# 实验报告与软件说明

基于C++和QT5实现了带Gui的网络嗅探器

**软件使用**

github链接：https://github.com/csapp1024/pyh\_sniffer.git

环境：Ubuntu20.04

安装库依赖：

sudo apt install qtchooser qt5-default libpcap-dev libboost-all-dev libqt5charts5-dev

运行：

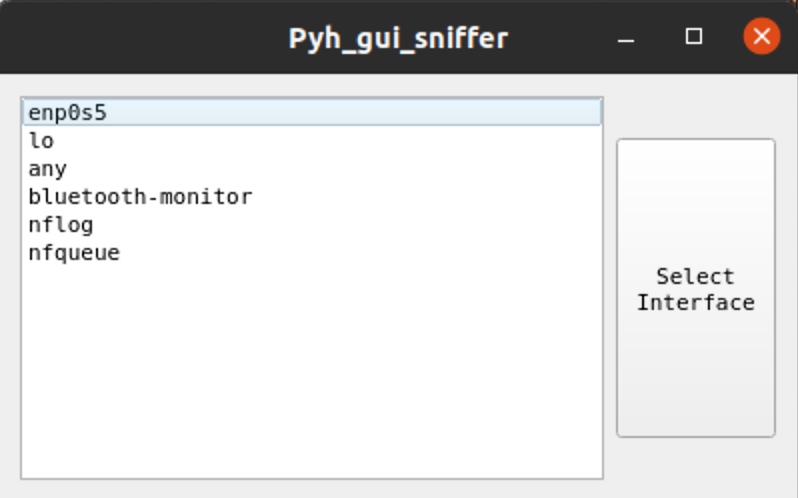
cd /Pyh\_gui\_sniffer/build

sudo ./Pyh\_gui\_sniffer

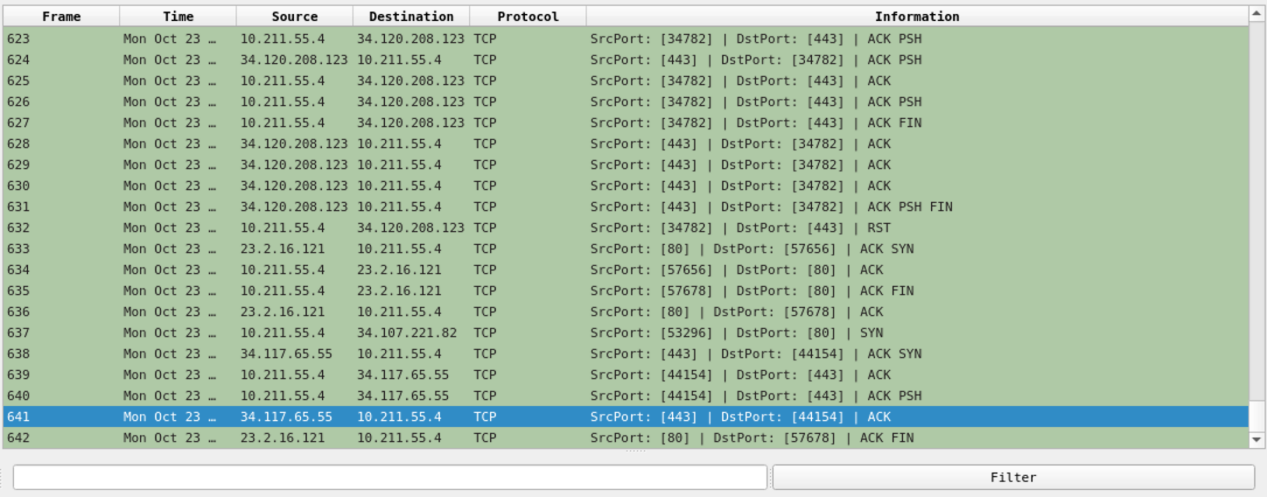
源码及qt 的pro文件均在src文件夹

**抓包以及包的各层的协议分析功能：**

1. 基于libpacp在线实时抓取网络上的包（NPS），提供系统的所有网络接口，可以选择不同的网络接口进行抓包



1. packet分析功能：程序在抓到包的同时就会从数据链路层开始分析，设计的GUI总体状态栏信息 的如下：



frame为抓到的包的序号，然后是抓包时间，源IP地址，目的IP地址，最上层协议，概括信息（主要是TCP协议里的源、目的端口以及标识位）

1. 数据链路层：

为packet的最开始，因此可通过

this->ether\_header = (struct ether\_header\*)(this->data);

获得指向以太网帧的头部起始指针，根据以太网帧的头部结构，包括源MAC地址，目的MAC地址，以及Ether Type字段（通常指上层，即网络层的协议）在系统自带的netinet/if\_ether.h头文件中有头部的定义：

struct ether\_header {

unsigned char dhost[6]; // Destination MAC Address

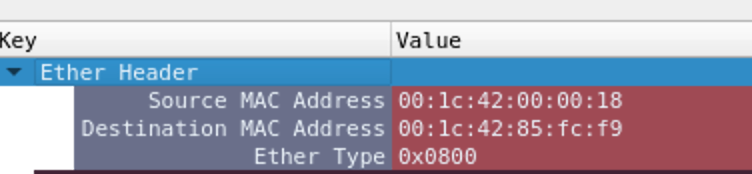
unsigned char shost[6]; // Source MAC Address

unsigned short type; // Encapsulated Protocol Type

// htons(type); to convert

};

因此通过头部结构很容易分析输出以太网的三个字段信息，如下设计的输出：



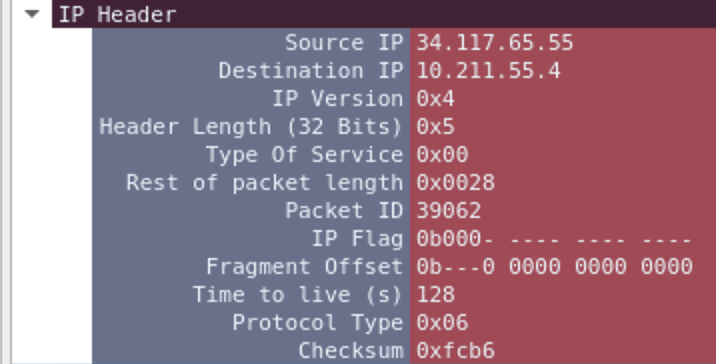
常见的有0x0800代表IP协议，0x0806代表ARP协议

1. 网络层（IP）：

网络层的报文段紧贴以太网帧的结尾，所以通过将指针偏移以太网帧的头部大小（无负载，默认14字节）来获取起始处：

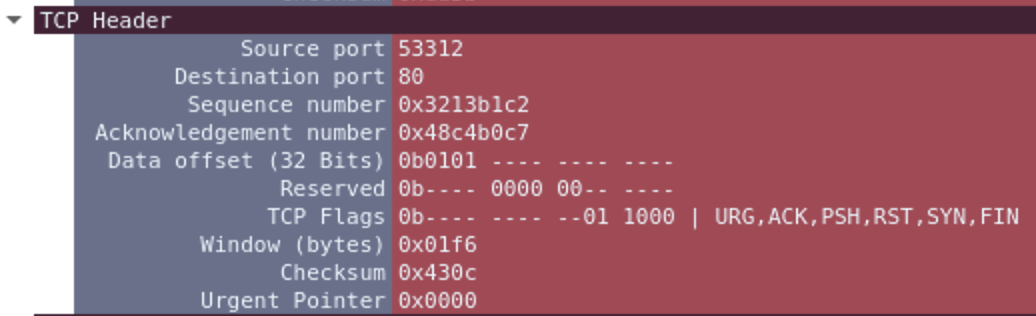
this->ip\_header = (struct ip\_header\*)(this->data + sizeof(struct ether\_header));

分析IP头部各部分，同样，在系统头文件中netinet/ip.h有ip头的结构体定义，可直接引用并输出各个字段，注意有些标识位为特定的几位需要进行位操作，通过设计的输出如下：



1. 传输层（TCP）：

分析类似以上，通过tcp头部给出的结构体输出各个字段：



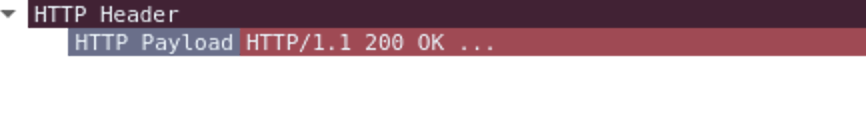
1. 应用层（http）：

本次设计主要针对http做分析，通过将指针偏移到TCP结尾处，通过以下判断：

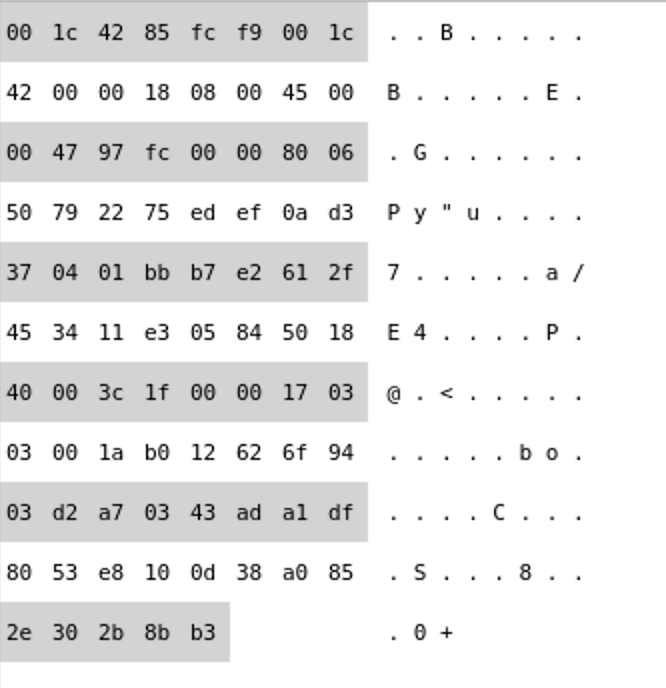
if (strncmp("HTTP", payload, 4)== 0 || strncmp("GET", payload, 3) == 0 || strncmp("POST", payload, 4) == 0 || strncmp("PUT", payload, 3) == 0 || strncmp("DELETE", payload, 6) == 0) {

来匹配http协议

仅输出第一行负载的内容

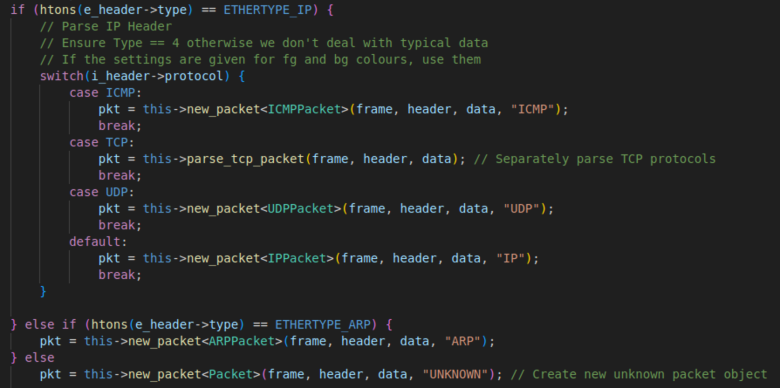


1. 本次设计除了将各层协议头部信息分析提取并输出外，也将整个包的编码（十六进制）以及每个字节对应的ASCII输出，具体如下：

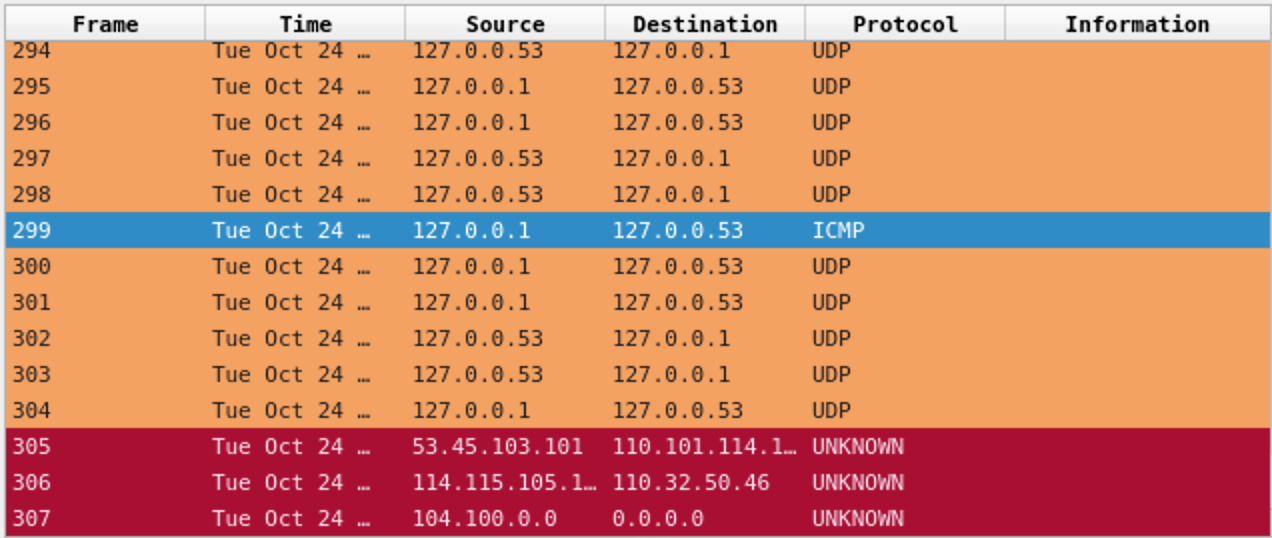


数据从以太网帧开始一直到最末尾。

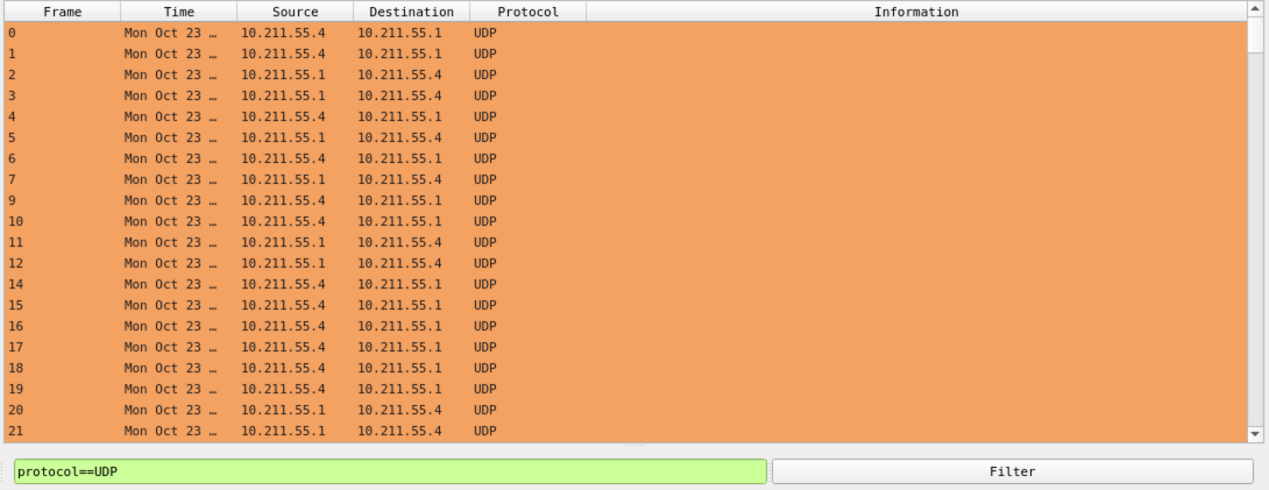
1. 其他情况的判断：在本次设计中，考虑的协议有 以太网，IP，ARP，TCP，UDP，ICMP，HTTP。由于协议众多，因此没有完全分析抓到的包的各种协议。从数据链路层开始逐层考虑上层协议（各头部字段可以看到）



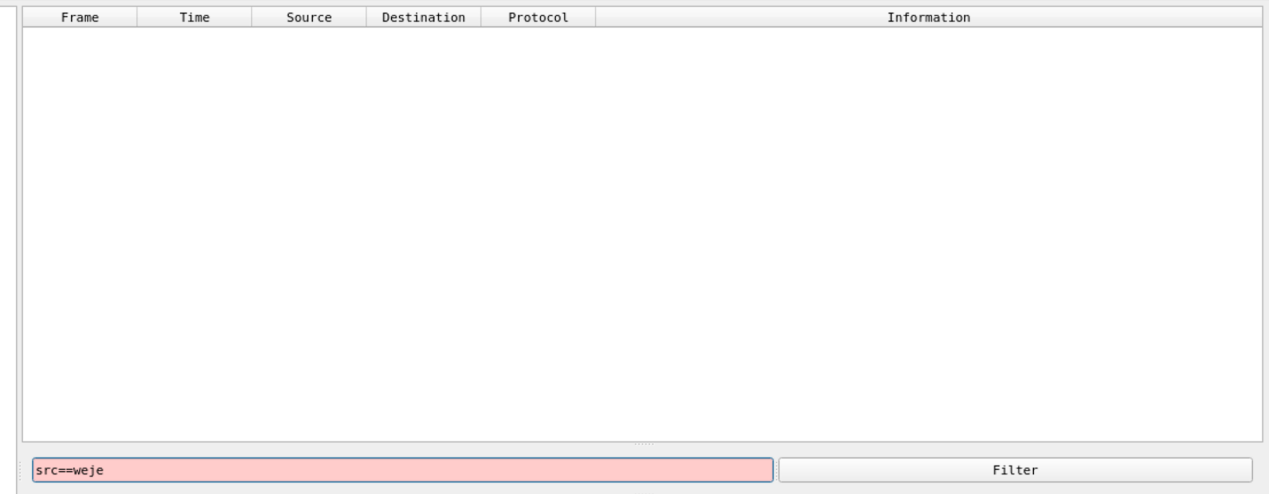
当网络层不是IPv4，ARP，传输层不是TCP/UDP时，把网络包的协议当做“UNKNOWN”处理（已分析的协议层信息仍然输出）如：



**过滤功能：**过滤功能主要是应用BPF语法，在libpcap库中，以使用pcap\_compile()函数来编译过滤表达式，然后传递给pcap\_setfilter()函数来应用过滤条件。如：protocol==TCP/UDP，即可在捕获的包中过滤出协议为TCP/UDP的包

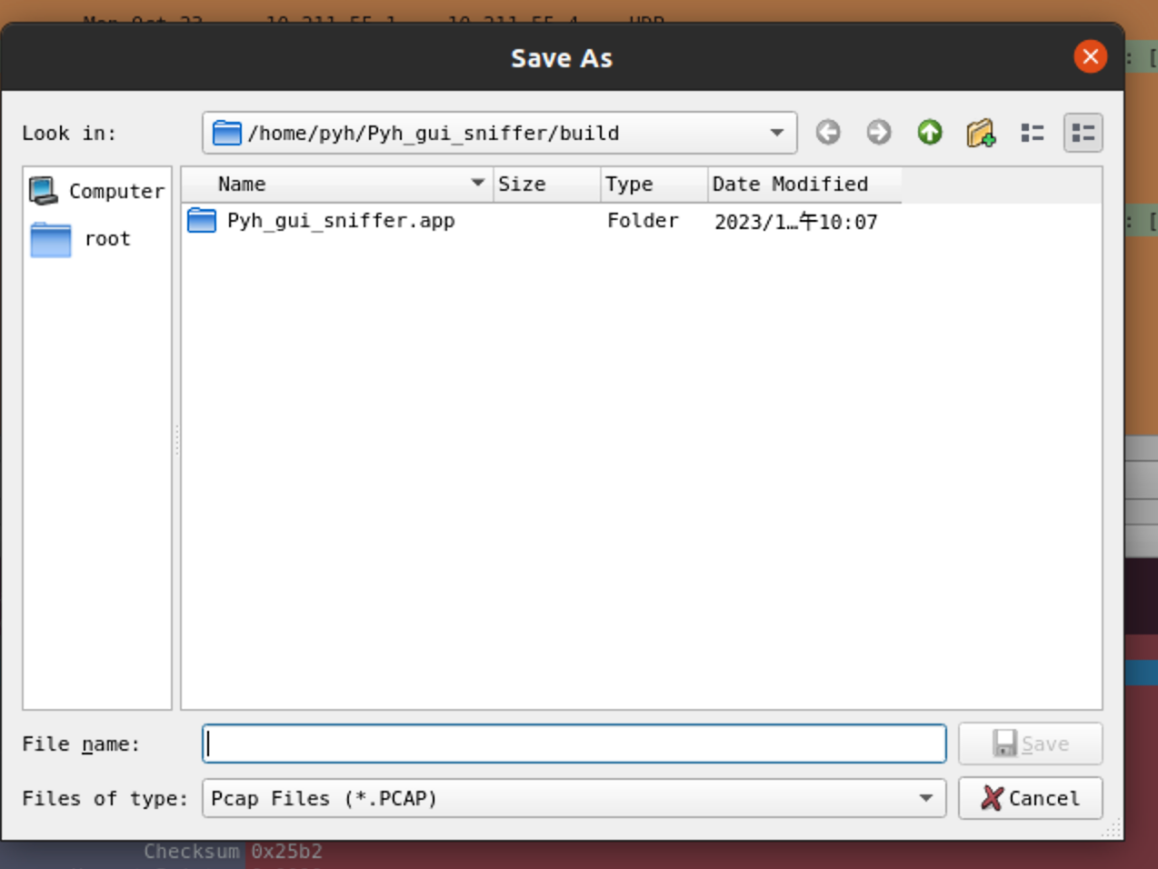


在本次设计中，所支持的过滤表达为，protocol==（）来过滤出某层协议的包，length==/>=/<=（）来过滤出某些你想要的长度范围内的包，contains==（）来过滤包含特定字符串的包，另外，当输入的语法不符合设计的定义时，过滤无效且输入框为红色



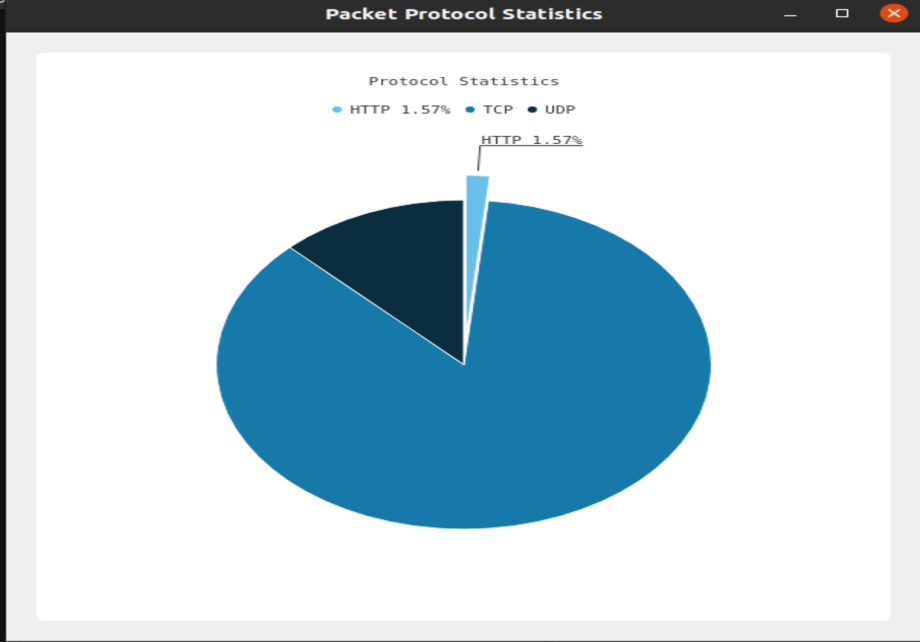
**其他的一些功能实现：**

**1）**添加了一些额外的功能，如能将抓到的包的所有数据保存为一个文档



当然，也可以将保存的文档打开继续分析

**2）**实现了一个饼状数据图，将抓到的包根据协议类型，计算每个类型的协议占总捕获包的百分比，示例如下：



3）课程设计要求均已完成，其额外的功能还有待完善。