**题目：网络层IP协议设计与实现**

**姓名：王怡恒**

**学号：2022300385**

**班号：10012201**

**时间：2024/11/12**

**计算机学院**

**目 录**

[1、实验目的 1](file:///D:\\qq\\3229804499\\filerecv\\西北工业大学试点班实验报告模板.doc" \l "_Toc341302685)

[2、实验要求 1](file:///D:\\qq\\3229804499\\filerecv\\西北工业大学试点班实验报告模板.doc" \l "_Toc341302686)

[3、实现原理及流程图 1](file:///D:\\qq\\3229804499\\filerecv\\西北工业大学试点班实验报告模板.doc" \l "_Toc341302688)

[4、算法设计 3](file:///D:\\qq\\3229804499\\filerecv\\西北工业大学试点班实验报告模板.doc" \l "_Toc341302689)

[5、运行结果与分析 5](file:///D:\\qq\\3229804499\\filerecv\\西北工业大学试点班实验报告模板.doc" \l "_Toc341302690)

6、实验过程发现的新技术问题及建议................................7

[7、参考文献 8](file:///D:\\qq\\3229804499\\filerecv\\西北工业大学试点班实验报告模板.doc" \l "_Toc341302691)

## 实验目的

1. 学习理解网络层协议，了解IP协议工作原理。
2. 实现IP分组的构造与发送，包括分片机制和校验和计算，以及数据链路层的封装过程。
3. 学习IP层和以太网的交互，实现数据链路层和网络层数据的交换。
4. 掌握IP分组的接收与处理，包括分片的识别、重新组装和上层协议数据的正确传递。
5. 分析并解决IP分组传输中的问题，如分片完整性检测和路由器转发时首部校验和的重新计算。

## 实验要求

1. 功能1-发送方：构造一个IP分组，支持分片，计算简单校验和，并交付给数据链路层（封装成数据帧并发送）

（2） 功能2-接收方：从数据链路层接收到一个IP分组，检查目的IP地址是否匹配，检测校验和；如果分片需要重新组装；对IP分组首部进行解析并打印，根据协议类型，将IP分组数据部分交付给上层协议（写到一个文件中）

## 实验原理

1. IP分组的构造与发送：

发送端首先构造IP分组，包括基本首部和可变首部。基本首部包含版本、首部长度、服务类型、总长度、标识、标志（包括DF和MF标志）、片偏移、生存时间（TTL）、协议类型、源IP地址和目的IP地址等字段。如果数据长度超过MTU（最大传输单元），则进行分片处理，每个分片都包含完整的首部信息，并且通过设置MF（More Fragment）标志来指示后续分片的存在。发送端还需要计算并添加校验和以确保数据的完整性。

1. IP分组的接收与处理：

接收端从数据链路层接收IP分组，并首先检查目的IP地址是否匹配。如果匹配，继续检查校验和以验证数据的完整性。如果数据报文被分片，接收端需要使用一个缓冲区数组和指针来模拟队列，以便收集所有分片，并在所有分片到达后进行重组。重组过程中，接收端需要根据标识、片偏移和MF标志来正确地组装原始数据报文。如果在规定时间内未能收到所有分片，或者校验和检测失败，则可能需要调用ICMP协议来报告错误。

1. 层次结构的设计：

实验采用层次结构设计，发送端从主函数开始，通过调用network\_ipv4\_send()函数，并传递句柄参数，实现多次封包只打开一次设备的目的。该函数进一步调用ethernet.h中的函数服务，句柄逐层向下传递，直到底层直接发送数据。接收端同样采用层次结构处理，由ethernet.h中的回调函数调用network\_ipv4\_recv.h中的函数，处理分片的接收和重组。

1. 实验效果验证：

实验通过发送和接收不同大小的数据报文（有分片和无分片的情况）来验证IP协议的实现效果。发送端生成的数据报文通过网络传输到接收端，接收端正确接收并重组数据报文，最终验证数据的完整性和正确性。实验结果通过命令行结果和文件比较来展示，确保数据在传输过程中的准确性和完整性。

1. 具体IP分组设计：

（1） IPV4： 0100;

（2） 首部长度：0101-1111；

（3） 服务类型：IP分组优先级别: 3 bit;8个优先级别，0最小，7最大；D:低延迟（delay）;T:高吞吐量(throughput);R:高可靠性(reliability:被路由丢弃概率较小);C:选择代价更小路由；ToS只是用户要求，对网络并不是强制，路由器进行路由选择等处理时仅仅作为参考。

（4） 总长度： 0-65535；单位字节

（5） 标识： 通过一个计数器实现；

（6） 标志：占 3 bit，目前只使用两个bit：0+DF+MF.

DF(Don’t Fragment), DF=1表示”不允许分片”,DF=0表示”允许分片”.

MF(More Fragment),MF=1表示后面有分片组;MF=0表示这是分片后最后一个分组.

（7） 片偏移：12 bit；指出较长分组，主要指数据部分在分片后，某一个分片(或分组)的数据部分第一个字节在原始数据中的相对位置；片偏移以8个字节为偏移单位。

（8） 生存时间 (TTL :Time To Live): 8 bit

（9） 协议：上层协议号（TCP = 6,UDP = 17，ICMPV4=1, IGMPV4=2）

（10）源IP地址：32比特

（11）目的IP地址：32比特；

## 算法设计

**发送方发送流程（注意计算校验和）：**

接收到上层协议数据单元作为数据（读文件）；

IF（数据长度小于等于1400个字节）

构造一个IP分组(20个字节基本首部+40个字节可变首部：全0 + 数据)；

IF（数据长度大于1400个字节）

构造多个分片，每个分片是一个小的IP分组;

交付一个IP分组或多个分片给数据链路层；

回到开始，接收到上层协议数据单元作为数据（读文件）；

**接收方接收流程：**

从数据链路层接收到IP分组；

IF （目的IP地址是本地IP地址或者广播地址）

接收

ELSE 丢弃；

IF (校验正确)

接收;

ELSE 丢弃（调用ICMP协议）；

将接收的分组（或分片）首部解析在屏幕上显示。

IF 该IP分组是一个IP分组（MF = 0，片偏移=0）

解析出数据交上层协议处理（写入文件）

IF该IP分组是一个IP分片

接收各个分片，如果分片在30秒内成功接收到，则组装数据，将数据按照协议类型交付上层（写入文件）； 否则，所有分片全部丢弃（调用ICMP协议）。

**Send：**

Main.cpp：和第一次选网卡类似。

Ethernet.cpp：和第一次计算CRC，构造IP分组类似。

Network\_IPV4\_send.cpp：

calculate\_check\_sum（）：算IPv4数据包头部的校验和。校验和是一个16位的数字，所以len每次-2。

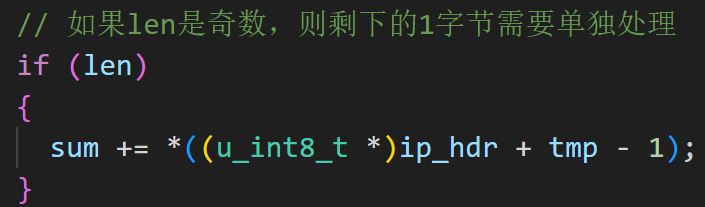


图 1 奇数情况单独考虑最后1字节

load\_ip\_header（）：初始化一个 IPv4 数据包的头部。

load\_ip\_data（）：数据填充到 IP 数据包的有效负载部分。

network\_ipv4\_send（）：将文件内容分片并发送作为多个IPv4数据包，注意0x0000代表无分片，0x2000代表有分片。

**Receive：**

Main.cpp：和第一次选网卡、捕获数据包并调用回调处理函数类似。

Ethernet.cpp：前半部分和第一次计算CRC、匹配并输出目的mac地址类似，主要我在回调函数里加了下面的语句，使其有所分割：

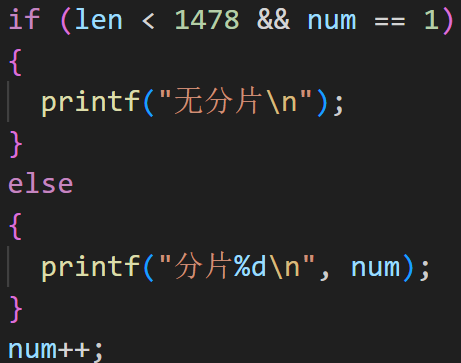


图 2 打印分片信息

Network\_ipv4\_recv.cpp（）：

calculate\_check\_sum（）：计算检验和与头部中的信息比较看是否丢失或者出错。

is\_accept\_ip\_packet（）：判断是否是发往自己的数据帧，是的话计算检验和并看TTL是否为0。

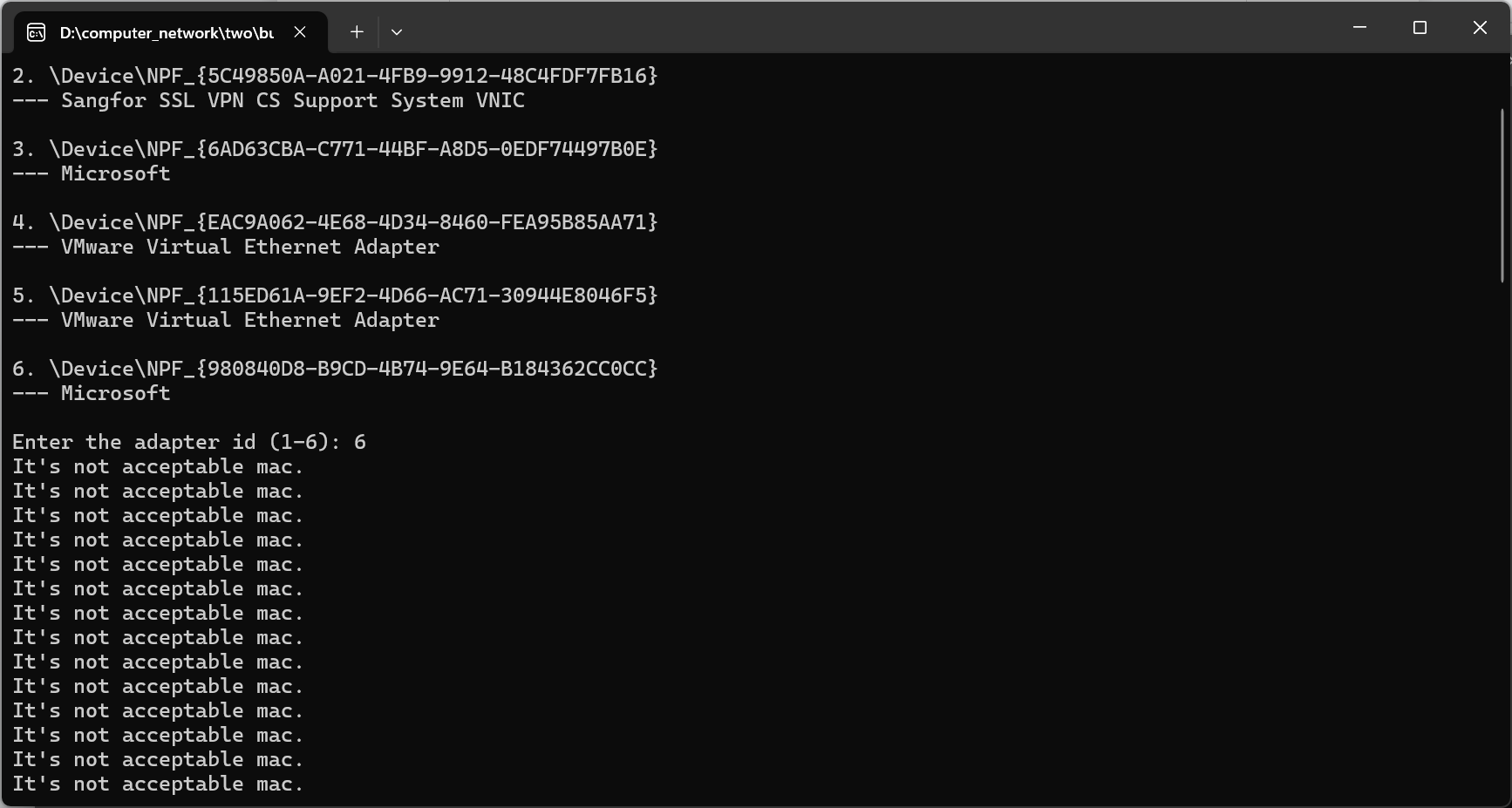
load\_data\_to\_buffer（）：先把接收到的数据放在缓冲区。

load\_data\_to\_file（）：将缓冲区的数据写入文件。

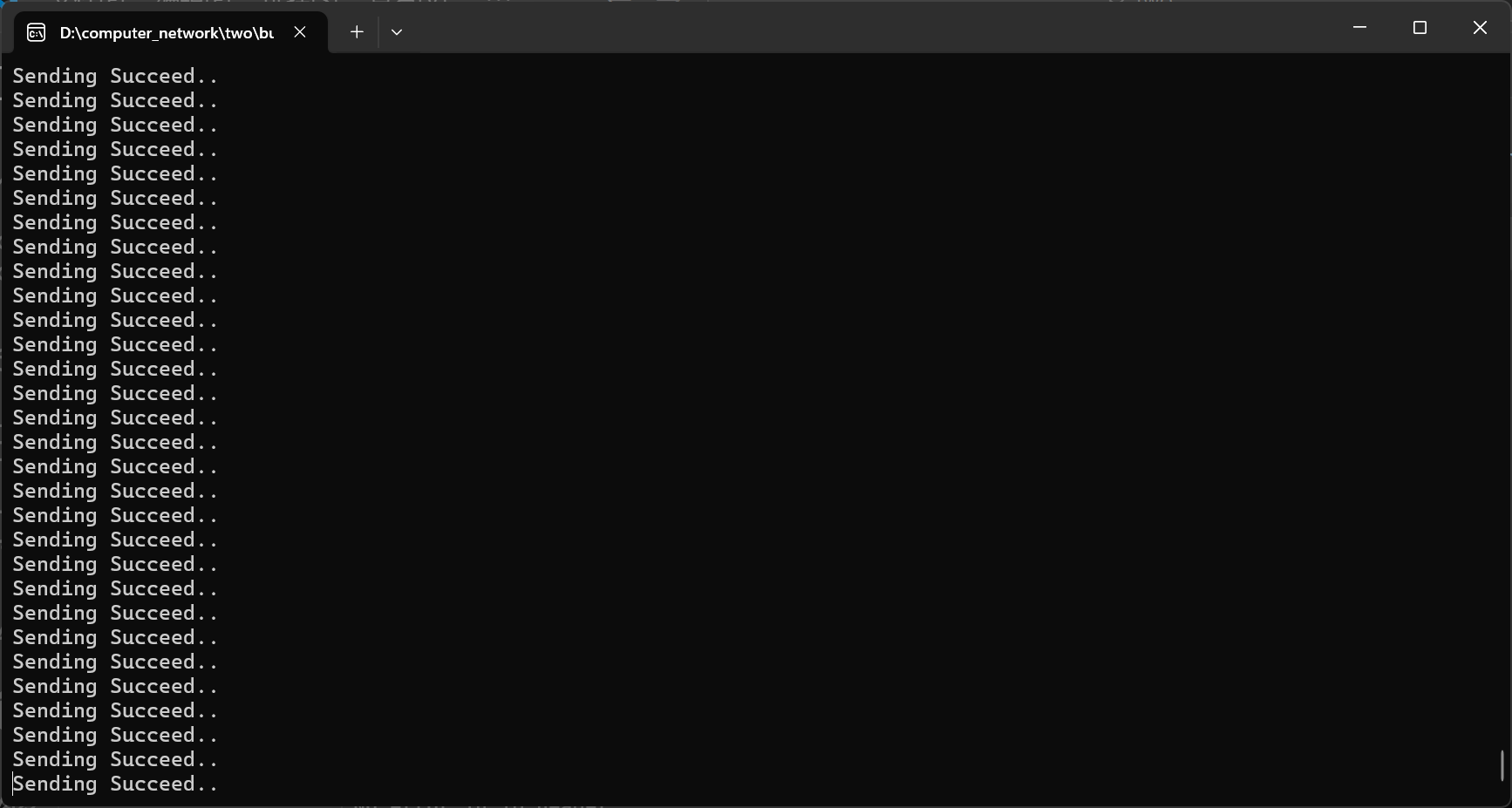
network\_ipv4\_recv（）：解析 IPv4 数据包头、验证数据包是否可接受、时间间隔检查，确保不超过30s、判断数据包是否为分片包。如果是分片包并且与之前的 ID 一致，将其数据存入缓冲区；如果是最后一个分片或没有更多分片，将缓冲区中的完整数据写入文件，并重置相关变量、输出调试信息。

## 实验结果

在vscode的CMake下完成，运行结果如下：

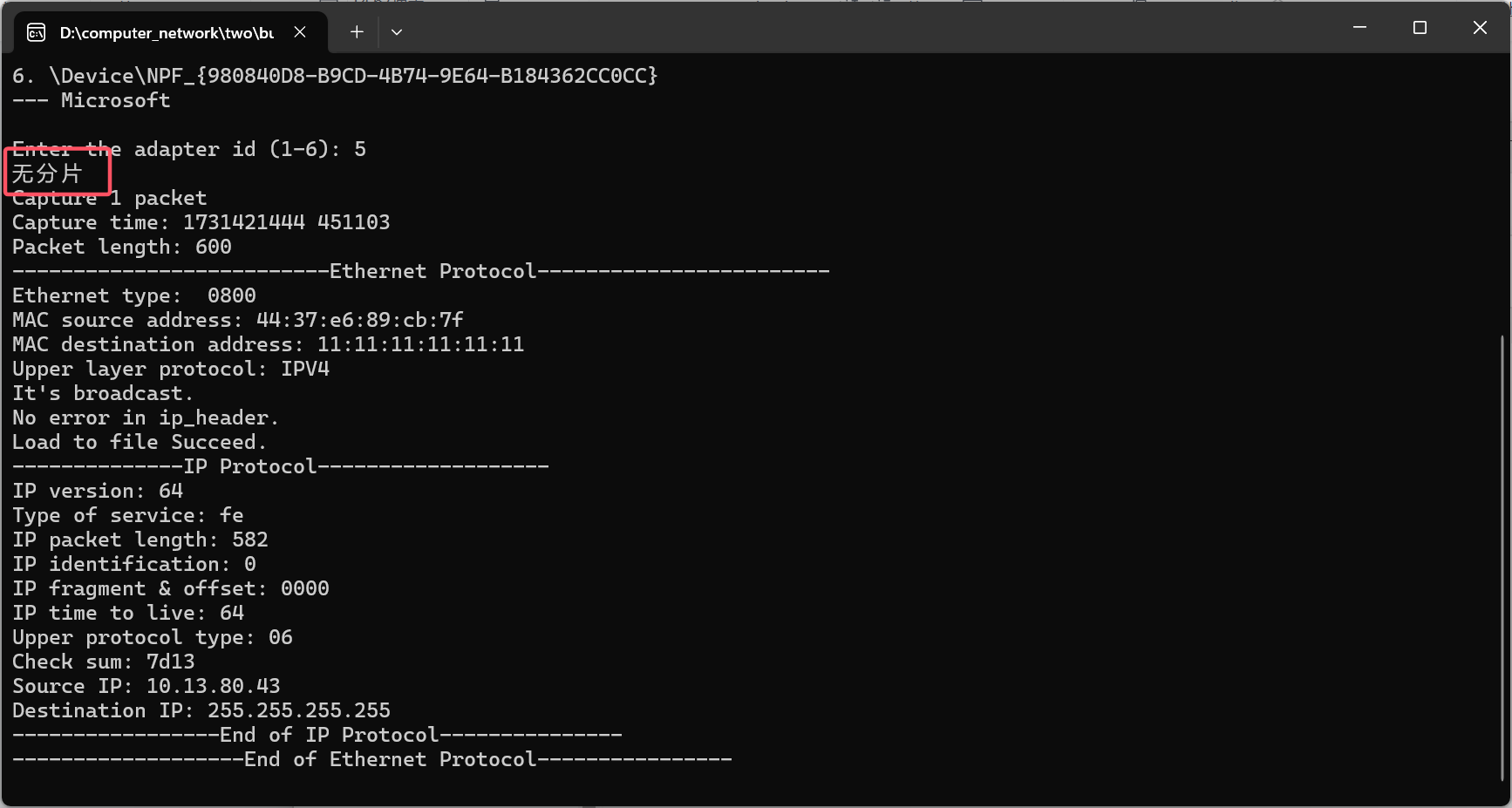


此时正在等待发送的数据报，所以命令行中只显示“It is not acceptable mac”,如果设计不停发送，此时运行send，结果如下：

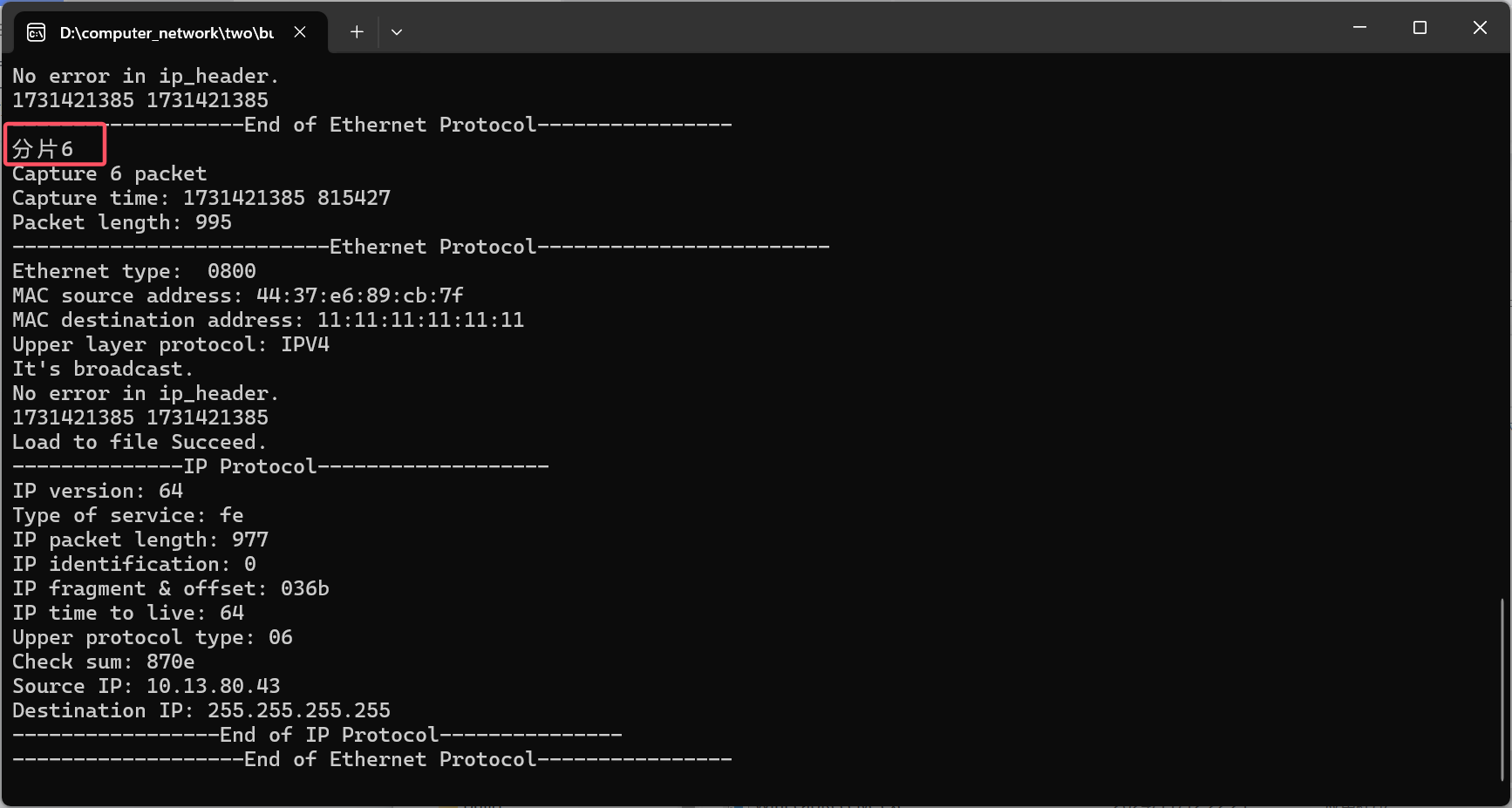


如果把while（1）去掉，只发送1次，那么send终端输入网卡号后停止，receive终端接收信息如下：

无分片：



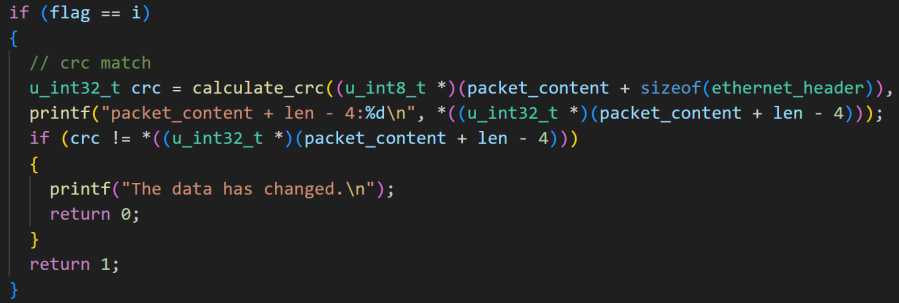
有分片：



包括捕获数量、捕获时间、数据报长度、协议类型IPv4、mac源地址、mac目的地址、成功提示以及与IP协议相关的内容。

## 实验过程发现的问题

1. 由于devc++无法进行多文件编译，导致第一次成功的环境这次无法使用，于是改变到vscode中使用CMake来实现，下载好CMake后总是出现找不到对应目录，在沈天乐同学的帮助下修改了CMakeList.txt的代码得以成功，梁乐彬同学也教了我CMake的使用方法。
2. 初始代码接收端是收不到信息的，后来逐步发现是由于在几个函数中没有写return导致的，加上之后就可以收到。
3. 当数据过长时，分片过多会导致接收端查看不便，于是增加“分片x”的打印信息将其分割，结果就很清晰明了。
4. 发现有的网口接收端选择后会显示“It is not acceptable mac”，有的就啥也没有等待发送端的发送。
5. 原始代码有点问题，在Ethernet.cpp中的is\_accept\_ethernet\_packet（）函数，原代码的CRC检验在目的MAC地址检验之后，但是MAC匹配后会直接return 1出去，无法进行后续的CRC检验，于是我把CRC检验部分放在了判断MAC匹配的if中，如下：



但是这样之后有了新的问题——CRC检验无法通过，我把传入calculate\_crc（）的参数打印出来后发现len是个负数，导致计算函数中的for循环进不去，于是我看len的生成逻辑：

int len = strlen((const char \*)packet\_content);

但是strlen会因为读取到数据中的“\0”导致长度计算提前终止，导致len计算错误，于是我就把packet\_header->len赋值给len，这是真正的数据帧的长度，这样就能成功接受到数据帧。

## 参考文献

《第二节课程-网络层IP协议设计与实现》

《IP数据报文发送与接收》

《cmake使用详细教程-CSDN》