Lab1A 实验报告

1. 项目简介

1.1 实验环境

Npcap_SDK_1.12，QT6.6.0， Qmake

1.2 实现功能

基于 QT6 实现的仿 WWireshark 的 GUI，实现了网卡选取、网络抓包、网络协议解析功能，支持 UDP、ICMP、TCP、ARP、DNS、TLS(部分)协议的分析

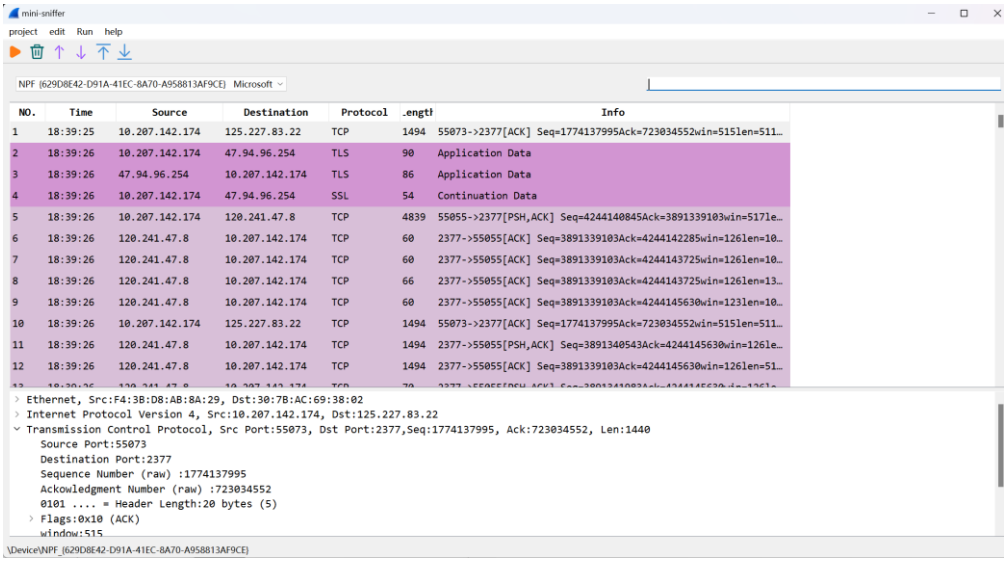


图 1-1 项目 demo

1.3 源码地址

OnceSh/mini-sniffer (github.com)

2. 项目架构及功能

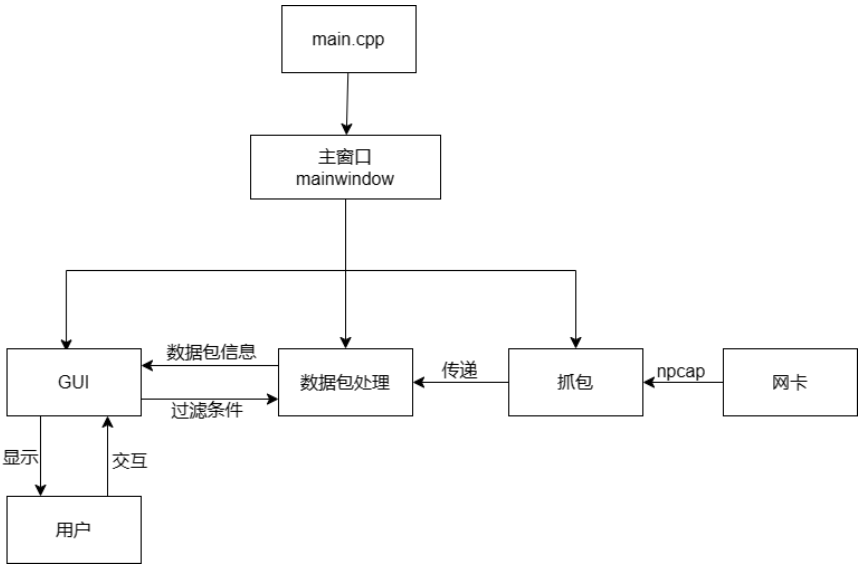
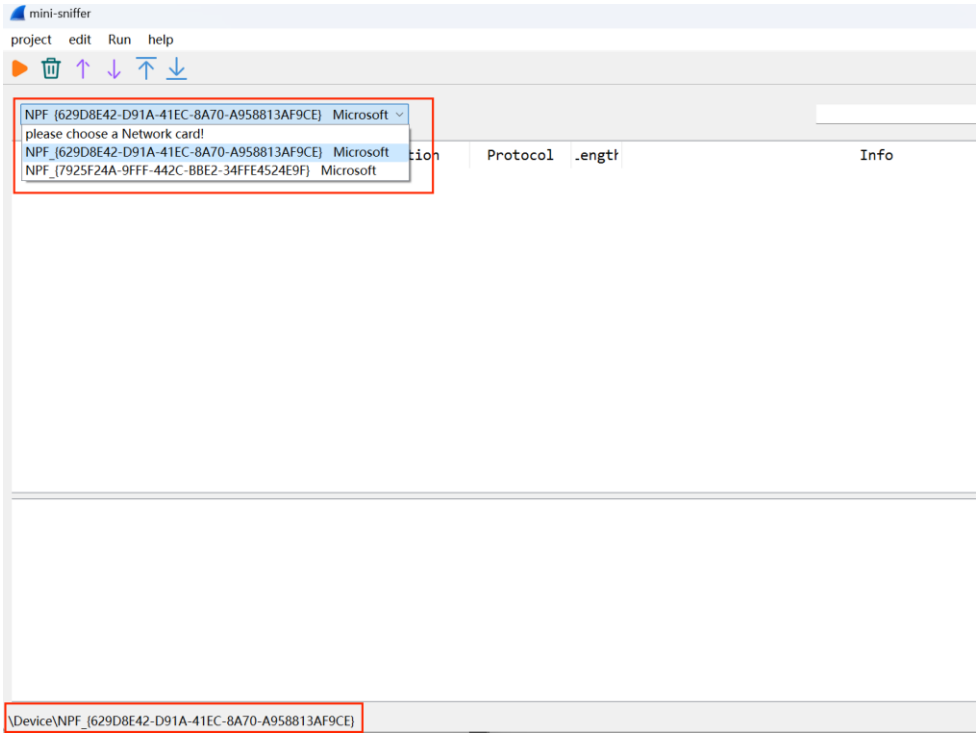


图 2-1 sniffer 架构图

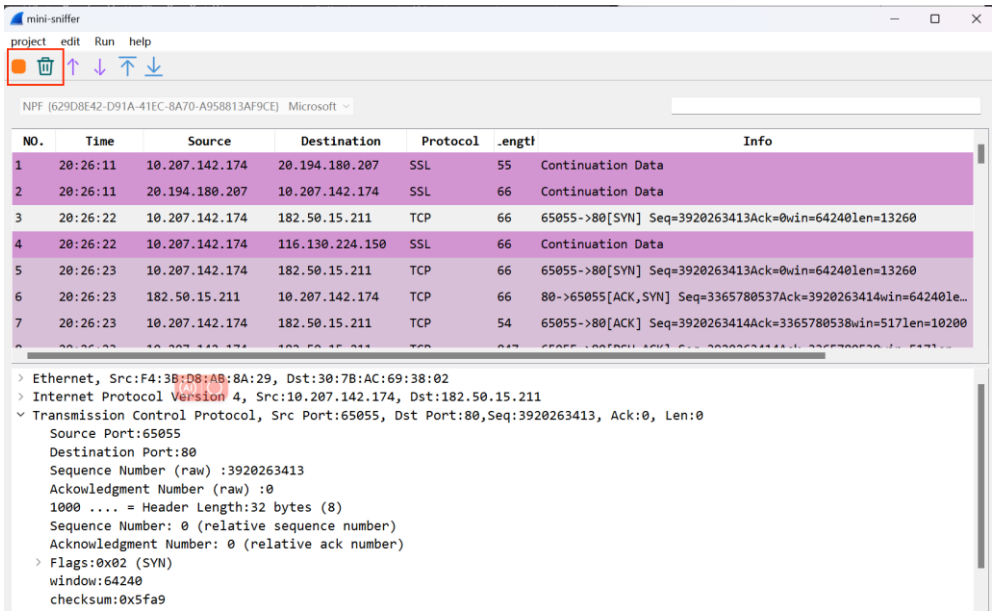
其中，QT 编写的 GUI 负责将网卡选取、过滤规则等用户操作传递给数据包处理程序；选取网卡并开始抓包后，将通过抓包线程调用 npcap 对应函数抓获数据包，并将数据传递给数据包处理程序/

2.1 网卡选取



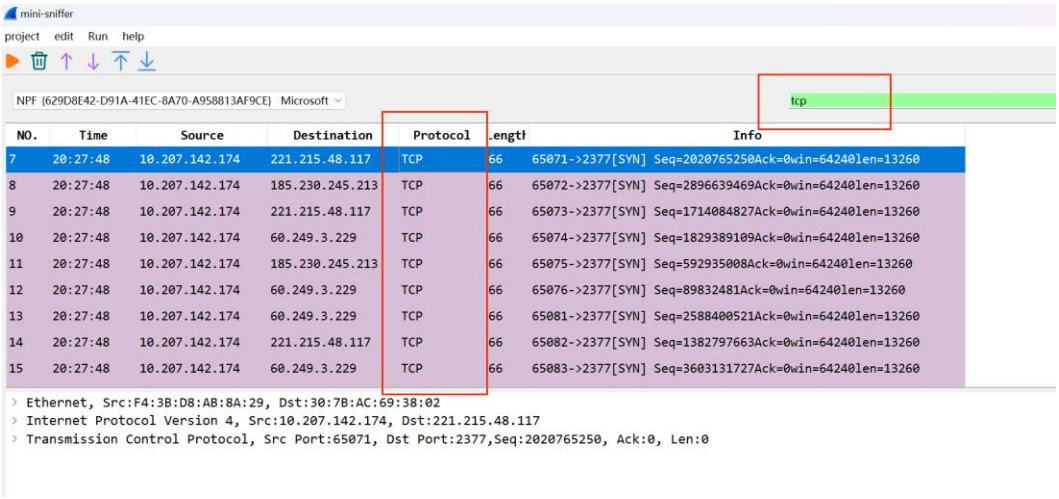
选取之后会在页面左下角显示目前抓包的网卡

2.2 数据包抓取、停止与清除



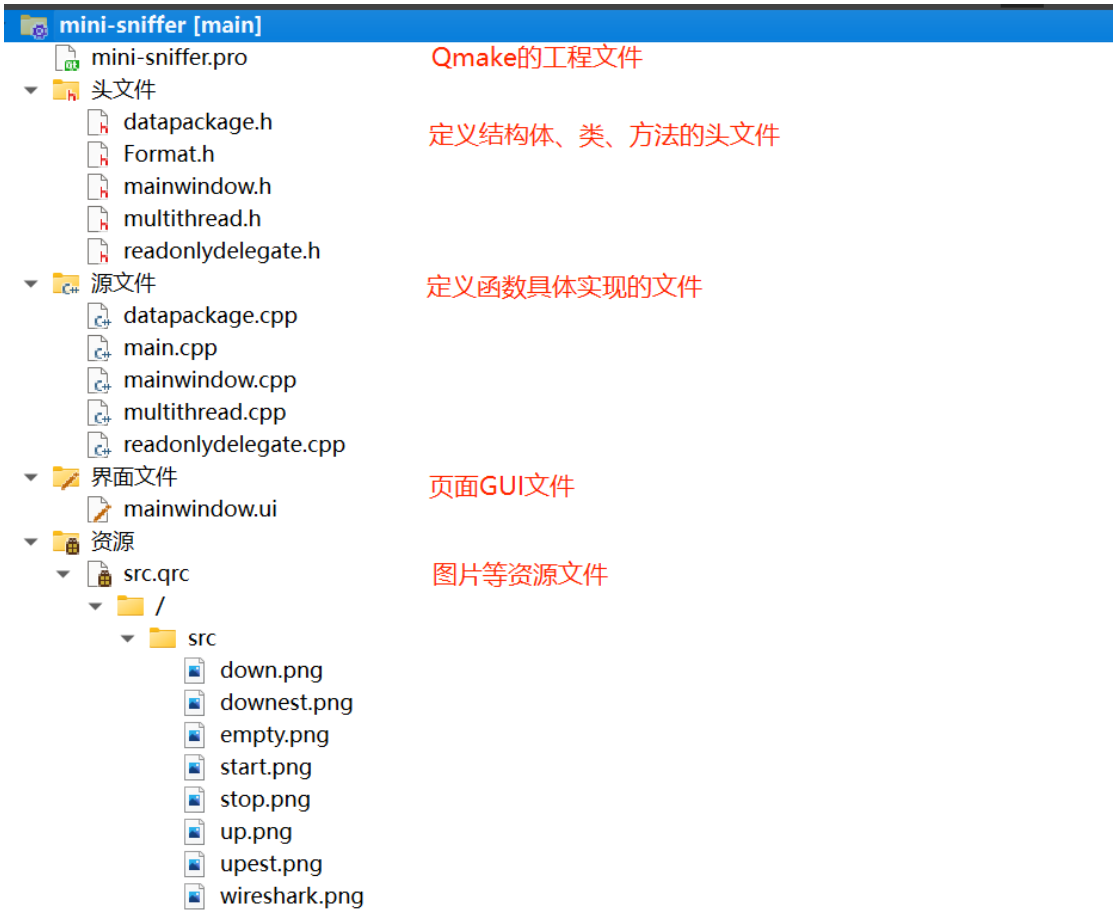
如图，进行抓包后，页面上显示了部分协议的内容及协议树解析

2.3 数据包过滤



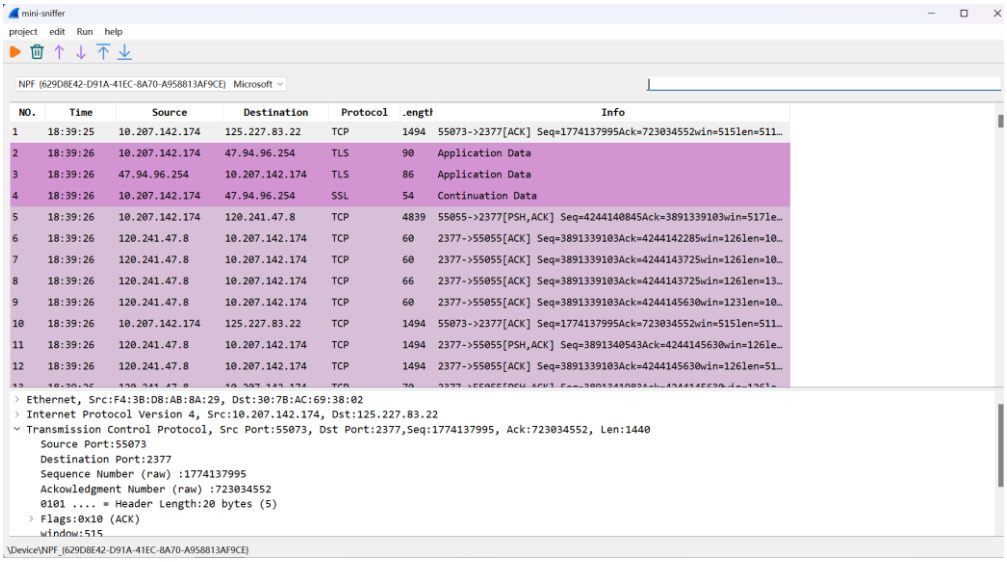
右上角的输入作为过滤条件，按下回车键即可进行过滤；过滤过程必须暂停抓包才行；过滤字段仅支持协议类型

3．代码实现



代码架构如上图所示，项目整体为 QT6 的 QT Widgets Application，使用 Qmake 作为构建系统，MinGW 作为编译工具

3.1 GUI 设计



在某一版本之后，Wireshark 的 Windows 版本 GUI 就改为了 QT 制作，但其 GUI 却是一个外壳，用来将用户输入和内核的 tshark 进行交互。

在本项目中，借鉴了 Wireshark 的设计，根据展现方式使用对应控件，主要内容存在于 mainwindow.ui 文件中。

选择网卡：comboBox 控件，可以有下拉列表，进行选取

过滤规则：lineEdit 控件，单行输入，简单文本

数据包信息：tableWidget 控件，多行输出，且每行都有多个单元

协议树信息：treeWidget 控件，树状显示各协议信息

工具栏：toolBar 控件，显示启动/暂停、清除操作的按钮

分裂器：应用于 tableWidget 和 treeWidget，可以任意调整两者中间的分割线

3.2 协议数据包报头格式

在 Format.h 中存储了各种协议的数据包结构体，包括 Ethernet、IPv4、ARP、TCP、UDP、ICMP、DNS、TLS、SSL 协议。下面以 TCP 报头结构体为例，其余相同。

```
// Tcp header
/*
+----------------------+----------------------+
|         16 bit       |       16 bit        |
+----------------------+----------------------+
|      source port     |  destination port   |
+----------------------+----------------------+
|              sequence number                |
+----------------------+----------------------+
|                 ack number                  |
+----+---------+-------+----------------------+
|head| reserve | flags |     window size      |
+----+---------+-------+----------------------+
|     checksum         |   urgent pointer     |
+----------------------+----------------------+
*/
typedef struct tcp_header{    // 20 byte
    u_short src_port;         // source port [2 byte]
    u_short des_port;         // destination [2 byte]
    u_int sequence;           // sequence number [4 byte]
    u_int ack;                // Confirm serial number [4 byte]
    u_char header_length;     // header length [4 bit]
    u_char flags;             // flags [6 bit]
    u_short window_size;      // size of window [2 byte]
    u_short checksum;         // checksum [2 byte]
    u_short urgent;           // urgent pointer [2 byte]
}TCP_HEADER;
```

如上图，根据对应 TCP 报文格式，定义相应长度的数据类型，unsigned char 为 1 字节、unsigned short 为 2 字节、unsigned int 为 4 字节。

3.3 数据包类及对应方法

在 datapackage.h 和 datapackage.cpp 中实现了数据包类的定义,在该类中,定义了数据包长度、时间戳、数据包内容、数据包类型这些私有变量。在抓包之后，会获取这些信息，并调用对应方法填充到私有变量中。

在该类中，还存在相应数据包解析的方法，根据已有的数据包内容、数据包长度、数据包类型,调用相应的数据包参数获取函数,获取协议数据包对应字段,从而输出显示到 GUI。这些参数获取函数，是通过协议标准，对各协议相应字段的内容进行对应加工处理，从而获得对应参数。如 IP 协议类型，需要从数据包头部+14 的长度开始解析，因为前 14 字节是 Ethernet 报头。

3.4 抓包线程类

在 multithread.h 和 multithread.cpp 中实现了抓包线程。在该类中，定义了网卡设备文件指针、数据包内容指针、时间信息、完成标志位等，并重载了 run 函数，定义了各类协议的解析函数。

在这些解析函数中，通过对数据报头的协议类型字段、端口等信息，识别其是 ICMP、TCP、ARP、UDP 等，并返回其协议代码。

在 run 函数中，主要在抓包的过程中打上时间戳、序列号、数据包类型、数据包长度等信息，方便在之后使用。

3.5 主窗口类

在 mainwindow.h 和 mainwindow.cpp 中，主要定义了 GUI 界面指针、网卡设备指针、数据包队列、已接收数据包数量、开始/暂停标志位等，并定义和实现了各个组件对应的功能和数值传递。

程序启动时会设置 GUI 的相应显示设置，捕获机器所有的网卡信息并添加到 comboBox 控件中，设置各变量初值，等待按下按钮改变标志位时启动抓包。

在选择网卡的 comboBox 控件中，下拉可以看到抓取的所有网卡，选取其中一个后会在页面左下角展示选取的网卡信息。

在开始/暂停抓包按钮，设置相应动作，

在清除按钮，即清空所有指针、标志位、变量，并将 GUI 页面显示信息清空。

在数据包信息显示的 tableWidget 控件中，根据不同的协议展示不同的默认设置颜色，并根据数据包类型进行对应解析和展示。不同的协议有不同的 info 信息，需要对应进行设置。

在协议树信息展示的 treeWidget 控件中，根据数据包类型进行对应解析和展示。调用之前在数据包类中定义的各种获取字段信息的方法，对数据包进行逐层解析和去头，比如最外层的 Ethernet 协议报头，之后是 ARP 报头等。

在控制上下查看的按钮控件中，如果按上，则在 treeWidget 中显示上一行；按下，则在 treeWidget 中显示下一行；按至顶则显示第一行；按至底则显示最后一个数据报。

在过滤规则的输入控件中,输入框会根据其内的内容改变颜色,只有为 UDP、TCP、DNS、ARP、ICMP、SSL、TLS 时会为绿色（只实现了这几个规则的过滤），其他的输入会显示红色。当抓包停止时按下回车键，会根据输入框中的内容进行过滤，在 tableWidget 控件中只显示指定协议的数据包，如果为空则显示所有。

3.6 其他文件

在 main.cpp 中，这是整个项目的入口函数，负责调用 mainwindow。

在 readonlydelegate.h 中，定义了一个 ReadOnlyDelegate 类，用于实现 GUI 中数据包信息所在的 tableWidget 控件信息只读、不被修改。

4. 实验小结

在整个实验的过程中，有以下收获：

（1）基本掌握了 QT 编写带 GUI 程序的方法，了解了很多控件的使用方法；

（2）熟悉了 npcap 的使用，对其接口函数有所掌握；

（3）对基本的以太网、IP、TCP、UDP、ARP、ICMP 等协议格式有了更深的认识，在整个解析协议报文的过程中，对其有了更深的认识；

（4）在初期确定方案的过程中，对 Wireshark 的架构和二次开发方法有了更深的了解。

5. 杂谈

参考资料：

QT 自制 wireshark (已完结)_哔哩哔哩_bilibili

likey99/mysniffer: 基于 npcap 的简单可视化网络嗅探器 (github.com)

djh-sudo/Network-capture: 网络抓包 (github.com)

实验留的时间大概只有三周，十一假期之前没留啥作业，放假回来各门课程依次留各种作业，课程多、时间紧、任务重，考虑到本身的编程能力确实没有那么出色，基本确定了是想要找个项目参考着来做的。

本来想着把 Wireshark 扒皮简化做一下，但后来发现 windows 版本的它是套了一层皮，QT 做的 GUI 仅用于前后端交互。之后就想着自己做，在 github 上搜了搜，python 就是调库做界面，作为备选项；MFC 本科用过，但这么古老的东西做一遍了解了解也就得了，不想再尝试；看到 C 语言的项目，有个用 QT 做的，后来就想着用 QT 了。按着学长的文档初步做了页面，后来在搜资料的时候在 B 站看到了那个教学视频，觉得不错，便跟着一步一步编程基本做完了。

整体的感觉，抓包占了很小一块，毕竟调库就完了。更多的时间用在了协议解析上，对着各个协议把该填的都填上，解析一层就去掉一层报文头部，最艰难的还是要有握手的，几部分颇为复杂。另外，不得不说，qDebug 也还挺好用的，可以输出到应用栏。（这么一说，倒是忘了试试用 printf 和 cout 会输出到哪里了）总之算是做完了吧，后续有些部分，保存文件、导出各种格式文件、更多协议支持、更多过滤规则等等，似乎还能补充不少。