在 No.1869 报文中重组为完整的原始 IP 数据报。
- ICMP 头部与数据长度分析:** 展开 No.1869 报文的 “Internet Control Message Protocol” 字段, ICMP 头部信息如下:
 - Type (类型):** 8, 标识为 Echo (ping) 请求;
 - Code (代码):** 0, 表示无附加含义;

- **Sequence number** (序列号): 1, 与 -c 1 参数对应;
- **Data** (数据部分长度): 4492 bytes, 与指定的 4500 bytes 存在差异, 原因是 ICMP 头部占用 8 bytes (Type、Code、Checksum、Identifier、Sequence number).



2 IP 数据报解析

2.1 明确解析需求

给定 IP 十六进制数据: 4500 002d 1234 0000 4001 1943 6455 77e8 df02 007b 0000 5fec 0000 0000 4920 4c6f 7665 204e 4e55 21, 需按以下规则解析并输出, 结果保存至 /home/headless/result.txt:

- **解析范围:** 分为 IPv4 头部、ICMP 头部、Payload 三部分, IPv4 头部固定 20 bytes (对应前 40 个十六进制字符), 后续为 ICMP 头部 (8 bytes, 对应 16 个十六进制字符) 及 Payload;
- **输出格式:** 需包含 “IPv4 Header” “ICMP Header” “Payload” 三大模块, 共 12 项字段, 其中 Version、Header Length、TTL、Protocol、Type、Code 为十进制值, 源 / 目的 IP 为点分十进制, Payload 为 ASCII 字符串, 字段与值用英文冒号分隔.

2.2 编写 C++ 解析代码

代码需补充部分如下:

```
int main(int argc, char **argv) {
    const std::string hexIpData =
    "4500002d1234000040011943645577e8df02097b08005f"
    "ec0000000049204c6f7665204e4e5521";
```

```

std::vector<uint8_t> ipBytes;
for (size_t i = 0; i < hexIpData.size(); i += 2) {
    std::string byteStr = hexIpData.substr(i, 2);
    ipBytes.push_back(
        static_cast<uint8_t>(std::stoi(byteStr, nullptr, 16)));
}

const uint8_t version = (ipBytes[0] >> 4) & 0x0F;
const uint8_t ihl = ipBytes[0] & 0x0F;
const uint16_t headerLength = ihl * 4;
const uint8_t ttl = ipBytes[8];
const uint8_t protocol = ipBytes[9];

struct in_addr srcIp;
srcIp.s_addr = *reinterpret_cast<const uint32_t *>(&ipBytes[12]);
const std::string srcIpStr = inet_ntoa(srcIp);

struct in_addr dstIp;
dstIp.s_addr = *reinterpret_cast<const uint32_t *>(&ipBytes[16]);
const std::string dstIpStr = inet_ntoa(dstIp);

const size_t icmpStart = headerLength;
const uint8_t icmpType = ipBytes[icmpStart];
const uint8_t icmpCode = ipBytes[icmpStart + 1];

const size_t payloadStart = icmpStart + sizeof(struct icmphdr);
std::string payloadContent;
for (size_t i = payloadStart; i < ipBytes.size(); ++i) {
    payloadContent += static_cast<char>(ipBytes[i]);
}

std::ofstream fout("result.txt");
if (!fout.is_open()) {
    return 1;
}

fout << "==== IPv4 Header ===" << std::endl;
fout << "Version: " << static_cast<int>(version) << std::endl;
fout << "Header Length: " << headerLength << std::endl;
fout << "TTL: " << static_cast<int>(ttl) << std::endl;
fout << "Protocol: " << static_cast<int>(protocol) << std::endl;
fout << "Source IP: " << srcIpStr << std::endl;
fout << "Dest IP: " << dstIpStr << std::endl;

```

```

fout << "==== ICMP Header ===" << std::endl;
fout << "Type: " << static_cast<int>(icmpType) << std::endl;
fout << "Code: " << static_cast<int>(icmpCode) << std::endl;

fout << "==== Payload ===" << std::endl;
fout << "Content: " << payloadContent << std::endl;

fout.close();
return 0;
}

```

2.3 编译与运行代码

在 Linux 终端中执行编译命令: g++ ip_parse.cpp -o ip_parse (无报错则生成可执行文件); 执行运行命令: ./ip_parse, 输出文件 result.txt 结果如下.

```

==== IPv4 Header ====
Version: 4
Header Length: 20
TTL: 64
Protocol: 1
Source IP: 100.85.119.232
Dest IP: 223.2.9.123
==== ICMP Header ====
Type: 8
Code: 0
==== Payload ====
Content: I Love NNU!

```

四 实验中遇到的问题及体会

实验初期, 在对 IPv4 分片的观察中发现了长度差异的疑问: 筛选出的所有 IPv4 分片长度均为 1490 bytes, 而原始 ping 请求中明确指定数据部分为 4500 bytes, 这使得分片数量与长度之间的对应关系难以理解. 经过深入分析后得知, 以太网的 MTU 通常为 1500 bytes, IP 数据报在传输时需满足“IP 头部 (20 bytes) + 数据部分 \leq 1500 bytes”的条件, 据此推算每个分片的数据部分最大应为 1480 bytes, 但实际通过抓包工具观察到的数据部分却为 1456 bytes (这一数值与分片偏移以 8 字节为单位的规则相符). 进一步计算发现, 3 个分片的数据部分总和加上 ICMP 头部的长度为 4376 bytes, 与预期的 4500 bytes 仍存在差异, 继续排查后才找到原因——数据链路层头部 (14 bytes) 占用了部分长度, 因此在计算时需要结合帧的整体长度进行综合考量.

在解析指定十六进制 IP 数据报的过程中, 曾出现字段偏移错误的问题: 初期误将 ICMP 头部的 Type 字段偏移定位到第 43、44 个字符, 直接导致了解析结果的错误. 为修正这一问题, 团队对照了 IPv4 头部的规范——IPv4 头部固定长度为 20 bytes (对应 40 个十六进制字符), 据此重新确认 ICMP 头部应起始于第 41 个字符, 调整偏移位置后, 再次进行解析得到了正确的结果.

通过此次实验,对网络分层模型的认知得到了深化。课本中“IP 协议工作于网络层,负责路由与分片”的理论知识,在实验操作中变得更加直观具体: IP 数据报无法直接传输,必须封装在数据链路层的帧结构中,且会因受 MTU 限制而进行分片;接收端则需要通过 IP 数据报中的标识 (Identification) 字段和分片偏移字段,将多个分片重新组合成完整的数据报,这一整个过程清晰地体现了网络模型“分层解耦、各司其职”的设计思想。同时,也进一步理解了 ICMP 协议作为网络层辅助协议的特性——它本身不具备独立传输能力,必须依赖 IP 协议进行封装后才能完成数据传输,两者之间的关联关系不再是抽象的概念。

实验同样让我们认识到协议规范的核心意义。在解析 IP 数据报的过程中,严格遵循各项协议规范是确保解析成功的关键,例如 Version 字段占 4 位、Header Length 字段以 4 字节为单位、分片偏移字段以 8 字节为单位等规则,任何一项的忽视都可能导致解析失败。以 Header Length 字段为例,若误将其单位当作 1 字节来计算,会直接导致后续所有字段的偏移位置全部出错,无法正确读取数据。这一经历也印证了网络设备之间能够实现互通的基础,正是对严谨协议规范的共同遵循,任何字段在格式或长度上的微小偏差,都可能导致数据解析的彻底失败。

此外,问题排查能力在实验中得到了显著提升。从最初对 IP 分片长度差异的疑惑,到结合 MTU 数值、IP 头部长度进行理论计算,再通过 Wireshark 工具查看“Reassembled IPv4 in frame”选项验证分片重组逻辑;从解析 IP 数据报时出现的字段偏移错误,到对照 IPv4 协议格式重新修正字段定位,整个过程遵循“发现问题—理论分析—验证解决”的逻辑闭环。这种主动排查问题的过程,比单纯按照步骤执行实验更能加深对知识的理解和掌握,同时也让我们熟悉了“结合协议规范 + 关注工具细节”的网络实验调试思路,为后续类似实验积累了宝贵经验。