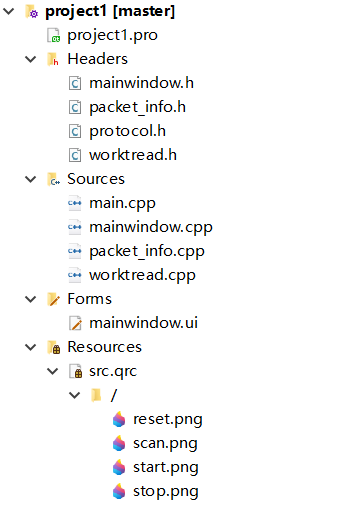
# Ex1A\_Report

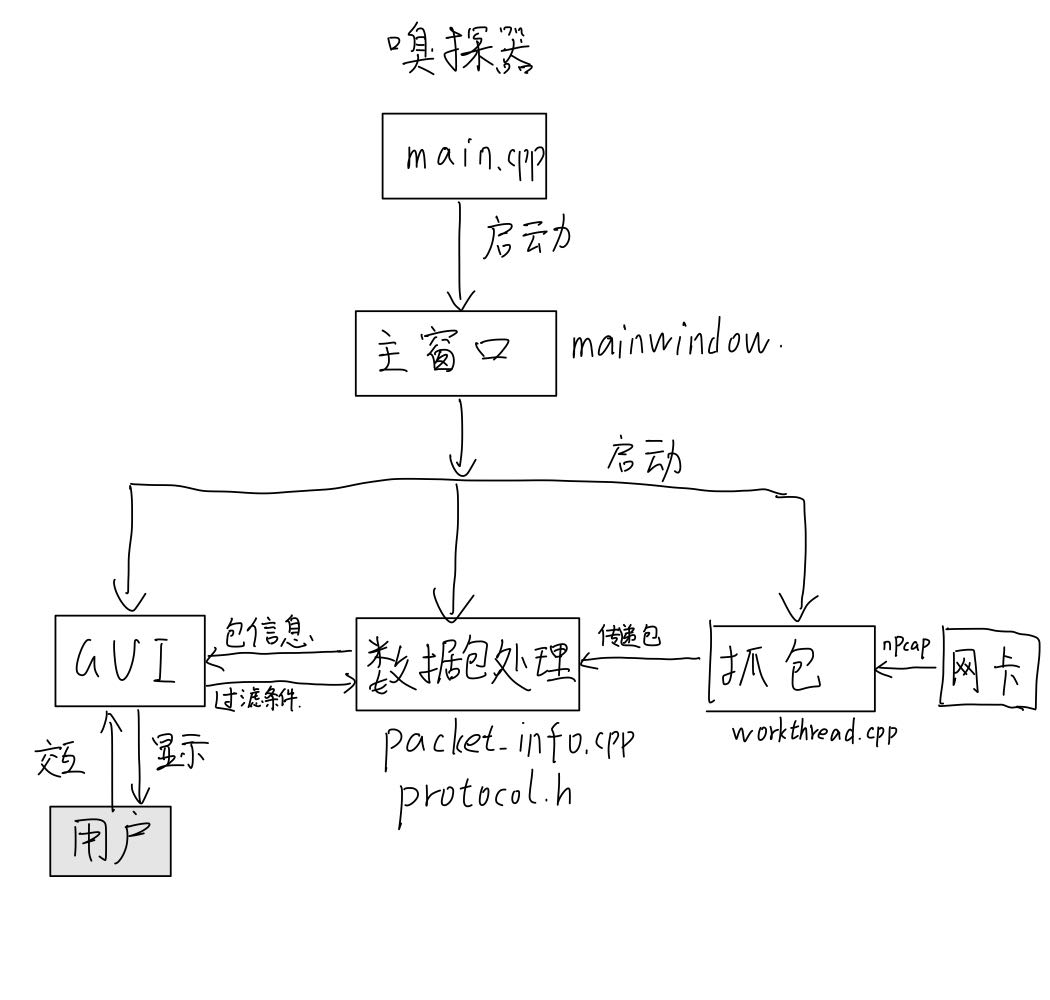
1. 简介

嗅探器的网络抓包功能基于Npcap\_1.60，开发中使用了Npcap\_SDK\_1.12提供的API，该项目地址[Npcap: Windows Packet Capture Library & Driver](https://npcap.com/)。图形界面的开发基于QT6.2。源代码的Github地址：[likey99/mysniffer: 基于npcap的简单可视化网络嗅探器 (github.com)](https://github.com/likey99/mysniffer)。

代码结构如下：

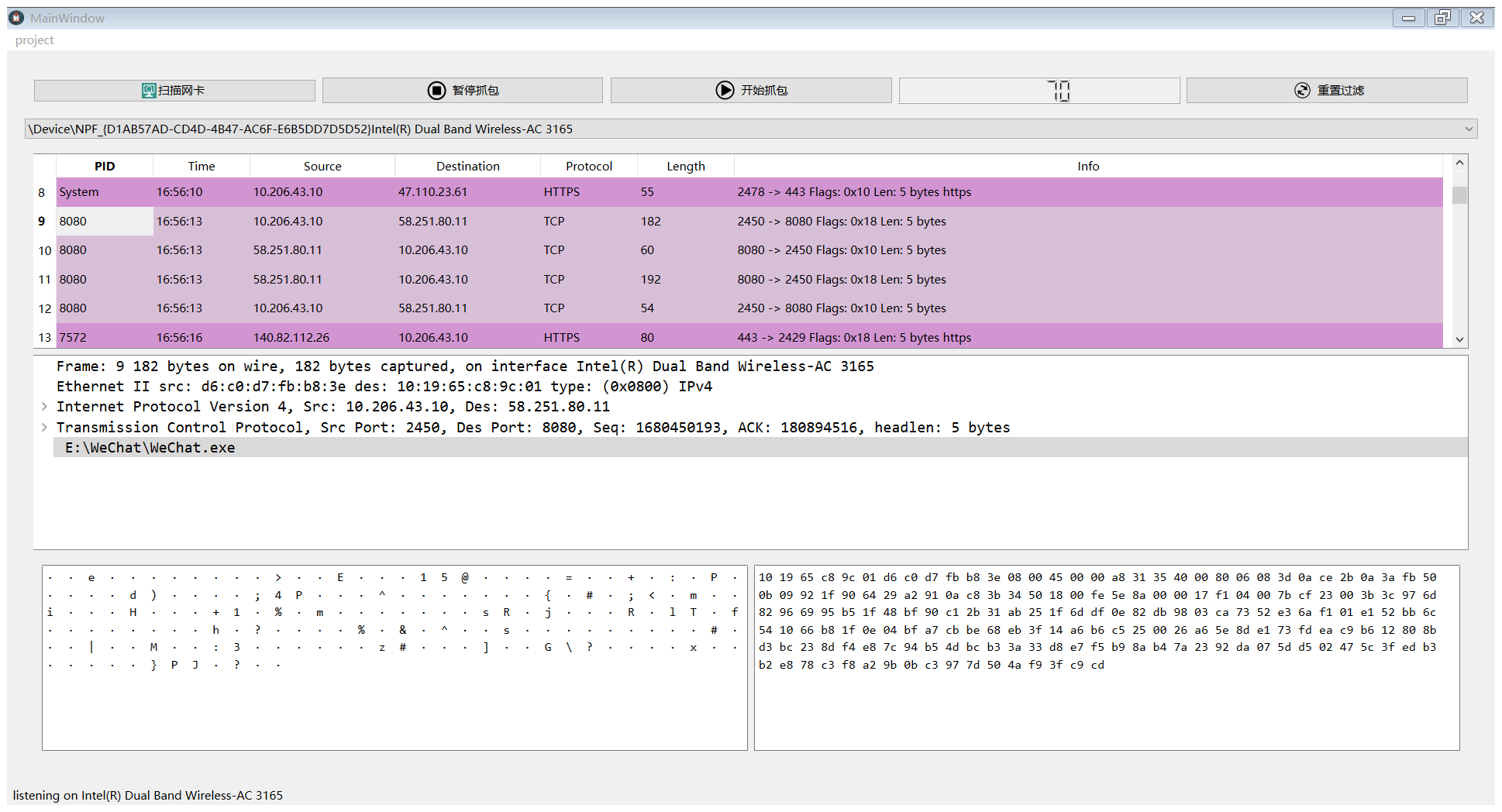


系统架构如下：



其中mainwindow文件负责图形界面的显示，与用户进行交互，进行数据包的解析和过滤；workthread文件为抓包程序运行的线程，负责从设备抓取数据包并传递给主线程；protocol文件定义了Ethernet、IP、TCP等网络协议的包格式头，用于数据包的解析；packet\_info文件定义了数据包类，包括数据包的基本属性和提取信息的方法。

程序运行界面如下：



使用流程如下：首先点击“扫描网卡”，然后选择合适的网卡，点击“开始抓包“，该网卡上的数据包就会显示在最上方区域，其中包含了一些基本信息：该数据包所属进程PID、时间戳、源地址、目的地址、协议类型、长度、端口号和一些标志位信息。

任意点击一行，即可在中间区域看到该数据包的协议解析结果，包括物理层、网络层、数据链路层、传输层的协议头解析结果，最后一行给出该数据包所属进程的路径和名称。最下方两个区域会分别显示数据包的ASSIC码值和16进制值，其中ASSIC值中不在打印字符区域内的以” · ”代替。

双击在最上方区域内的某一个格子，就可以以该格子内的值对数据包进行过滤，只显示符合过滤条件的数据包，其中PID标识为System的不能以PID为过滤条件，长度、时间戳也不能作为过滤条件，最后一列以端口号作为过滤条件，包括来往的数据包，没有端口号的无效。

点击“重置过滤“会清空过滤条件，显示所有数据包。

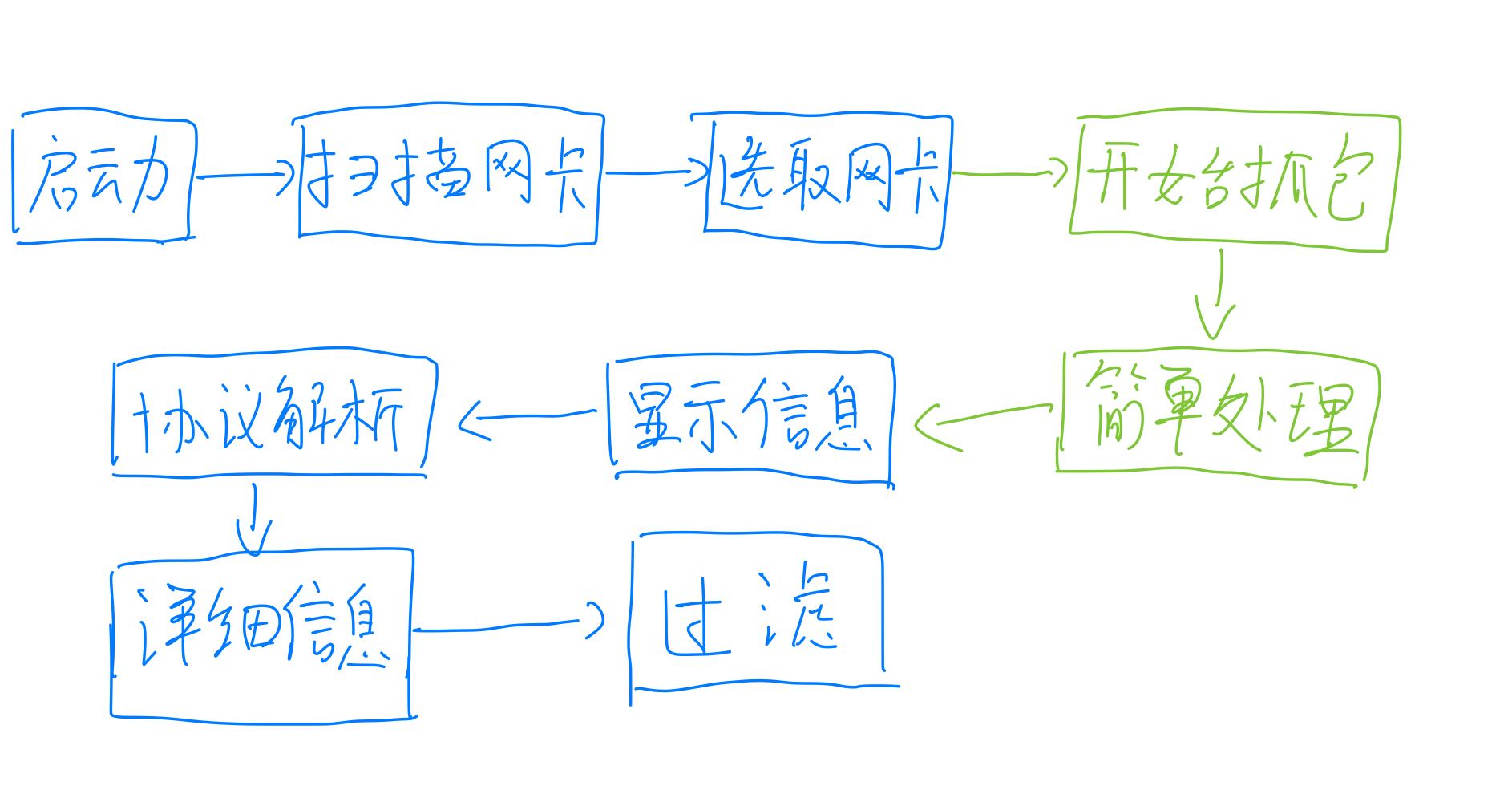
点击“暂停抓包“会停止抓包，不会清空数据包。之后点击”开始抓包“可以再次启动。

在非暂停状态时点击开始抓包会清空数据包，并重新开始抓包。

右上角的数字表示已经抓到的数据包个数。

1. 代码解释

按照如下流程对程序执行代码进行介绍



1. 启动

从main函数启动mainwindow，在mainwindow构造函数中对图形界面的tablewidget、textEdit 、treeWidget进行基本设置，包括列标签、列宽、字体、编辑选项。同时新建一个线程对象workthread，用于之后的抓包。

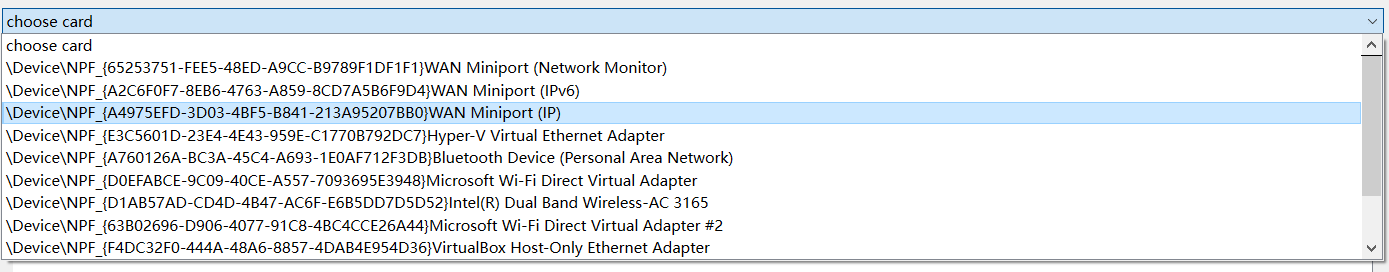


1. 扫描网卡

在用户点击“扫描网卡“按钮后，会调用**on\_pushButton\_clicked**()函数，调用npacp库中的pcap\_findalldevs（）对电脑上的网卡设备进行扫描，并显示在comboBox中。



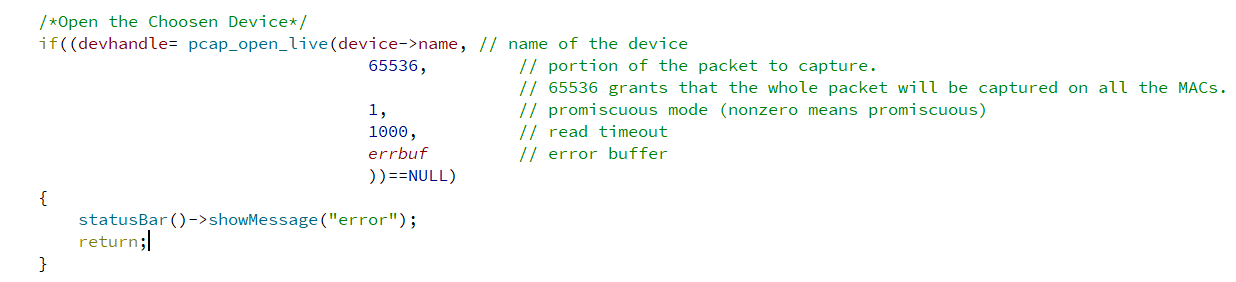
1. 选取网卡开始抓包



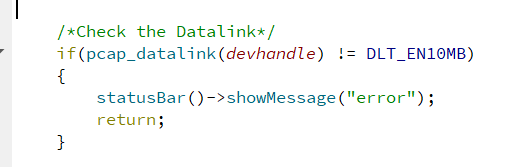
用户可以在comboBox中选取想监听的网卡，然后点击“开始抓包“按钮，会调用MainWindow::**on\_pushButton\_3\_clicked**()函数，该函数首先会判断程序的工作状态，如果在工作中，会刷新数据包列表，重新开始抓包，这个设置是为了方便在不同网卡间进行转换。



然后会调用Npacp库的pcap\_open\_live函数获取comboBox中选择的设备句柄，如果获取失败会在状态栏显示状态信息。

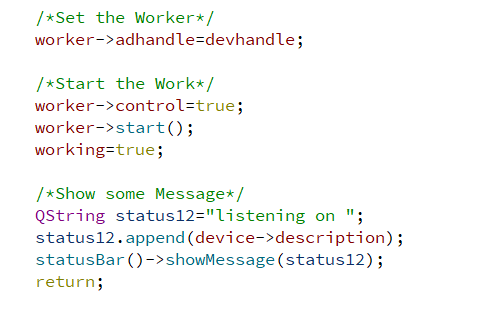


成功获取设备句柄后，首先检查当前设备的数据链路层是否工作在以太网下，程序目前只处理以太网下的网络数据包。



实际上在IEEE802.11环境下也能工作，这是因为网卡会自动把无线数据包进行处理，封装成伪以太数据包，此时原有的用于管理和控制的802.11包不会被捕获，如果想要捕获这些包，必须把网卡设置在监视模式下，这种模式下会显示所有接收到的包，包括其他主机的无线数据包。详情见[WLAN · Wiki · Wireshark Foundation / wireshark · GitLab](https://gitlab.com/wireshark/wireshark/-/wikis/CaptureSetup/WLAN#Windows)

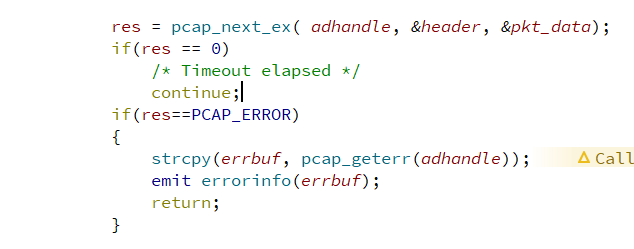
准备就绪后，会启动工作线程workthread，并显示工作状态。



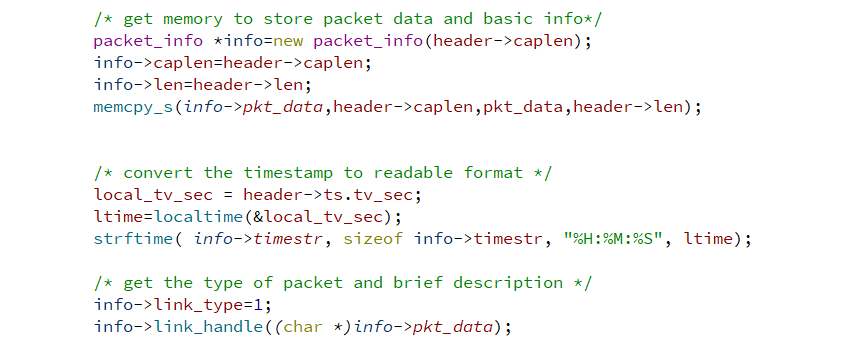
1. 简单处理

在workthread中对数据包进行抓取和简单处理。

首先调用npacp库中的pcap\_next\_ex啊函数获取设备上的一个数据包。



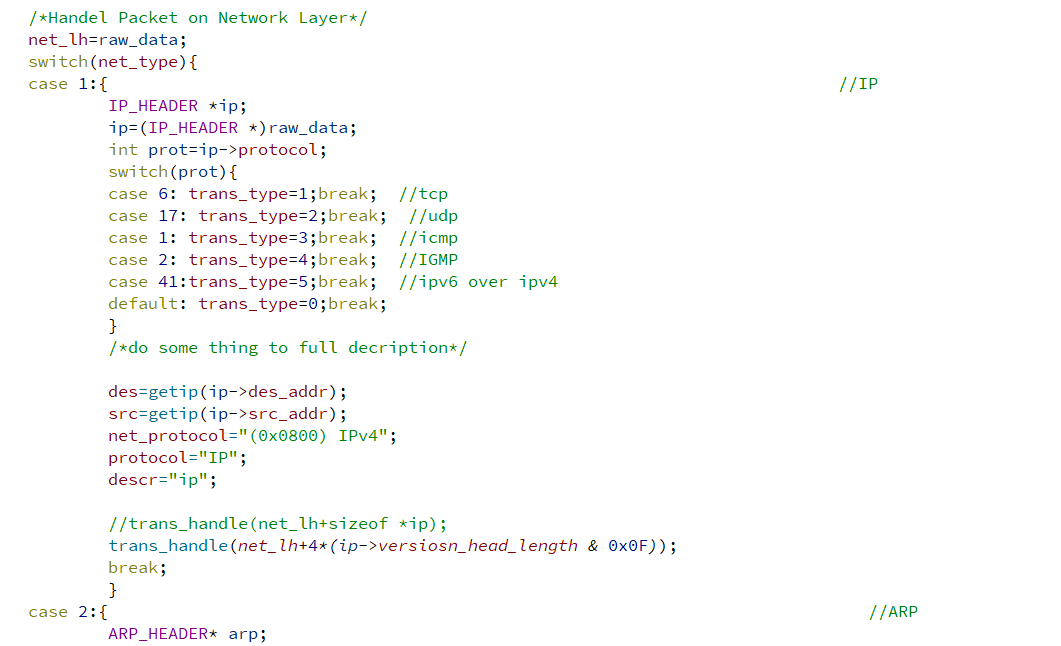
然后新建一个packet\_info对象，用来存储数据包的原始数据和一些协议解析信息，接着把数据包的原始信息拷贝进去，然后进行一些简单的处理。



调用link\_handle函数，对数据包进行简单处理，包括提取出以太网头，并将信息传递给网络层的**net\_handle**函数进行处理，同理依次调用传输层、应用层处理函数**trans\_handle、app\_handle**进行处理。



在链路层可以判断出网络层的协议类型，将这个信息保存下来，同时获取源地址和目的地址，并调用网络层处理函数。



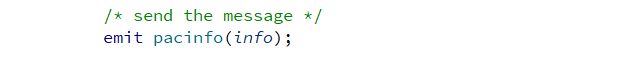
这里网络层能处理IP、ARP、IPv6三种协议，处理流程都是提取出网络协议头，获取源地址、目的地址、下一层协议，并调用传输层处理函数。



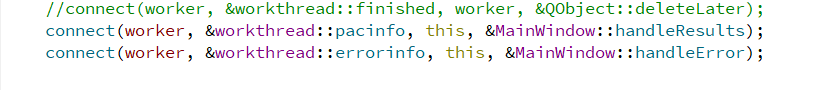
这里与之前类似，不过多赘述。所有的信息都保存在packet\_info中。可以识别的协议包括：Ethernet、IP、ARP、IPv6、TCP、UDP、ICMP、HTTP、HTTPS、IGMP、NetBIOS、IPv6 over IPv4、DNS。

1. 显示信息

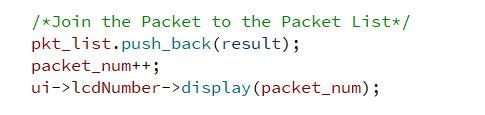
简单处理完一个数据包的各层信息后，工作线程就通过信号槽机制把packet\_info对象指针传递给主线程，主线程将进行信息的显示。



该信号与主线程的handleResults函数相连接，会自动调用该函数对数据包进行显示。



handleResults函数首先将新捕获的数据包添加到数据包列表中。



接着还需要将数据包与进程PID对应起来，调用refresh\_applist()函数，获取最新的进程列表，并根据端口和IP与数据包进行匹配，匹配成功就把PID填入packet\_info中。



为了获取当前的进程列表，首先需要调用**AdjustPrivileges**()函数提高当前进程的权限，具体原理参考[AdjustTokenPrivileges(进程权限修改) - 94cool - 博客园 (cnblogs.com)](https://www.cnblogs.com/94cool/archive/2012/10/30/2745785.html)。



为了获取进程与端口的关系，首先获取当前的TCP连接列表，列表中包含当前建立TCP连接的进程PID，然后根据PID可以获得对应进程的本地IP地址和端口，以及进程名，将这些信息存储在进程列表。这里由于权限不够，不能获取系统进程的信息，一律以“System“填充PID字段，并且不能作为过滤条件。



将数据包与PID绑定后，就可以将数据包信息显示在TableWidget中，为了增加区分度，为每种协议分配了一个颜色。

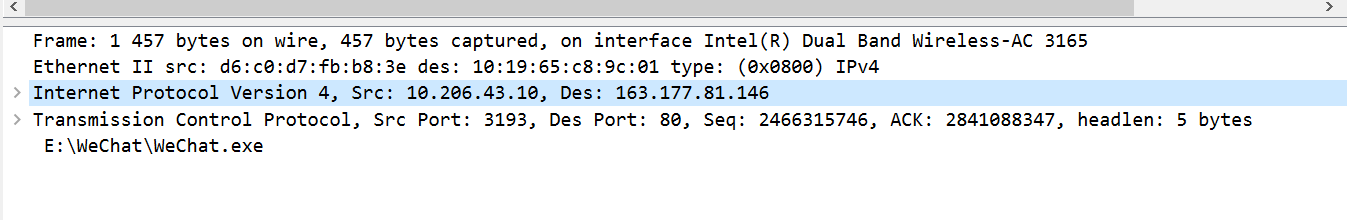


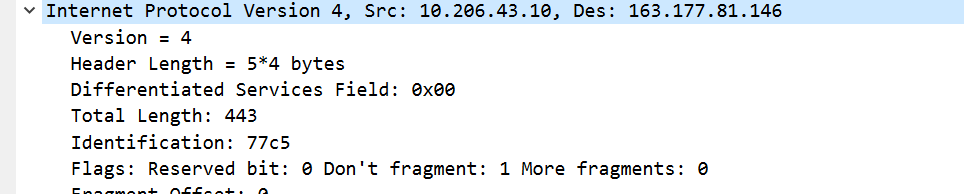
然后依次把一个数据包的"PID","Time","Source","Destination","Protocol","Length","Info"信息填入TableWidget中。



1. 协议解析

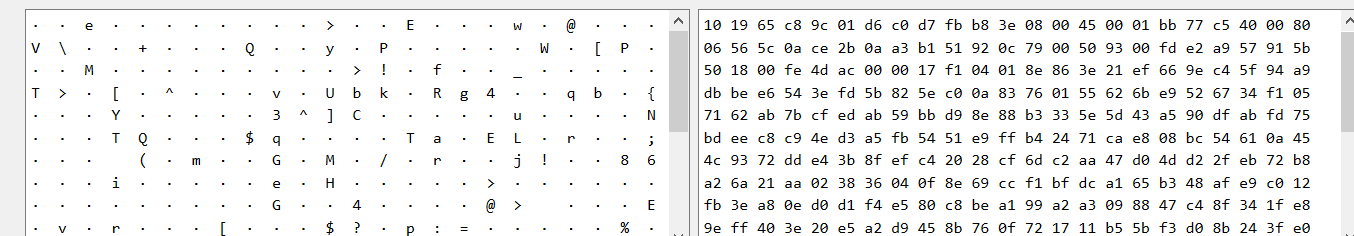
如果想查看一个数据包的详细信息，可以在TableWidget中点击对应的行，会调用**on\_tableWidget\_cellClicked**函数，就会在TreeWidget中显示各层详细信息。



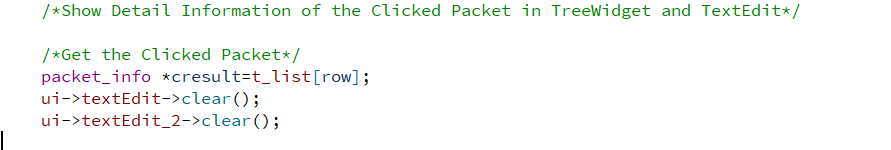


各项数据与wireshark类似。

在TextEdit中显示ASSIC和16进制的原始数据包。不在打印范围的ASSIC值用“·“代替。



实现过程：根据点击的行数索引到对应的数据包；



将原始数据转化为ASSIC和16进制并显示；



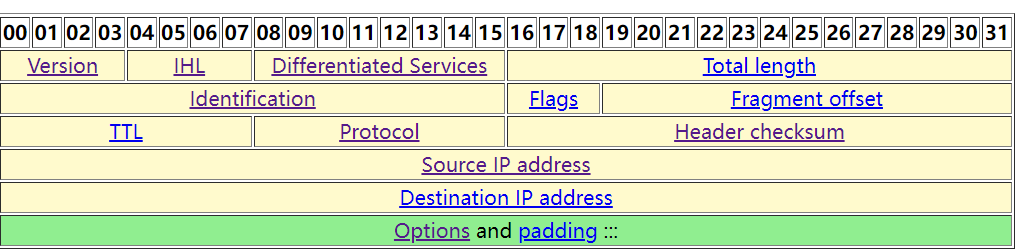
通过packet\_info的link\_info()、net\_info()、trans\_info()、app\_info()获取数据包各层的详细信息，并输出在treeWidget中。



1. 详细信息

link\_info()、net\_info()、trans\_info()、app\_info()这几个函数根据对应数据包的对应层网络协议，解析出每个字段的含义并以字符串形式返回。

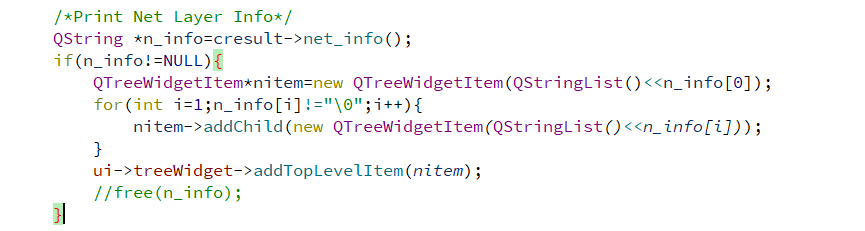
以IP协议为例，根据IPv4协议头进行解析：

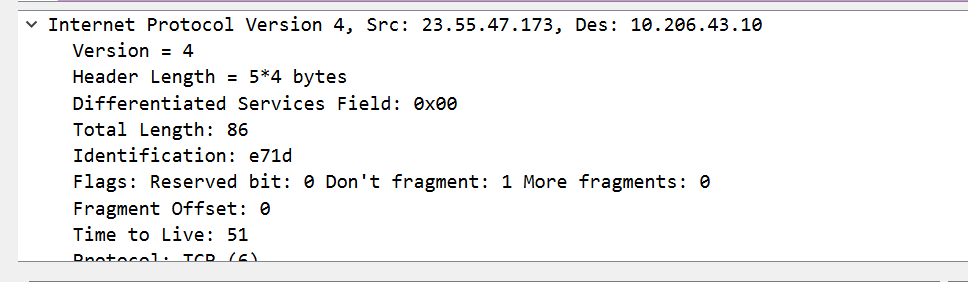


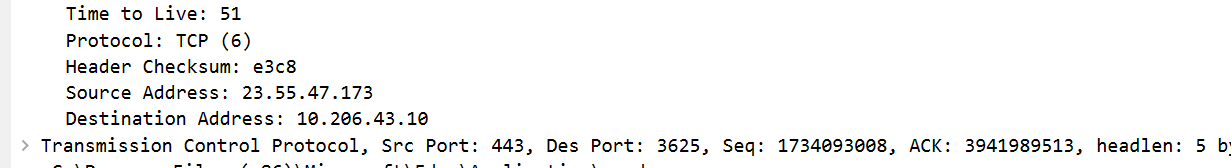


把每个字段的信息解析出来，以字符串形式返回，其他协议同理。可以解析的协议包括：Ethernet、IP、ARP、IPv6、TCP、UDP、ICMP等。

得到这些信息后，在treeWidget中进行显示。

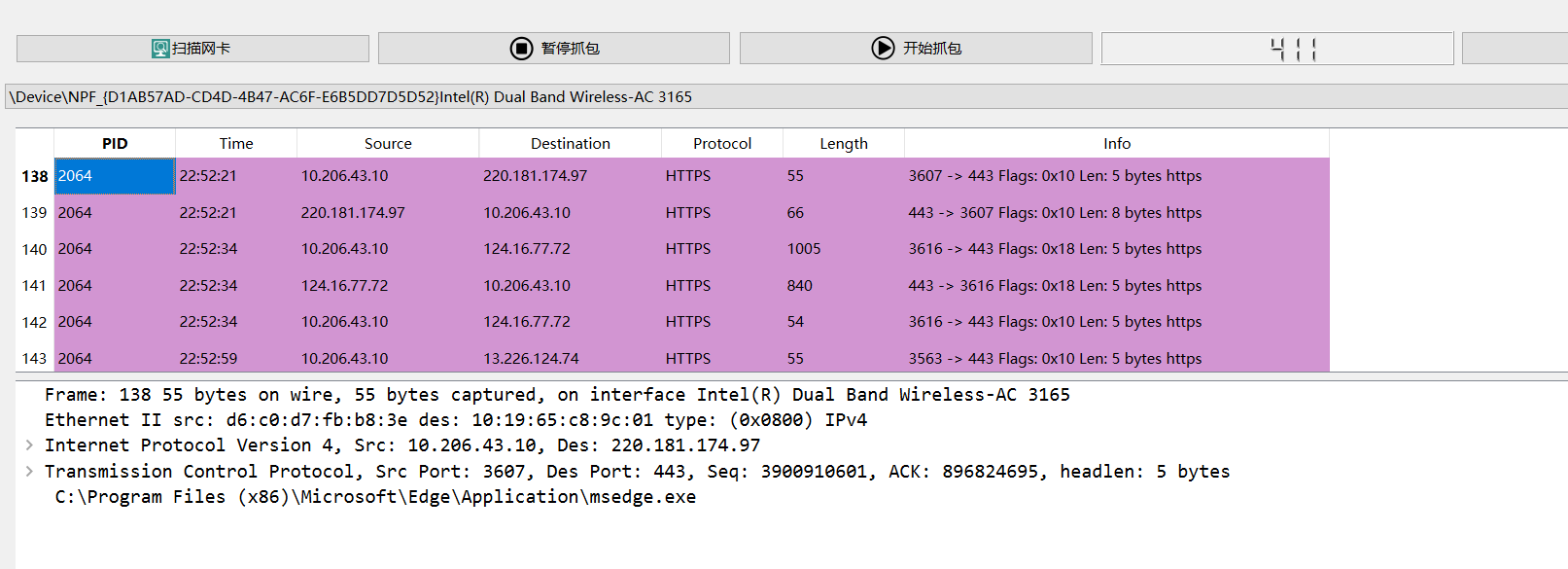






1. 过滤

双击tableWidget中的任意格子，即可将其值作为过滤条件对数据包进行过滤。以PID为例。下图双击了PID：2064，为Edge浏览器的PID，可以看到，从已捕获的400多个包中把PID为2064的Edge进程的数据包筛选了出来，其中PID标识为System的不能以PID为过滤条件，长度、时间戳也不能作为过滤条件，最后一列以端口号作为过滤条件，包括来往的数据包，没有端口号的无效。

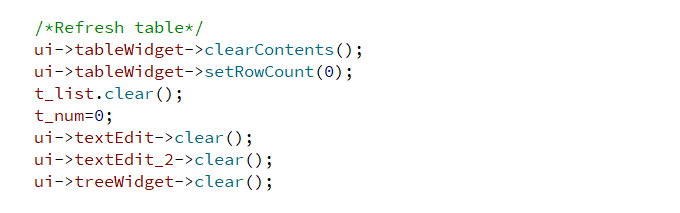


实现过程为首先调用**on\_tableWidget\_cellDoubleClicked**函数，它根据行列信息找出该格对应的值，并设置过滤类型和过滤值，接着调用**filter\_packet**()函数根据过滤条件刷新数据包显示列表。



有效的筛选类型为PID、源地址、目的地址、协议类型、端口号（传输层必须为TCP/UDP）。

**filter\_packet**()函数会清空当前的各项显示和显示数据包列表。



然后遍历所有的捕获数据包，跳过不符合过滤条件的



符合过滤条件的就被显示在各项Widget中，程序与之前类似。



最后还有一个“重置过滤“按钮，它会调用**on\_pushButton\_4\_clicked**()函数，将过滤条件清零，并调用**filter\_packet**()函数刷新图像显示。

