**Sniffer 项目文档**

袁鑫 肖鹏宇

**一.概述**

本项目实现了一个网络嗅探器，其功能有：对用户选定的本机网卡进行抓包和简单解析、包过滤、包查找、IP包重组、保存文件与文件读取、文件包重组和文件包过滤。本项目实现了一个用户友好的GUI。

本项目基于Python3环境，使用scapy库进行抓包、解析包、过滤包，使用PyQt5实现GUI。

**1.项目环境配置**

|  |  |
| --- | --- |
| 环境/库 | 版本及说明 |
| **Python** | **3.6.X (3.4.X以上)** |
| **scapy** | **2.4.3**  **(支持Python3.4以上)** |
| **PyQt5** | **5.13.0** |
|  |  |

**2.运行环境说明**

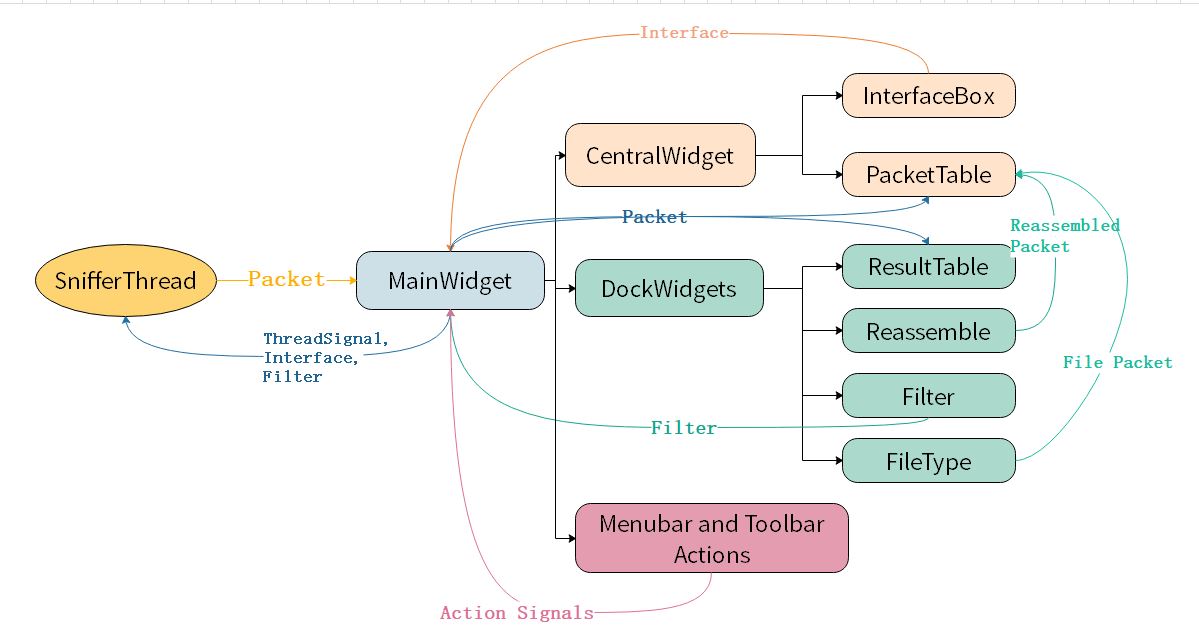
本项目适配Windows环境，建议使用Windows 10 系统运行；在Linux环境下，可能由于scapy的get\_windows\_if\_list函数不适配而导致无法运行。

**3.程序文件列表**

|  |  |  |
| --- | --- | --- |
| 文件夹 | 文件、子文件夹 | 说明 |
| **/src** |  | **文件源码** |
|  | **/icon** | **UI图标** |
|  | **main** | **主程序运行接口** |
|  | **MainWindow.py** | **主窗口及程序总控函数** |
|  | **SnifferThread.py** | **程序唯一线程，嗅探线程** |
|  | **CentralWidget.py** | **中间窗口（不可动）** |
|  | **InterfaceBox.py** | **网卡接口相关控件（中间窗口组成部分）** |
|  | **PktTable.py** | **抓包显示窗口（中间窗口组成部分）** |
|  | **ResultTable.py** | **包解析显示窗口（dock窗口）** |
|  | **Filter.py** | **添加BPF过滤规则窗口（dock窗口）** |
|  | **ReassembleWidget.py** | **包重组窗口（dock窗口）** |
|  | **ChooseTypeFile.py** | **文件包过滤查找窗口（dock窗口）** |
| **/testfiles** |  | **测试用文件** |
|  |  |  |

**二.程序结构**

**1.程序框图**



**2.结构说明**

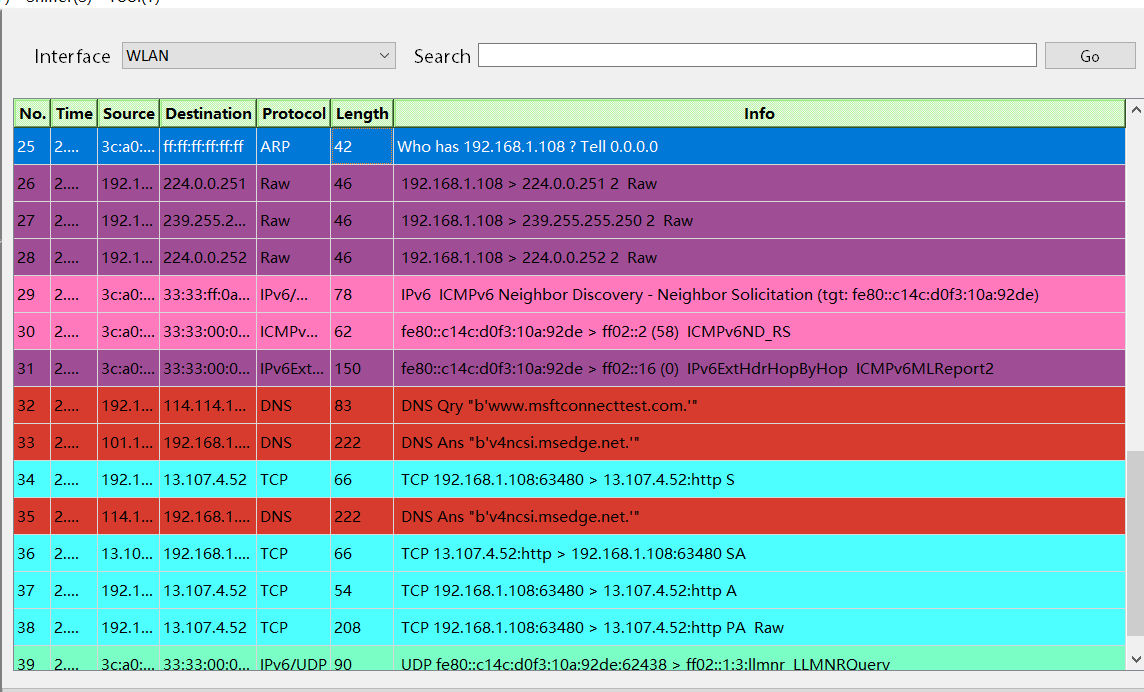
**(1)启动:**运行main.py，启动QApplication，弹出Splash加载界面，稍等偏刻后进入主界面MainWindow；此时，各窗口初始化；

**(2)MainWindow:**这是一个QMainWindow的继承类，是程序的主窗口；该类中实现以下事项：

设定了程序的总体布局；进行嗅探线程的创建和终止；各模块的信号、数据中转（使用pyqtSignal）；部分菜单栏功能函数（如pcap文件的存储和读取）。

MainWindow的默认布局是：从内到外依次为CentralWidget，DockWidgets，MenuBar和ToolBar。

**(3)CentralWidget:**中间窗口，不可移动的基准窗口，继承QWidget类。包含Interface选择、和抓包显示PktTable两个调用的子模块和包查找内置函数；为主窗口提供网卡变量Interf和切换网卡带来的暂停信号Signal\_SniffStop\_Interf。

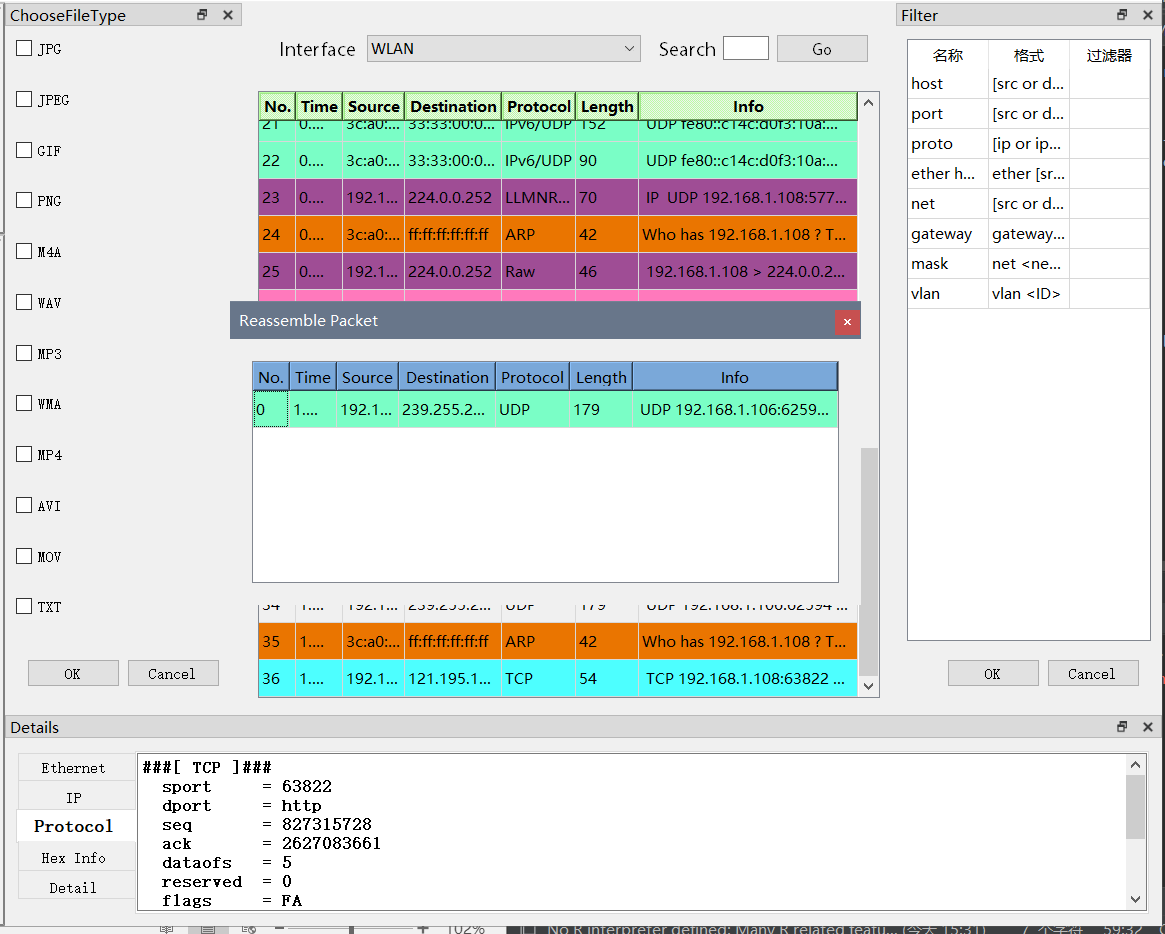


**(4)SnifferThread:**本项目中唯一的阻塞线程类，继承QThread，但重载了run函数，在run函数中使用scapy\_all.sniff函数进行嗅探，使用threading.event()事件进行线程阻塞。在SnifferThread.py中定义了若干与包解析相关的函数供调用。sniff函数的主要参数是网卡Interf、过滤规则Filter和停止信号；在主程序每次新建线程时，都会先发出停止信号，关闭可能存在的event，确保嗅探线程唯一运行。嗅探到的包，传递给主窗口，以字典形式存储，并传给PktTable等模块进行包解析和显示。

**(5)InterfaceBox:**选择网卡的小模块，通过scapy自带函数获得网卡名，并过滤不可用网卡（没有MAC地址的isatap隧道等）。通过pyqtSignal信号，实现没有选择网卡时不予启动，换网卡则停止嗅探并清空表格。

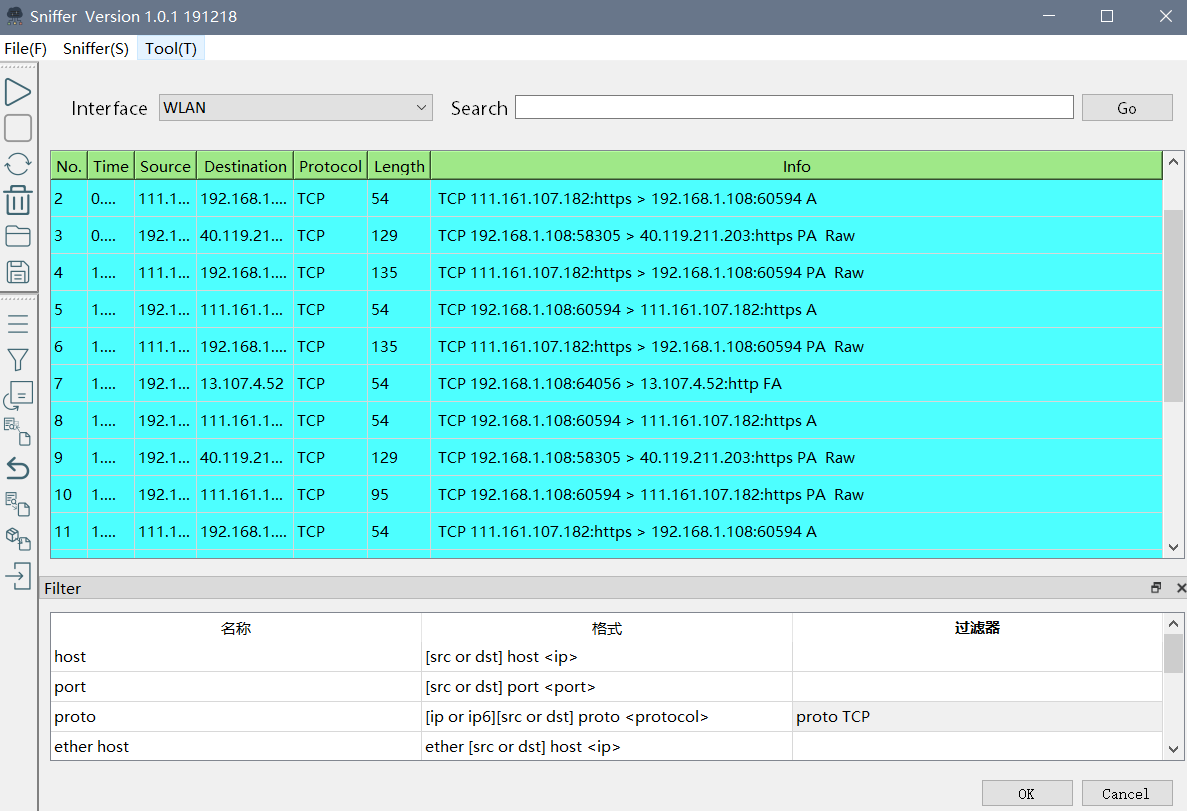
**(6)PktTable:**QWidget的继承类，内置一个QTableWidget实现表格式抓包展示；PktTable.py带有可供调用的包处理函数；利用QTableWidget自带的双击事件触发细节显示；

**(7)DockWidgets:**这是一系列位于DockArea的活动窗口，可以开关、拖动调节位置和大小、单独窗口显示，如下图有四个活动窗口，左右下各一，中间有一个单独显示的窗口。QDockWidget的特性使用户能够灵活方便地使用它们，并根据喜好、便利调节位置和大小。



下面是这几个活动窗口的简单介绍：

**(8)Filter:**为嗅探线程提供包过滤功能，按照scapy\_all.sniff函数需求，使用BPF语法。用户按规则填入后，会转成需要的格式作为过滤器；

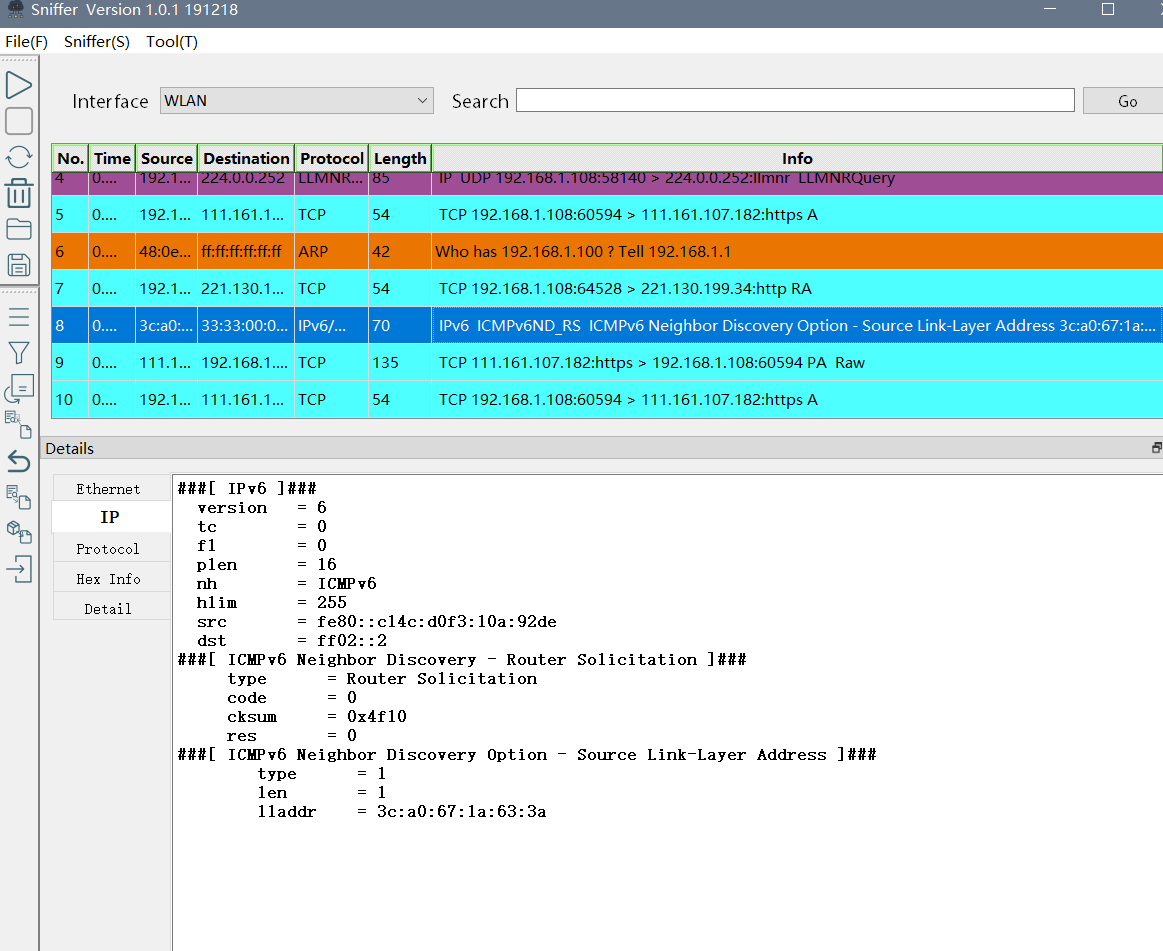


**（当填入协议过滤后，只监听TCP包）**

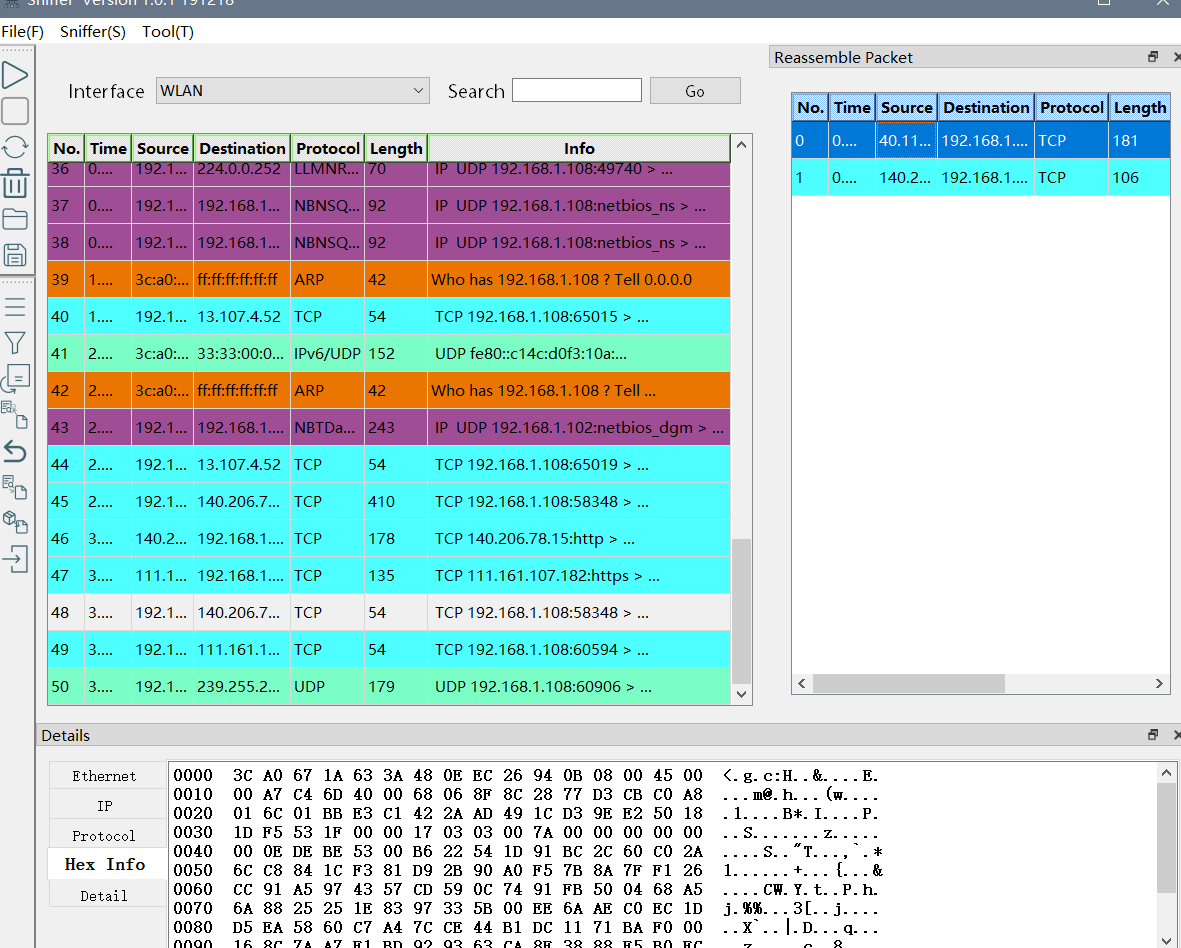
说明：本图是为方便介绍而制作的截图。filter窗口打开时，视为规则设置中，此时启动sniffer,过滤器是无效的。**要想过滤包，在填写完后，点击OK按钮，filter窗口会关闭，不打开filter窗口，启动sniffer，过滤规则生效；filter窗口打开状态下启动sniffer，无论之前是否设置规则，都视作不生效。**

**(9)ResultTable:**继承QWidget类，内置QTabWidget类。该窗口由PktTable表格行双击触发，该表格显示包的细节信息。

为实现标签橫置，对于该类内的QTabBar部件的固有方法paintEvent函数进行了重载。



**(10)****ReassembleWidget:**继承QWidget，调用PktTable，展示包重组功能。将选定的包重组后的包展示在这个区域，点击之，会在ResultTable中显示重组后的包的信息。



**(11)ChooseFileType:**实现文件的数据包按照文件类型过滤，勾选文件类型，根据特定的包前缀寻找出组成文件的数据包；

**(12)MenuBar&ToolBar:**一系列QAction添加到菜单和工具栏中，并添加图标，可以通过这些按钮直接使用某些功能。鼠标悬停在图标上显示说明。

**三、算法说明**

**1.几个较重要的变量说明**

在MainWindow.py中，有字典self.HistoryPacket和self.CurrentPacket，其中self.HistoryPacket是当前抓到的所有包的字典，self.CurrentPacket是当前显示在PktTable界面的所有包的字典，他们的键值对中key是包在PktTable中要插入到的行数，value是抓到的包。

self.HistoryPacket起到在对抓到的包进行操作时保存原有所有数据包的作用，self.CurrentPacket起到显示满足条件的数据包的作用。

在抓包的过程中，self.HistoryPacket与self.CurrentPacket是保持一致的，二者只有在停止抓包且对数据包做出操作时才有可能不一致，且做出除清空(Clear)和重新抓包(Restart)之外的操作时，self.HistoryPacket保持不变，self.CurrentPacket在self.HistoryPacket中选择合适的数据包并显示在PktTable中。当然，self.CurrentPacket的作用不止于此，会在下面的算法实现中加以介绍。

**2.各种功能的算法实现**

说明：算法实现部分侧重逻辑实现，对重要函数的功能会加以细节描述，对于对理解算法实现无太大帮助的函数、变量，仅描述其功能或直接略过。且由于本程序代码较多，所以只给出入口函数的代码，若想要了解更详细的代码，可以在源代码中从入口函数往下看，入口函数都在MainWindow.py的class MainWindow中。

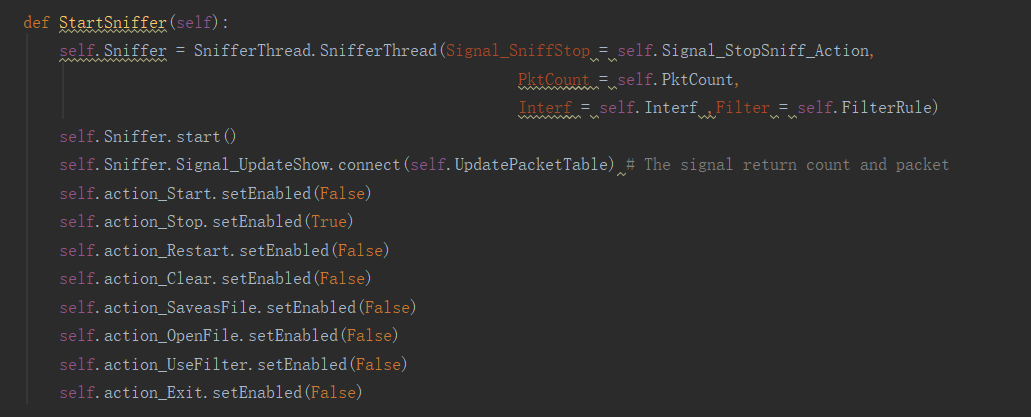
在代码可读性方面，本程序对变量的命名采取形象的命名原则，且绝大多数函数前都有对函数功能的说明，对理解代码逻辑有一定的帮助。

**（1）数据包的抓取、停住抓取、重新抓取、清除及显示**

a.数据包的抓取

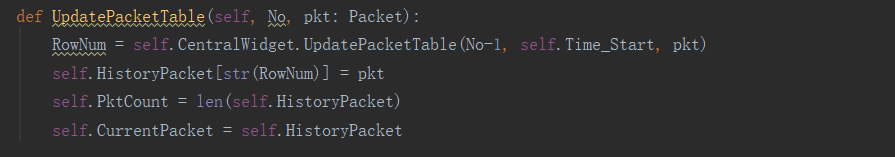
抓包功能的入口函数是Start()，界面中Start按钮关联了此函数。按下按钮即开启SnifferThread子线程抓取数据包。若开始时PktTable中没有任何数据，则先初始化，并开始计时，再抓取，否则直接进行抓取。





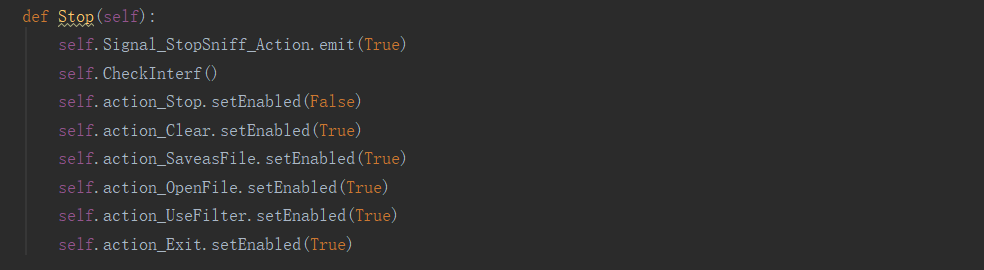
b.数据包的显示

数据报显示功能的入口函数是UpdatePacketTable()函数。显示数据包的界面是PktTable。在通过SnifferThread子线程抓到数据包后，UpdatePacketTable()函数完成两个功能，一是对self.HistoryPacket、self.CurrentPacket的更新，一是对PktTable 界面的更新，经过层层调用，通过BulidPktDict()提取要显示的信息，包括序号、时间、源地址、目的地址、控制协议、数据包长度以及其他信息，这些信息会通过InsertPkt()函数被显示到PktTable中去。



c.数据包的停止抓取

数据包停止抓取功能的入口函数是Stop()，通过关闭SnifferThread子线程来实现。



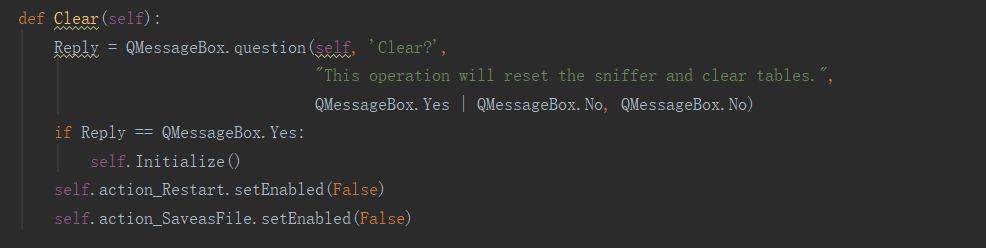
d.数据包的重新抓取

数据包重新抓取的入口函数是Restart()，此函数先调用Initialize()函数进行初始化，将PktTale、ResultTable全部清空，也将self.HistoryPacket与self.CurrentPacket清空，然后重新计时，并重新开启SnifferThread子线程进行抓包。



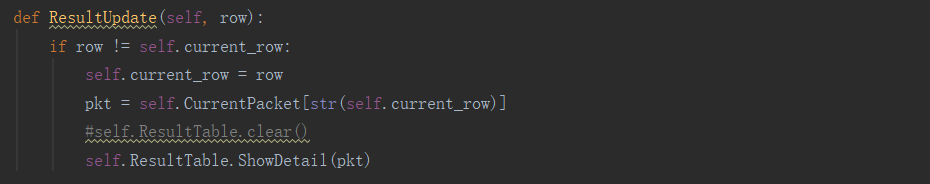
e.数据包的清除

数据包清除功能的入口函数是Clear()，通过调用Initialize()函数实现。



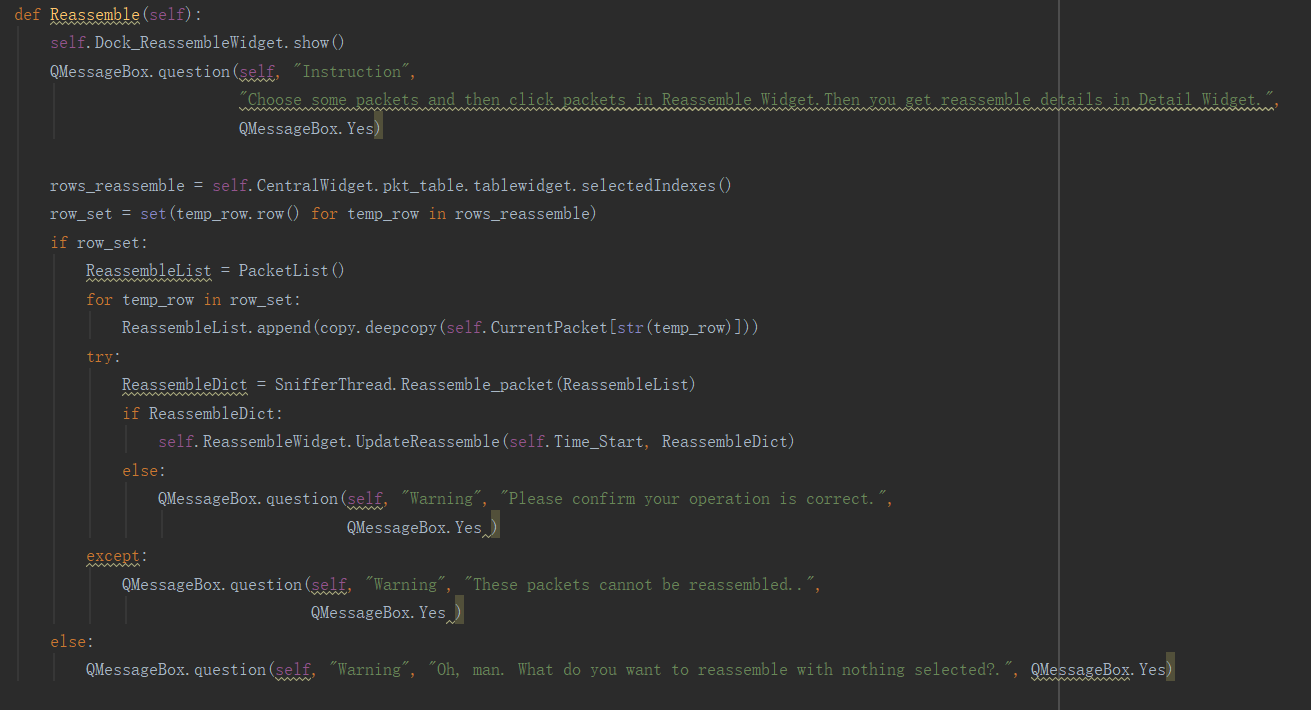
**（2）数据包的分层解析**

在CentralWidget中定义了一个双击事件，在主线程MainWindow中此事件与ResultUpdate()函数关联，当双击PktTable中某一行（已在PktTable中规定选中以行为单位）时，ResultUpdate()函数将会调用ResultTable中的ShowDetail()函数，并在self.CurrentPacket查询此行的包，将包作为参数传递给ResultTable中的ShowDetail()函数，ShowDetail()函数将会运用Scapy库解析此数据包的数据链路层信息、网络层信息、传输层信息，并将其显示出来。



**（3）数据包的IP分片重组**

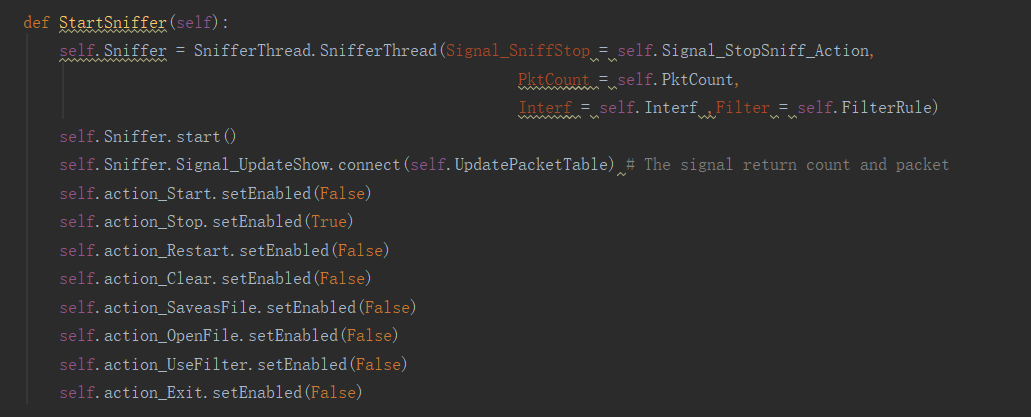
从同一数据包分片得来的数据包中，若是IPv4的报头，则id这一项是相同的，片偏移frag不同。实现分片重组的入口函数是MainWindow中的Reassemble()函数，此函数通过调用其他函数，将在PktTable中选中的数据包根据id进行分类，id相同的按照片偏移进行重组，并将重组结果在Reassemble Packet界面中进行显示，通过双击Reassemble Packet界面中重组后的数据包，也可以在Result Table中查看这些数据包的详细信息。



**（4）包过滤**

包过滤功能通过Filter界面以及SnifferThread子线程实现。Filter将用户输入的信息组装成满足BPF语法条件的包过滤信息，然后作为参数传递给SnifferThread子线程，再抓取数据包时就会根据包过滤条件进行抓取。

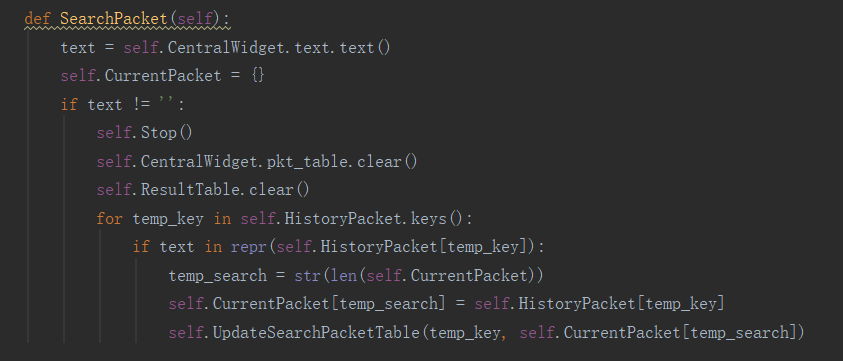


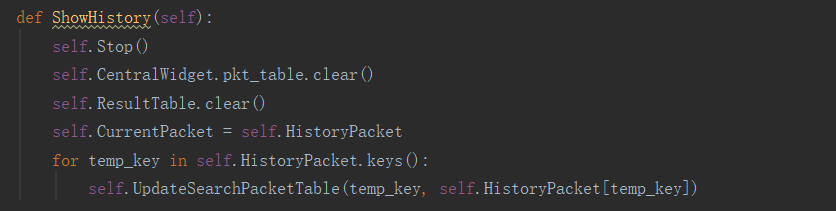


**（5）数据包查询与返回**

数据包查询的实现较为简单，此功能的入口函数是SearchPacket()。当用户在搜索框中输入查询条件并点击查询按钮时，程序将会遍历self.HistoryPacket，即遍历所有数据包，每一个数据包，通过repr函数将其转换成一个字符串，查看输入的信息输入的信息是否在此字符串中，若在，则将其加入sekf.CurrentPacket，并显示。

查询之后若想要显示原来包含所有数据包的PktTable，则要用到返回功能，此功能的入口函数是ShowHistory()，通过清空PktTable、ResultTable，并且将self.HistoryPacket中的数据包的相关信息重新插入到PktTable中实现。

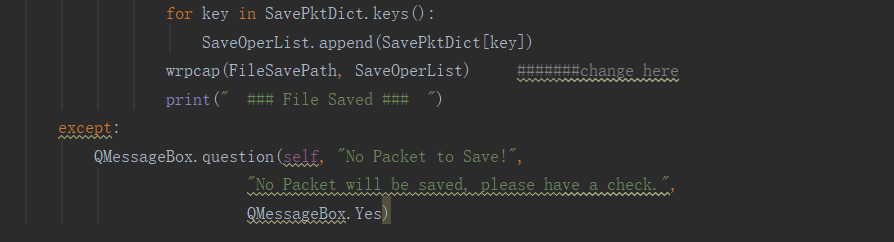
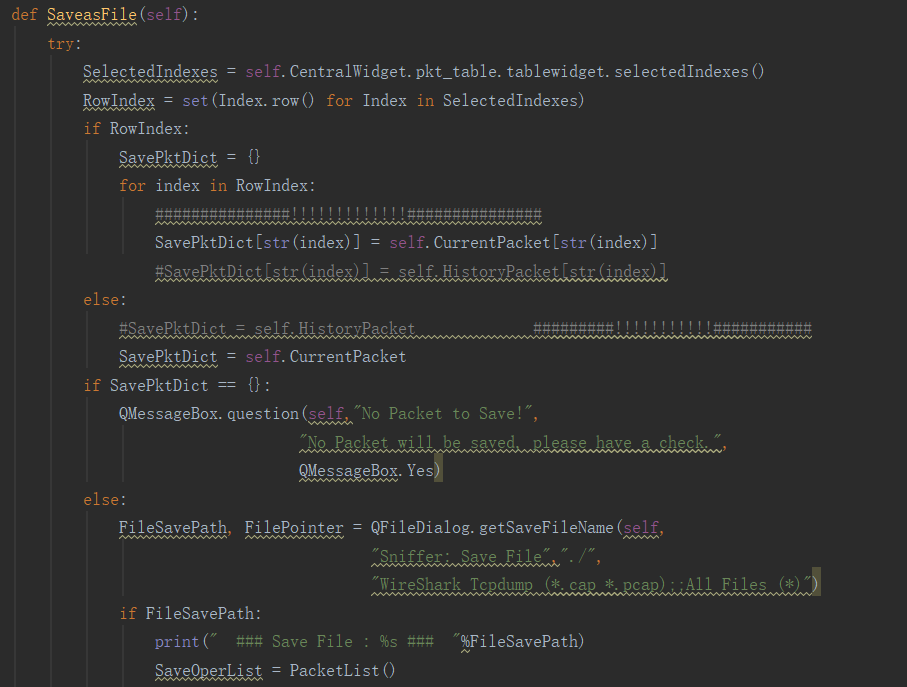


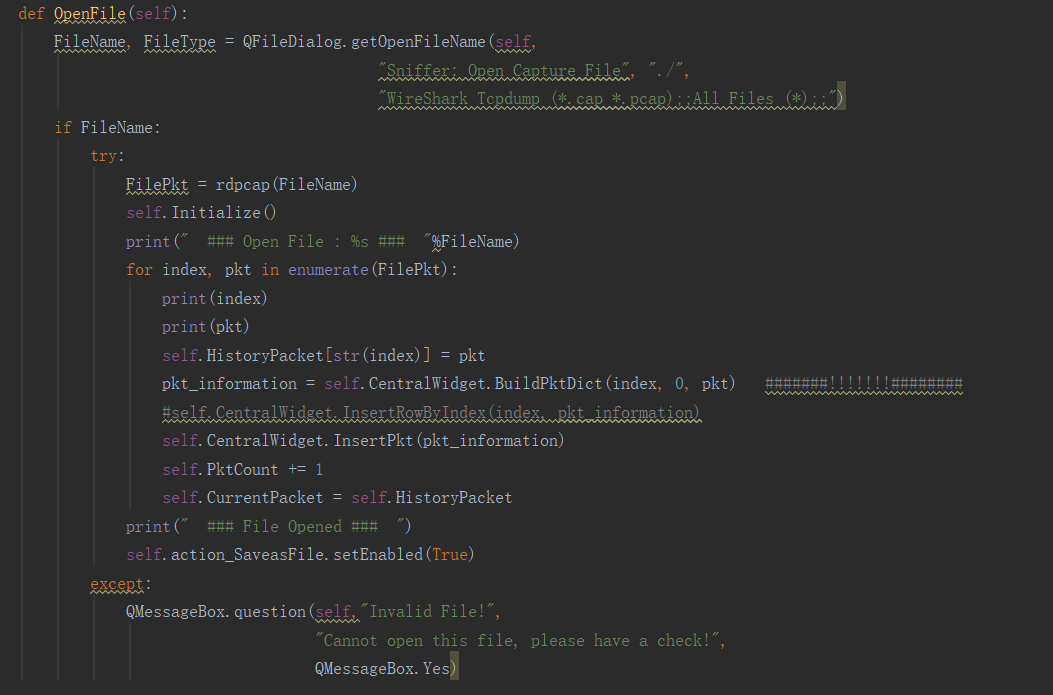


**（6）数据包文件的保存与打开**

保存数据包文件功能的入口函数是SaveasFile()，此函数首先在self.CurrentPacket中查找所有被选中的数据包，然后通过Scapy的PacketList()构造要存储的数据包的列表，再通过wrpcap()函数将它们写入文件，文件拓展名应为cap或pcap。

打开数据包文件功能的入口函数是Openfile()，此函数通过Scapy的rdpcap()函数从cap或pacp文件中读出所有数据包，更新self.HistoryPacket、self.CurrentPacket，并将读取的数据包显示在PktTable中。





**（7）文件重组**

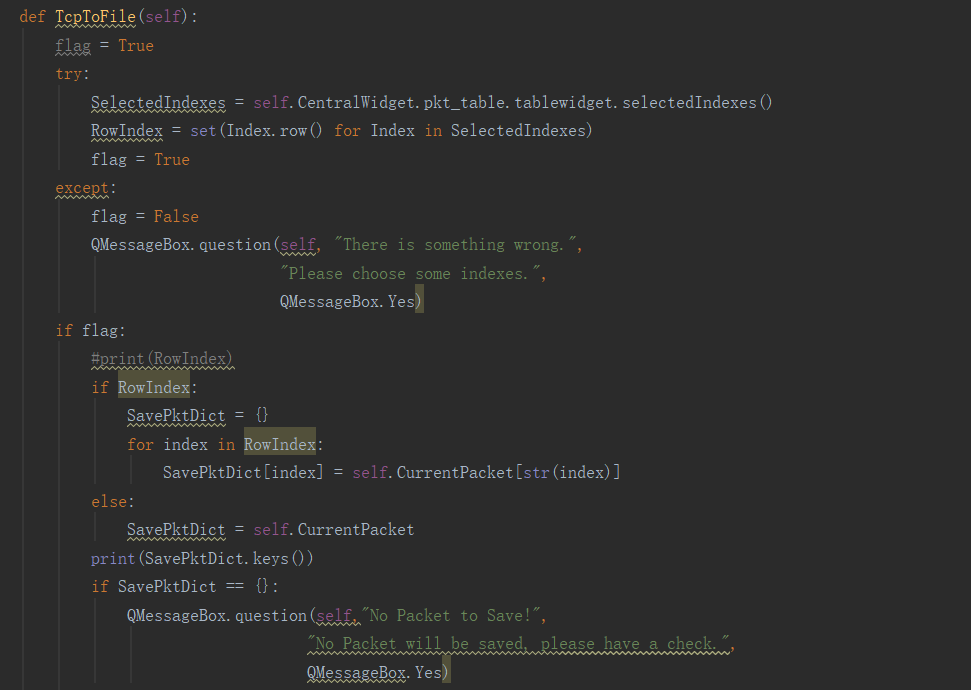
本项目采用两种方式实现文件重组功能，下面对这两种方式分别叙述。由于本部分入口函数过长，只给出部分入口函数代码。完整代码请参见源程序。

1. 对选中的所有数据包直接重组成文件

若是用户已经清楚想要把哪些数据包重组成为文件，可采用此功能。此功能的入口函数是TcpToFile()，此函数将选中的所有数据包的Raw部分组成一个bytes类型的变量，如果是文本文档，则可直接写入文件，如果是图片、音频或者视频文件，直接写入文件不会报错，但是经实际测试无法打开文件，所以本函数将会先使用base64对此bytes类型的变量进行编解码，然后再写入文件，在保存文件时，无论选择何种文件类型都不会报错，但是若选错文件类型可能会导致文件无法打开，例如将txt文档的数据写入MP4文件，自然无法打开，若是想知道此文件应是何种类型，可采用（8）中的文件类型过滤功能进行判断。

此功能对于用二进制方式传输数据的场景都可以使用。

此功能支持对乱序到达的数据包进行重组。



1. 通过数据包前的传输信息自动查找所有数据包并重组成文件

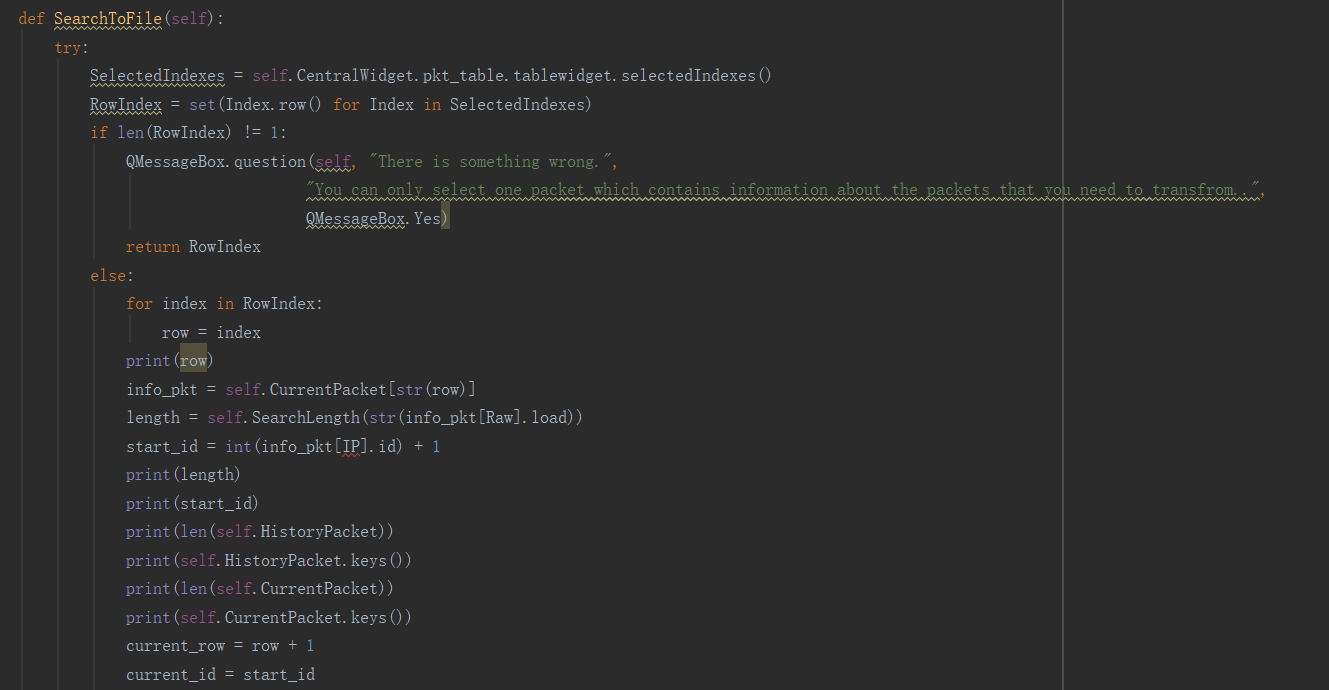
若用户不知道要将哪些数据包重组成文件，但是找到了传递文件参数的数据包，则可采用此功能。从功能的入口函数是SearchToFile()，此函数首先从传递参数的数据包中提取出文件长度length，以及此传递参数数据包IP报头中的id，然后对位于此数据包之后的所有数据包进行遍历，并且建立查询字典，字典的key是数据包IP报头的id，value是数据包在PktTable中的位置，即行数。此字典建立之后，由于我们已经知道了传递参数的数据包的id，记为start\_id，则第一个数据文件的id应该是start\_id+1，第二个数据文件的id应该是start\_id+2，以此类推，在查找过程中将查找到的数据包的Raw部分构建成一个bytes类型的变量，当此变量的长度小于length时，则继续查找，直到长度为length，停止查找。

若是在查询字典中找不到某一id的数据包文件位置，则认为发生了丢包现象，此时停止查找，向用户报告，并根据用户是否要坚持存储将已收集到的数据写入文件。

同上一种方式，若不知道此文件是何种类型，可采用（8）中的文件类型过滤功能进行判断。

此功能仅适用于用QQ传输文件，对于使用其他方式传输文件，本方式不保证能够成功。

此功能同样支持对乱序到达的数据包进行重组。



**（8）文件类型过滤**

文件类型过滤功能的入口函数是ChoosePktFile()函数。

首先，我们先介绍一些MainWindow.py中唯一的全局变量——字典symbol。此字典的key是文件类型，包括4中图片类型、4种音频类型和3种视频类型。由于未找到相关资料，所以我们找到了众多相关类型的文本文档、图片、音频和视频，自行测试得到了以二进制方式读取它们时它们的二进制前缀（或接近前缀的部分）的特点，也就是传输过程中第一个数据包的数据部分Raw的特点。

对于文本文档，如txt类型、doc或docx类型的文件，它们没有任何前缀，Raw部分就是文件内容，所以在构建symbol时未加入这三种类型的文件，但是我们仍然采用了一种方式去判断是否是文本文档，下面会加以叙述。对于其他类型的文件，如JPG、JPEG、GIF、PNG、M4A、WAV、MP3、WMA、MP4、AVI、MOV，我们都找到了它们的前，中特有的部分，如JPG的前缀中一定会有b'\xff\xd8\xff\xe0\x00\x10JFIF'，其他类型的文件则没有，这种前缀会为我们判断文件是何种类型提供一种可靠的参考。

我们首先遍历所有数据包，找到传递文件参数的数据包，从中提取文件长度以及（7）中提到的start\_id。然后就像（7）中第二种方式一样建立查询字典、综合数据，不过不同的是，在找到第一个数据包时，我们会进行类型判断。

判断文件类型时，我们之前会收集用户要过滤的文件类型，然后将这些文件类型的前缀用来判断。首先，由于文本文档无前缀，我们将其留到最后进行判断。如果第一个数据包的数据部分含有用户要过滤的某个文件类型（除文本文件类型）的前缀，则我们直接判断此文件的文件类型就是与匹配的前缀相对应的文件类型，若是都不匹配，则我们判断是否是文本文档类型。判断是否是文本文档类型时，我们的规则是，如果此文件不是我们可以确定判断的11种文件类型，则认为此文件是文本文档，否则，认为其不是文本文档。

进行文件类型判断之后，如果判断出来此文件的类型不属于用户要过滤的任何一种文件类型，则放弃构建查询字典，继续进行遍历，查找下一个传递文件参数的数据包。否则，若判断出来的文件类型属于用户要过滤的某种文件类型，则按照（7）中第二种方式查找到所有的数据包，当然，丢包时会告知用户。查找完此文件之后，会继续遍历没有遍历的部分，继续查找时候还有文件类型满足用户的过滤要求。

遍历完之后，会跳出提示框提示我们过滤出来了几个文件，并且按出现的顺序将文件类型列出，过滤之后的数据包（包括传递文件参数的数据包，方便重组文件）会显示在PktTab中，用户可采用（7）中两种方式中的任意一种进行重组文件，由于我们已将文件类型按顺序告知用户，此时用户可以确定的选择文件类型。

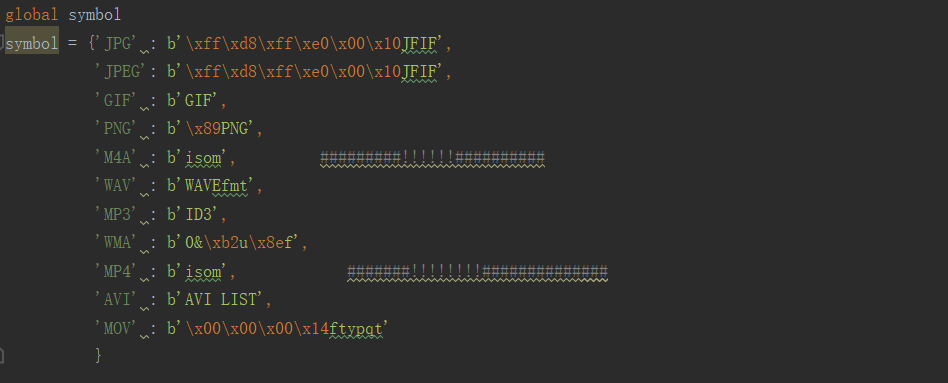
当然，如果是文本文档类型，提示信息中会显示‘TXT’，不过用户当然可以将其存储为‘doc’或‘docx’类型的文件。由于JPG与JPEG实际上可以认为是同一种文件类型，所以在勾选要过滤的文件类型时只选择其中一个没有任何问题，当然，如果都勾选也不会报错，在提示框中会同时显示这两种类型。

此方法的缺陷是无法区分M4A和MP4文件，我们没有找到区分二者二进制前缀的特殊标识符。

此功能仅适用于用QQ传输文件，对于使用其他方式传输文件，本方式不保证能够成功。事实上，（7）中的b方式使用的环境，本功能也适用。

此功能支持对乱序到达的数据包进行过滤。

同样，由于本部分代码过长，只给出入口函数的部分代码及全局变量字典symbol。

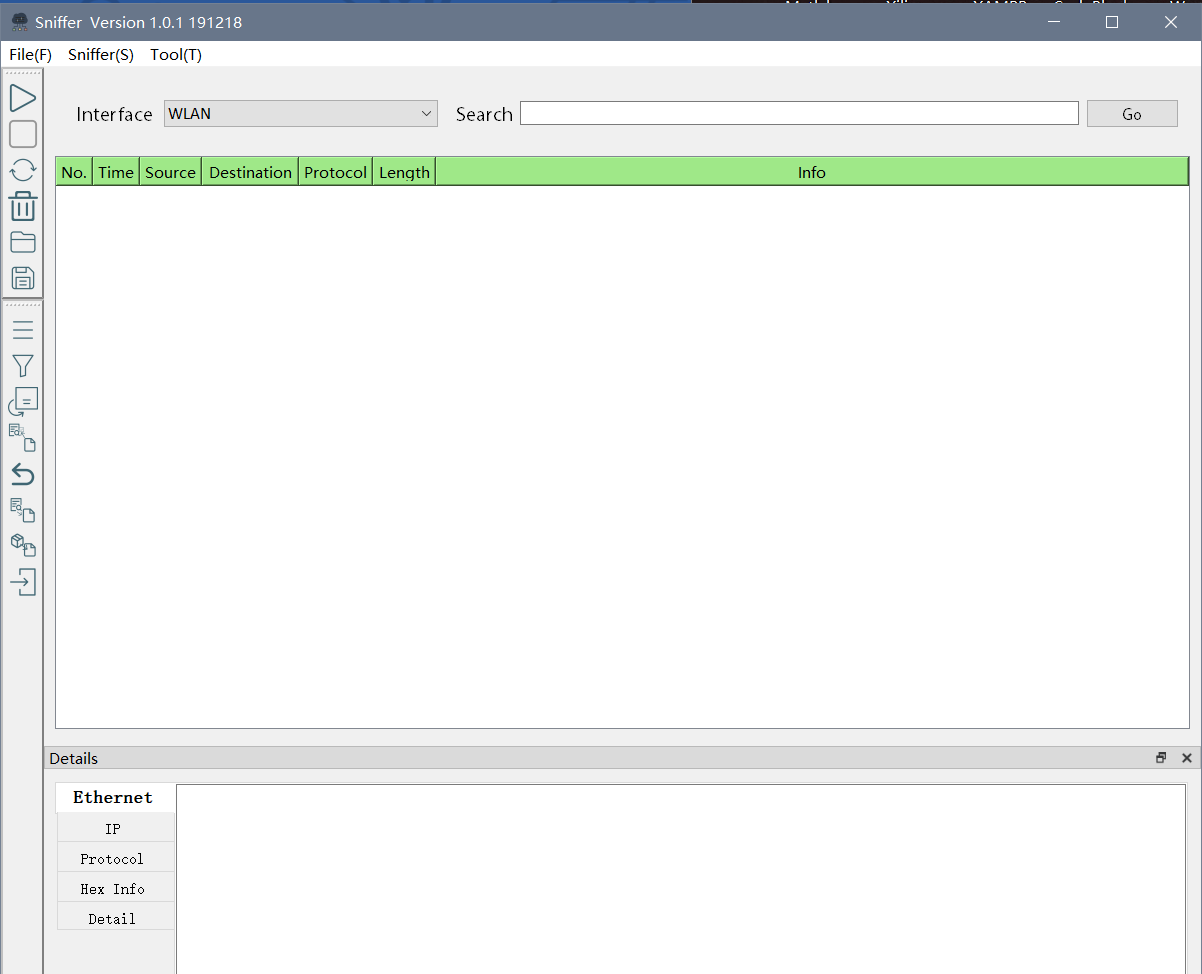




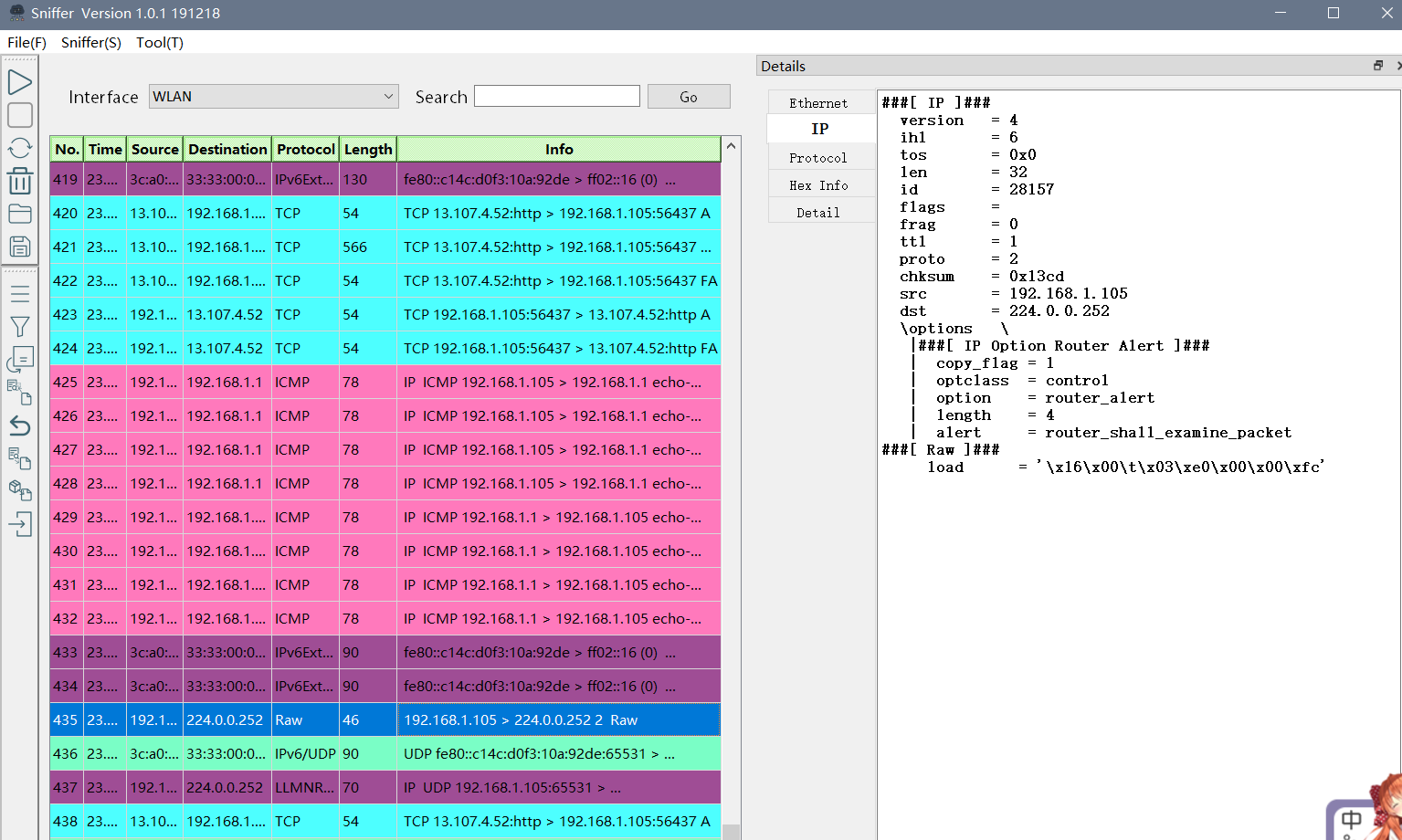
**四.测试过程**

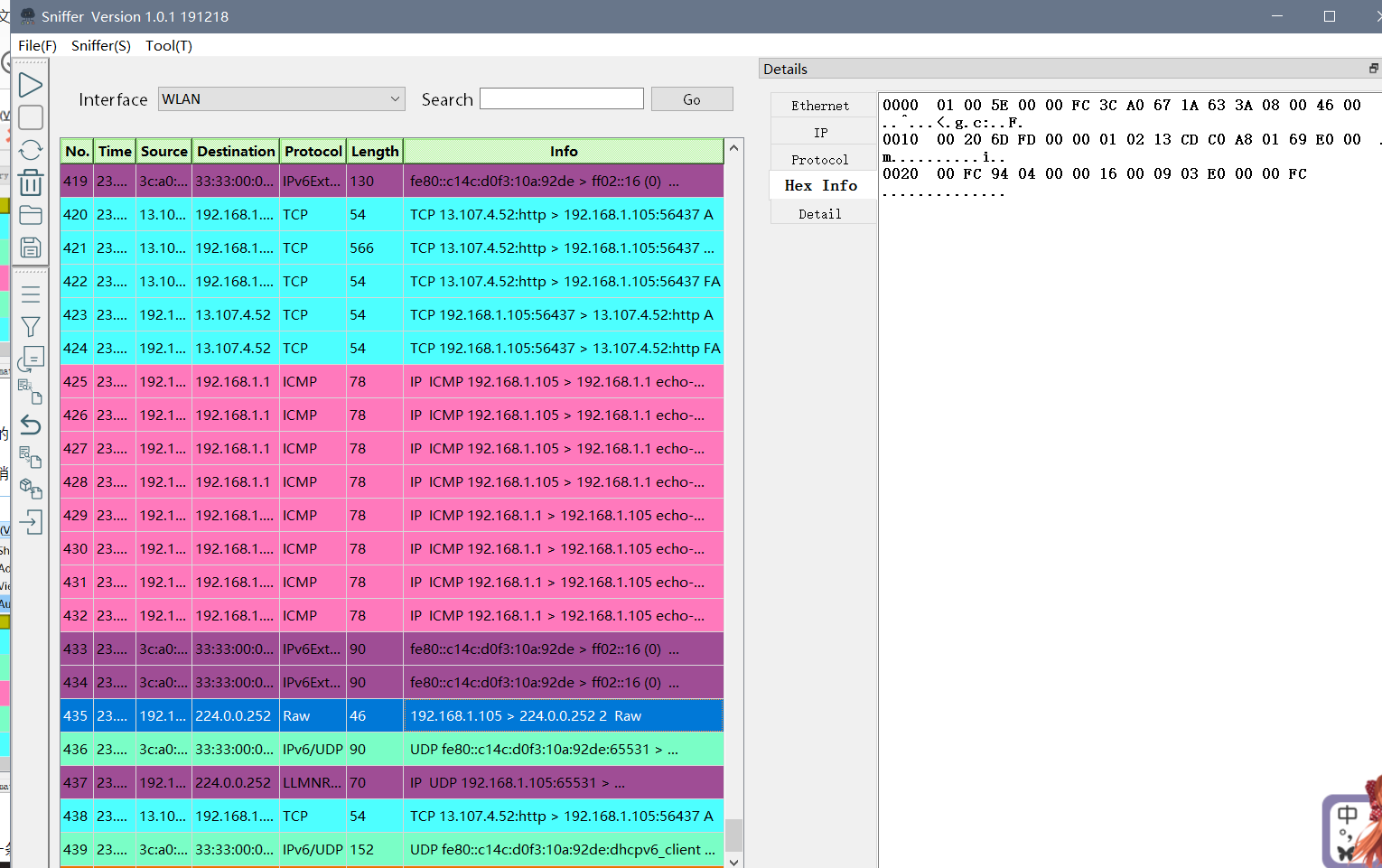
**注：由于程序功能较为复杂，尽在此部分提供截图来显示测试部分，详细功能测试请见录屏文件或自行运行程序测试。**

**1.对指定网卡抓包并解析，换网卡**

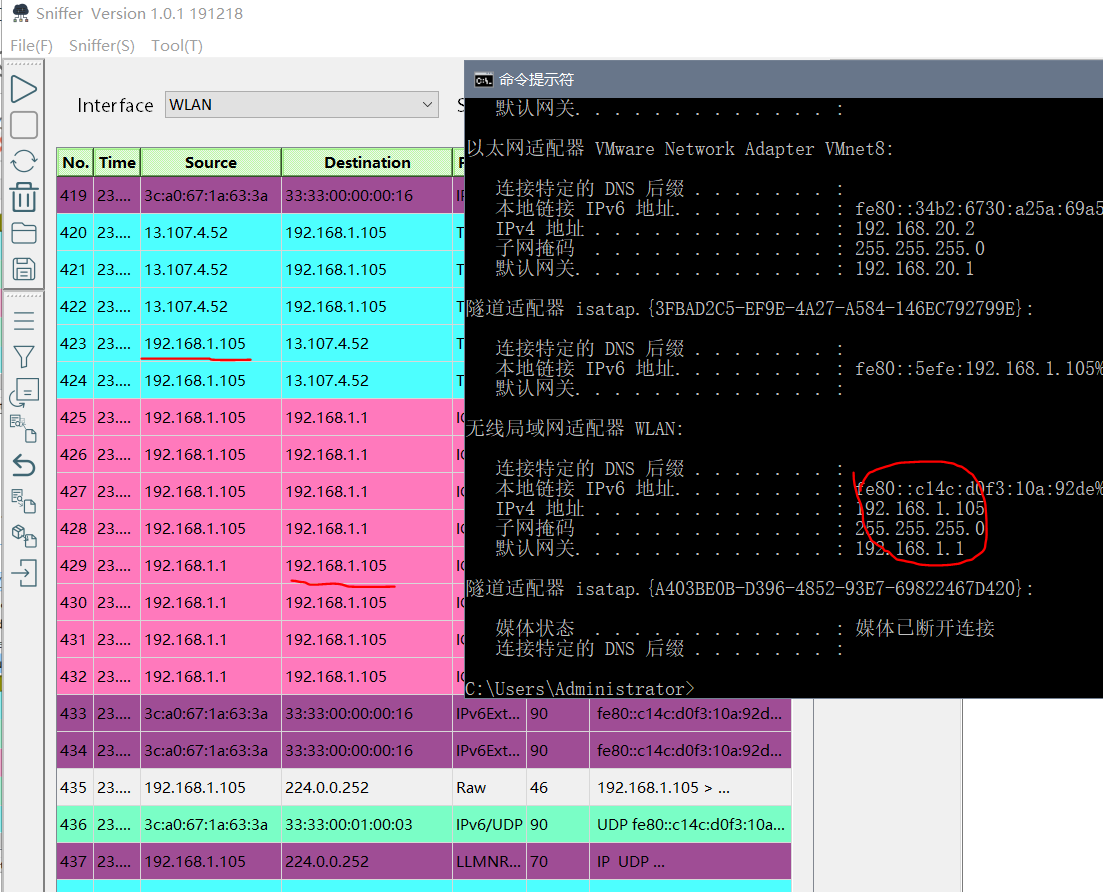


打开程序，在接口列表中选择WLAN（无线网卡），点击左边的开始键开始抓包。为了观察方便把Details框拖到右边，调整大小。双击某个包，在Details框中显示详细信息，点击各栏目查看。

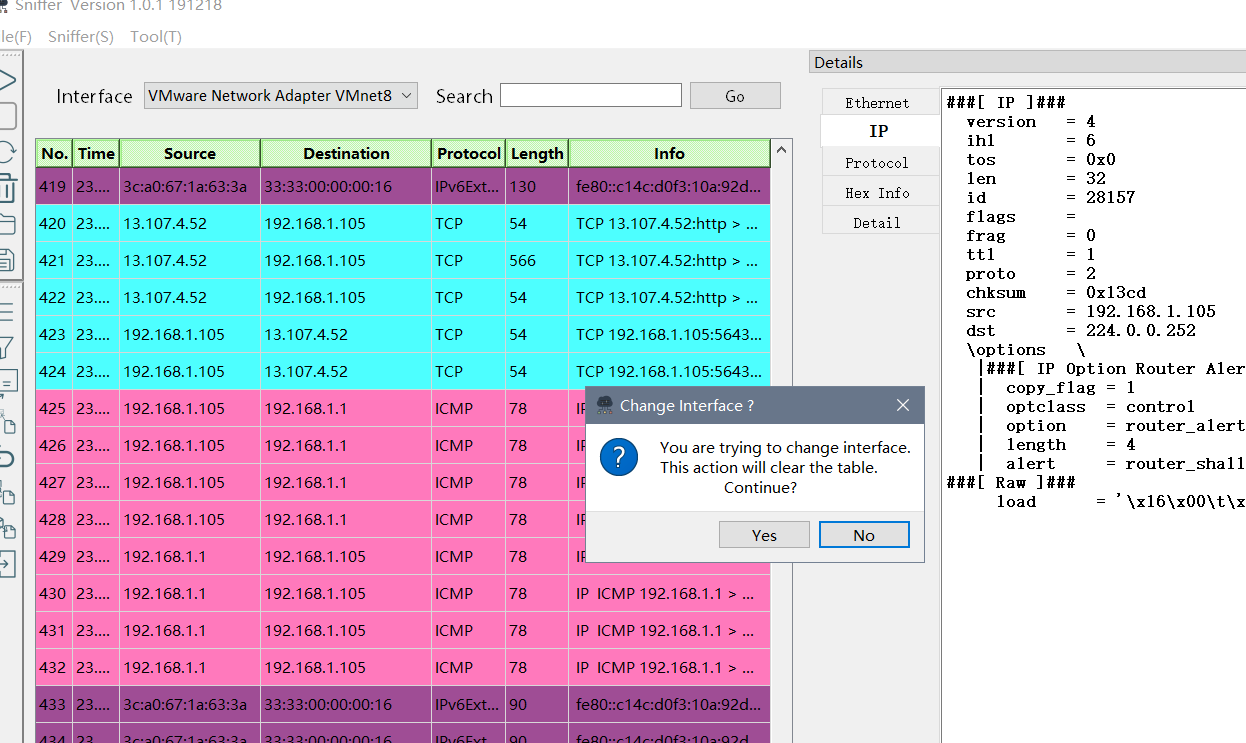


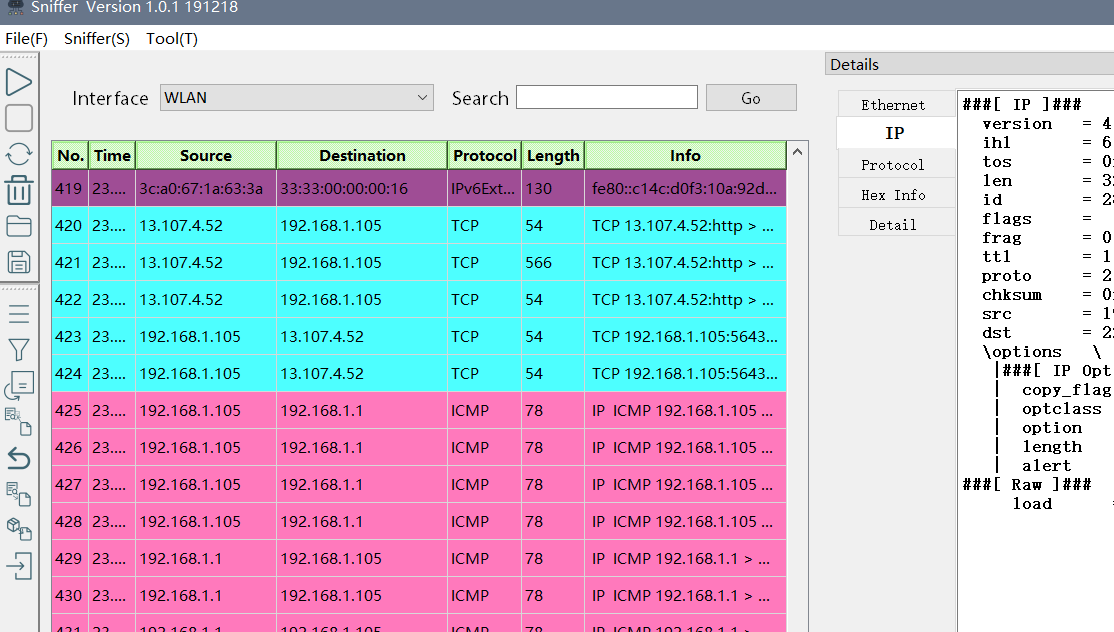


对比ipconfig中的ip和源地址目的地址，确认侦听的是选定的网卡。

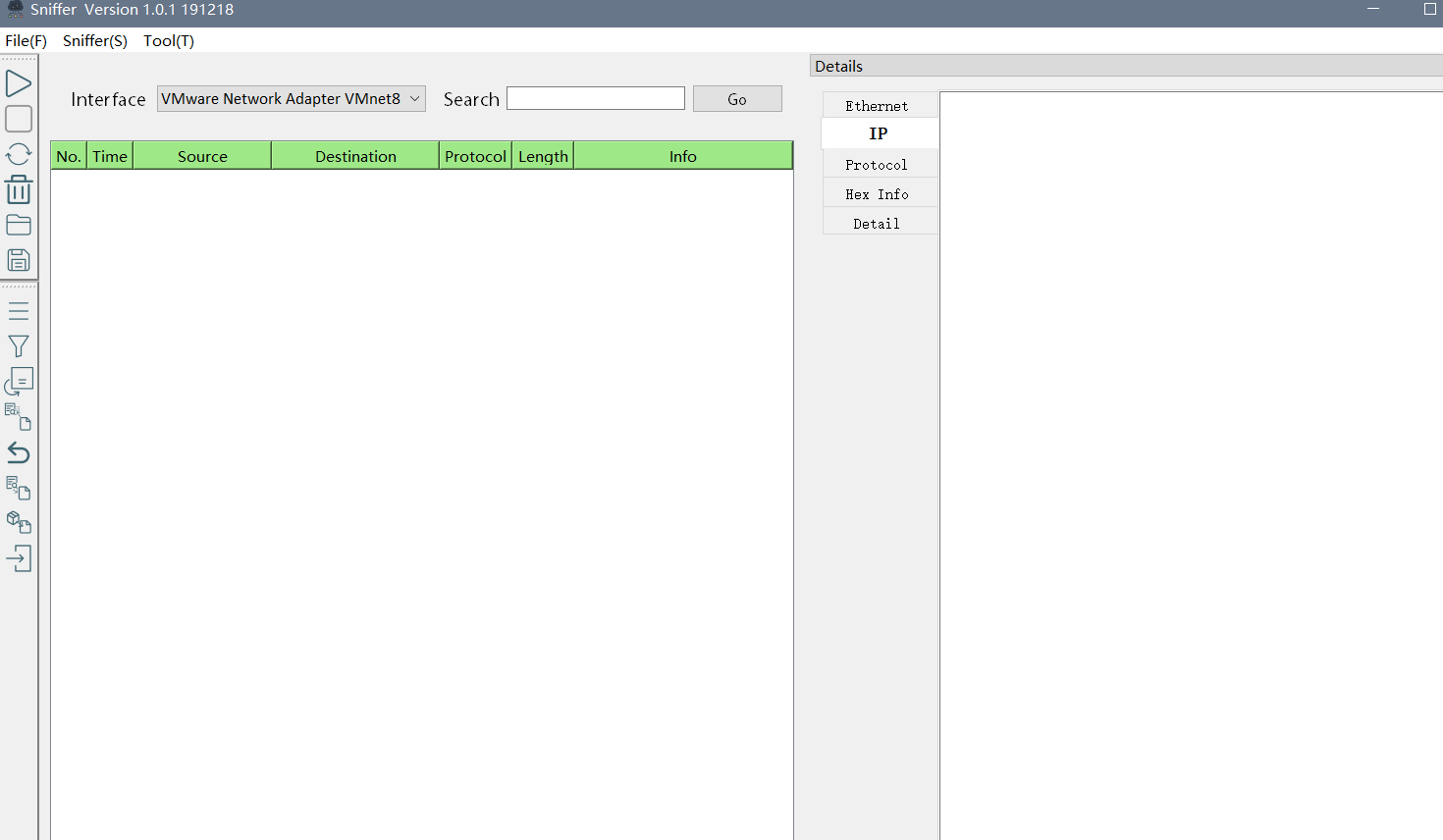


更换网卡，跳出提示信息，选择NO，回到原网卡。

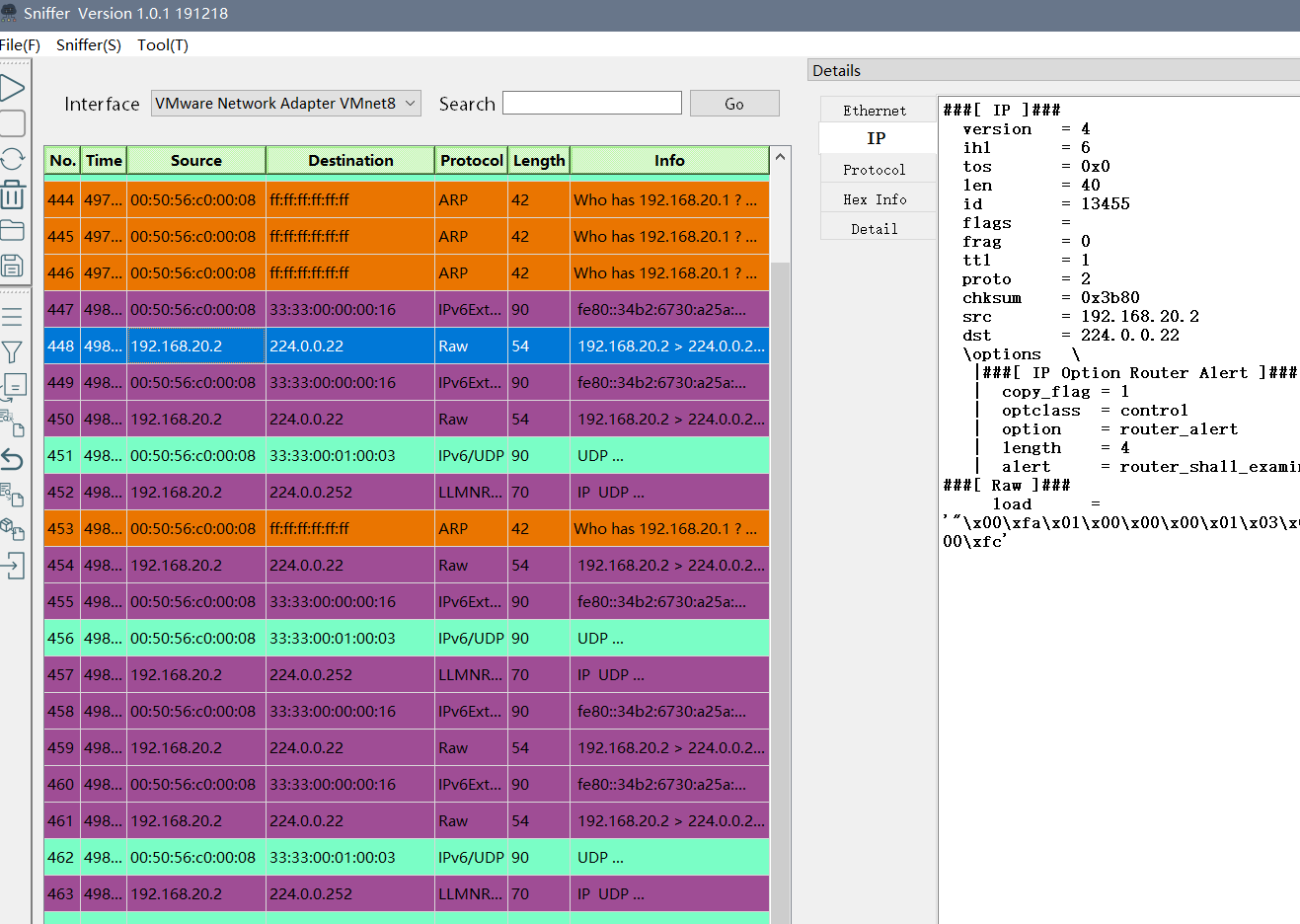




再次更换网卡，选择Yes，看到原数据全部清空。

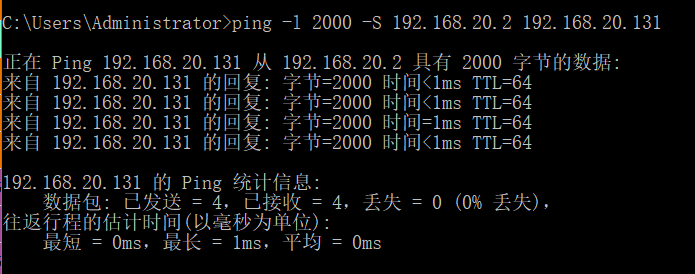


点击开始侦听该网卡。

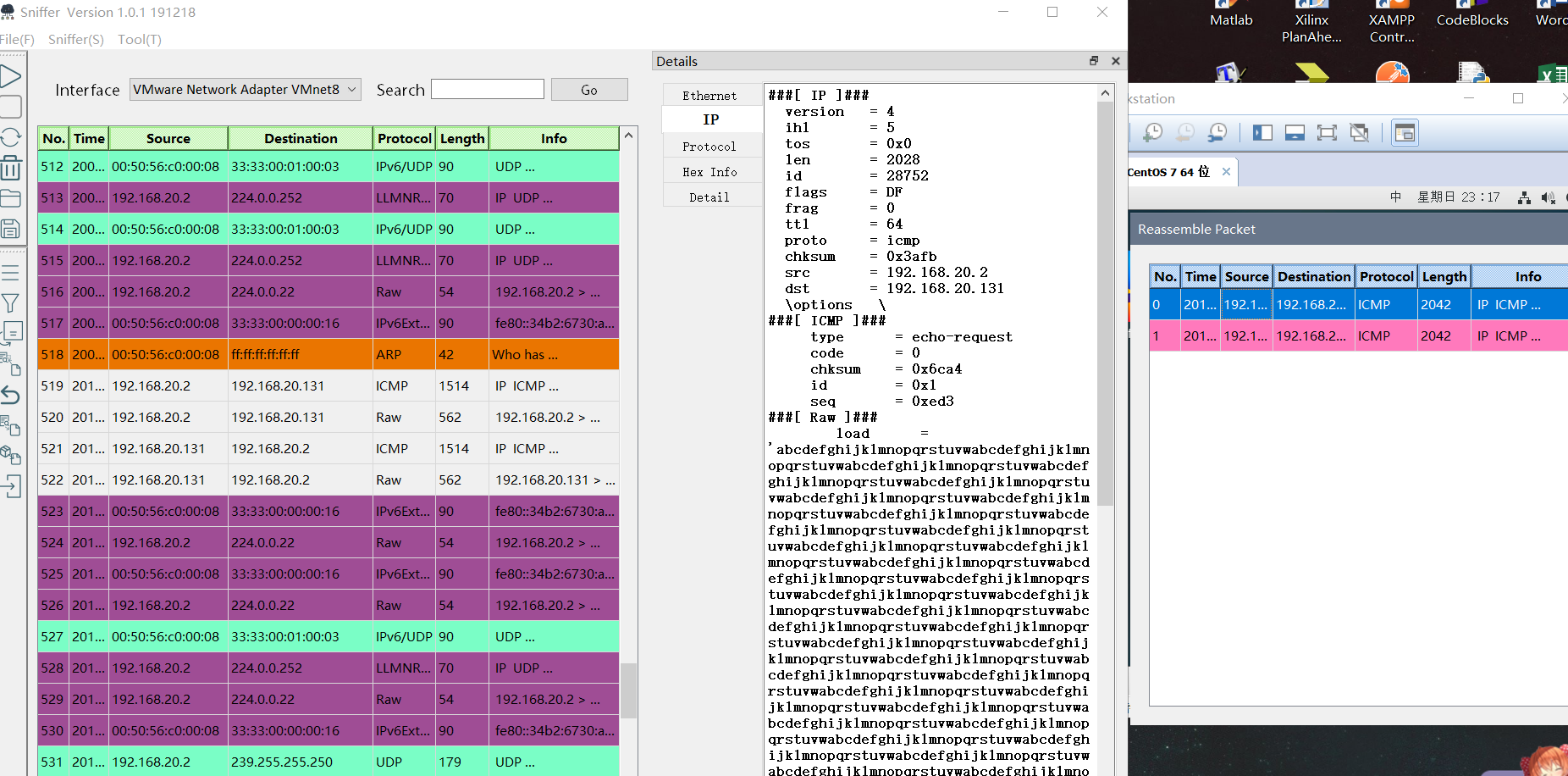


**2.包分片重组**

监听VMWareNet8，使用ping命令发长为2000的包给虚拟机192.168.20.131。



找到ICMP包与Raw包，选择后点击左边的重组按钮，Reassemble窗口（此处我们将其拖了出来）中显示重组后包，双击在Details中查看包信息。

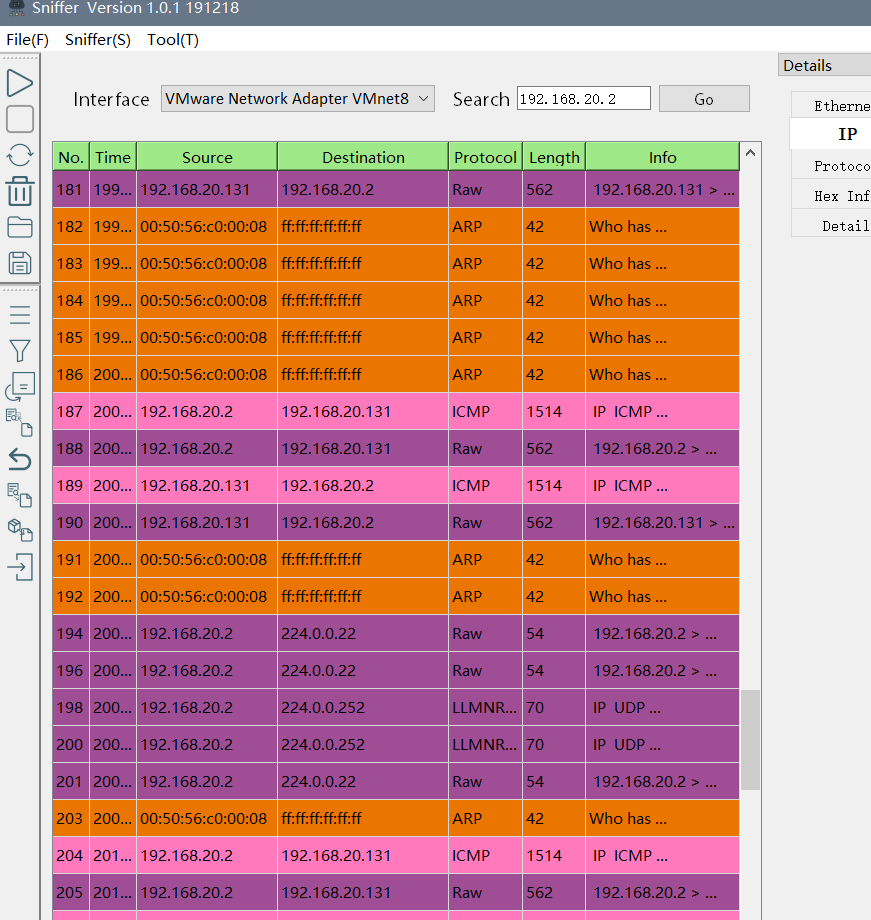


重组得到的数据包的长度为2042字节，其中有14字节的数据链路层信息，20字节的IP头部，8字节的ICMP头部，剩余部分刚好为2000字节。

**3.数据包查询**

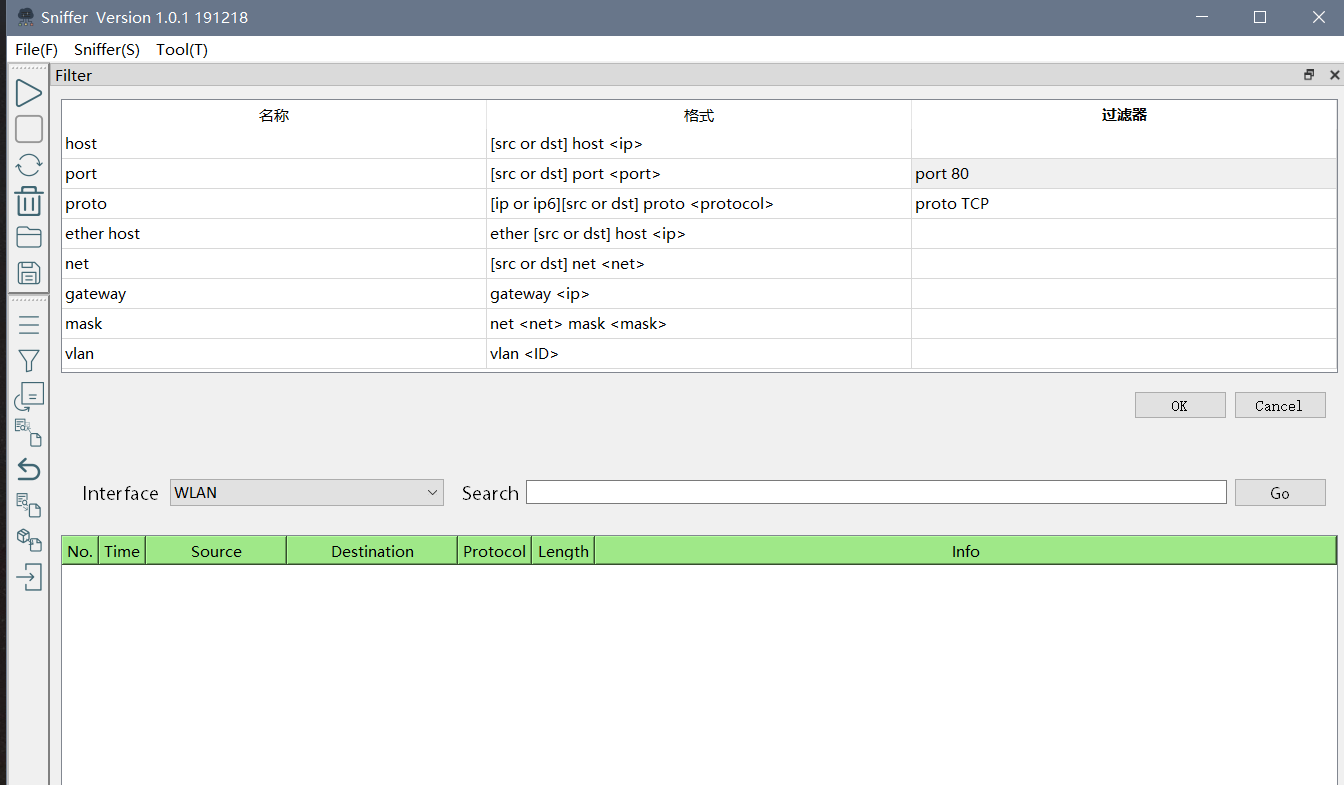
接2，我们再使用查找来找这个包，search 192.168.20.2，可以看到相关的包。（由于查询规则是只要在数据包的任何部分出现查询字段，都算符合条件，所以有些包的Source和Destination并不是192.168.20.2，但是它们的其他部分包含查询条件）

在查询结果中也可以正常解析数据包，使用Back返回功能可以显示原先包含所有包的PktTable界面，详见录屏。

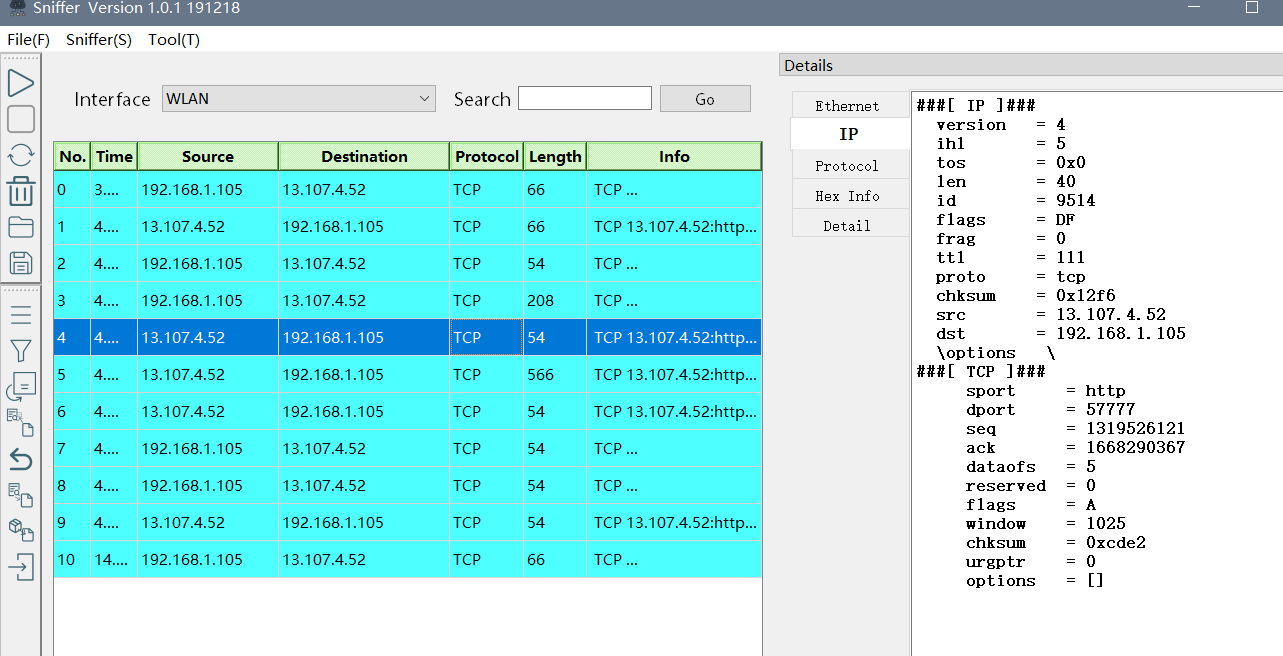


**4.包过滤**

点击Filter并输入规则 port 80, proto TCP。点击OK键应用过滤器，点击开始。

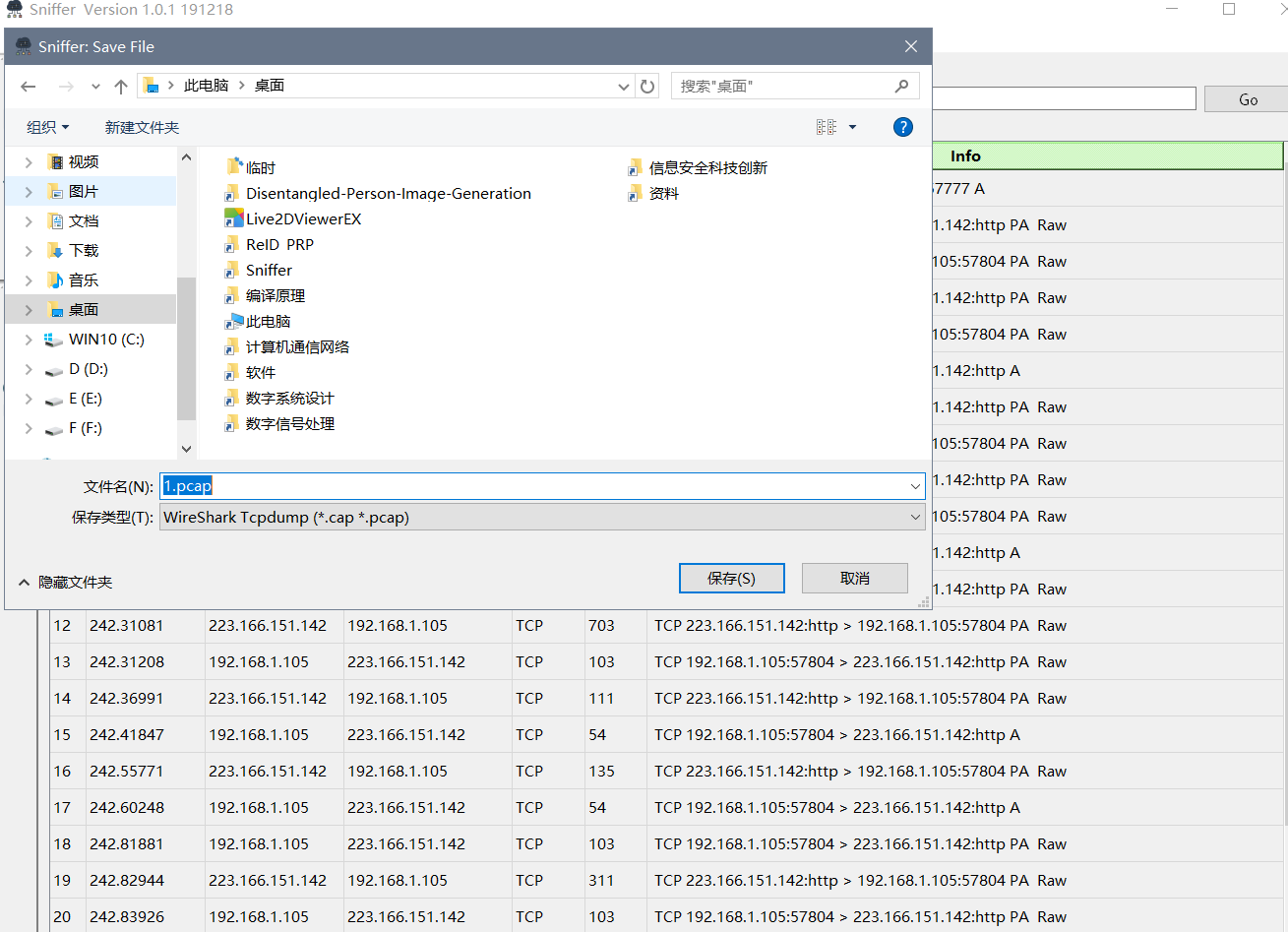


可以看到均是80端口（http）的TCP包。



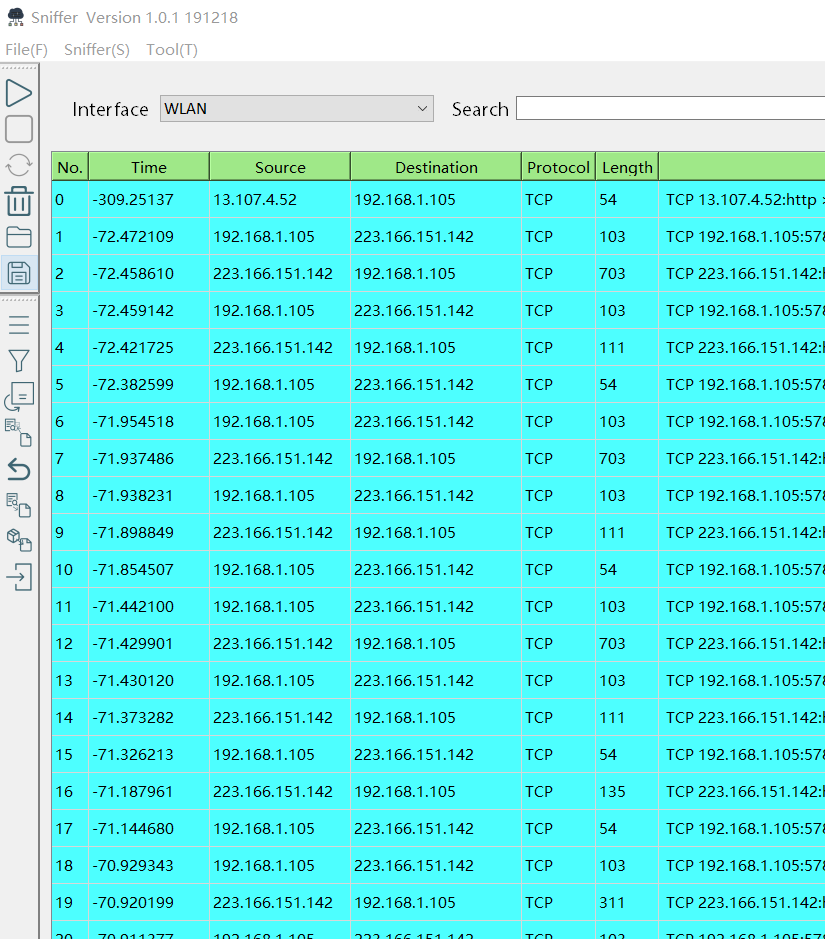
**5.数据包文件保存和读取**

选中一些数据包，点击保存按钮，将刚才的数据包存为文件。

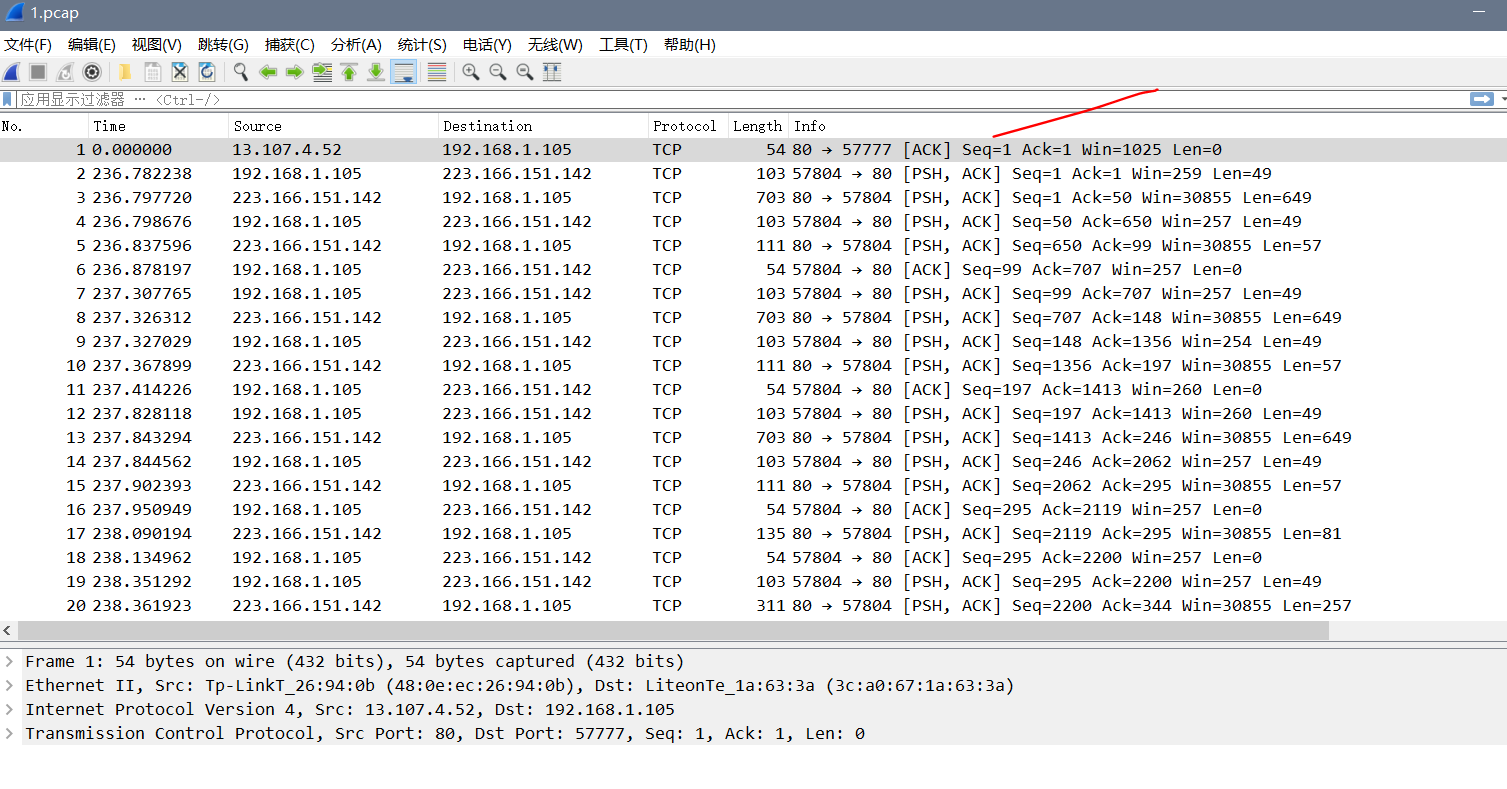


使用本程序和WireShark分别打开该文件，可用性良好。

本程序中，包时间显示为负数，是文件中数据的抓包时间（保存为文件前的时间）到现在的时间差。



WireShark正常打开文件。

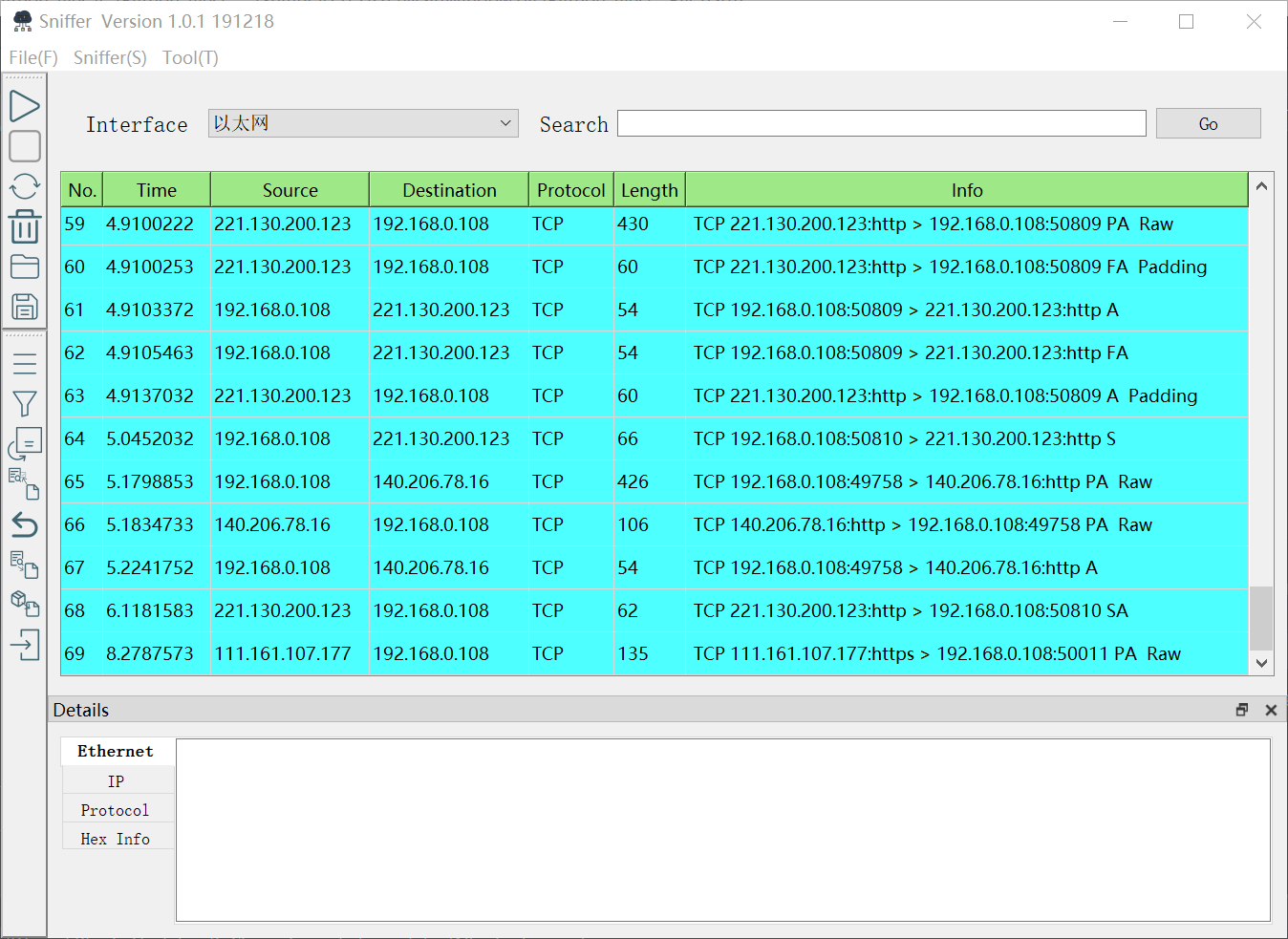


**6.文件重组**

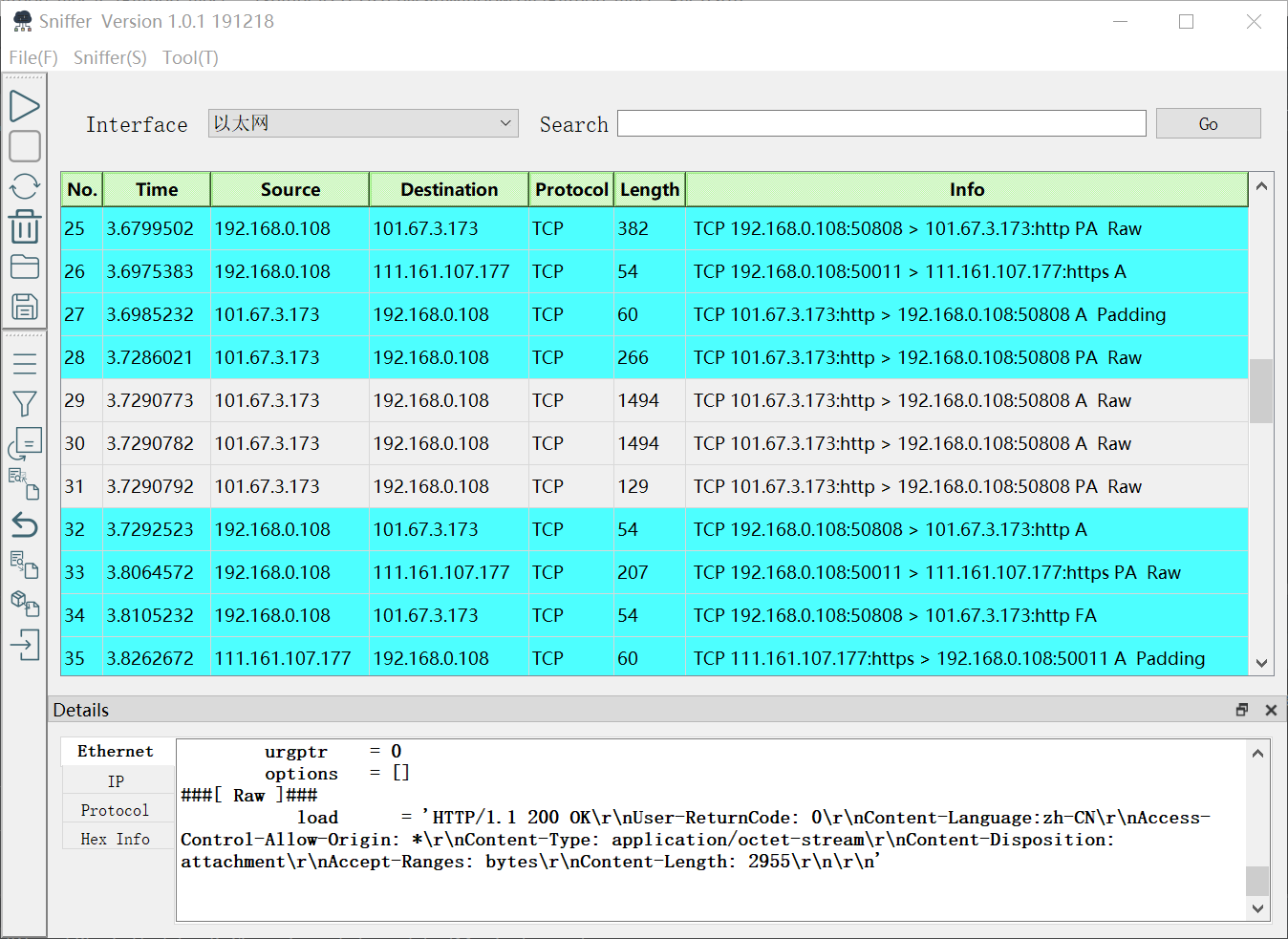
在本测试部分提供对TXT文件、JPG文件、MP3文件、MP4文件的重组测试。

1. 使用a方法测试重组TXT文件与JPG文件

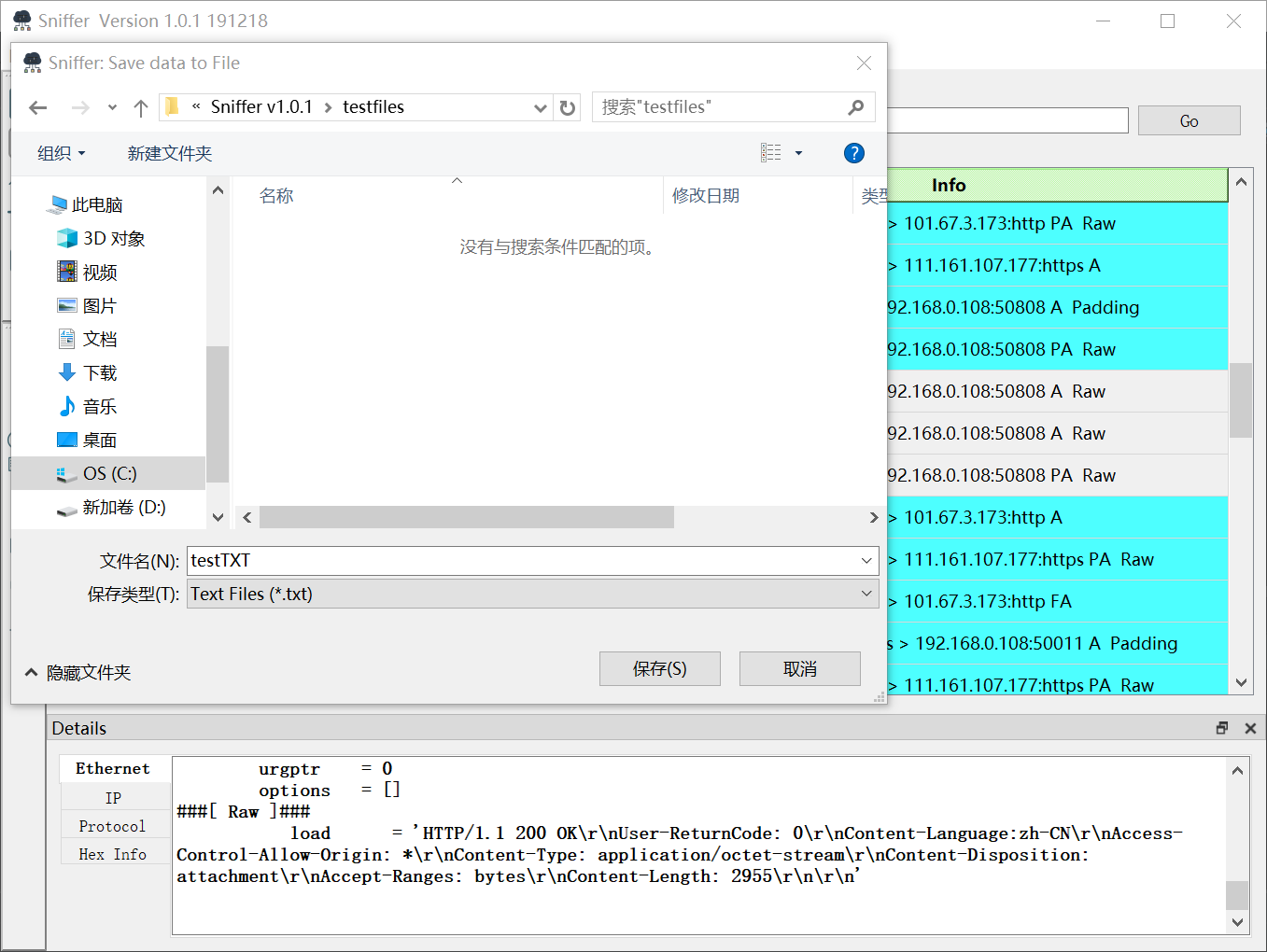
首先，我们通过QQ传输TXT文件，并用Sniffer抓包，抓包结果如下



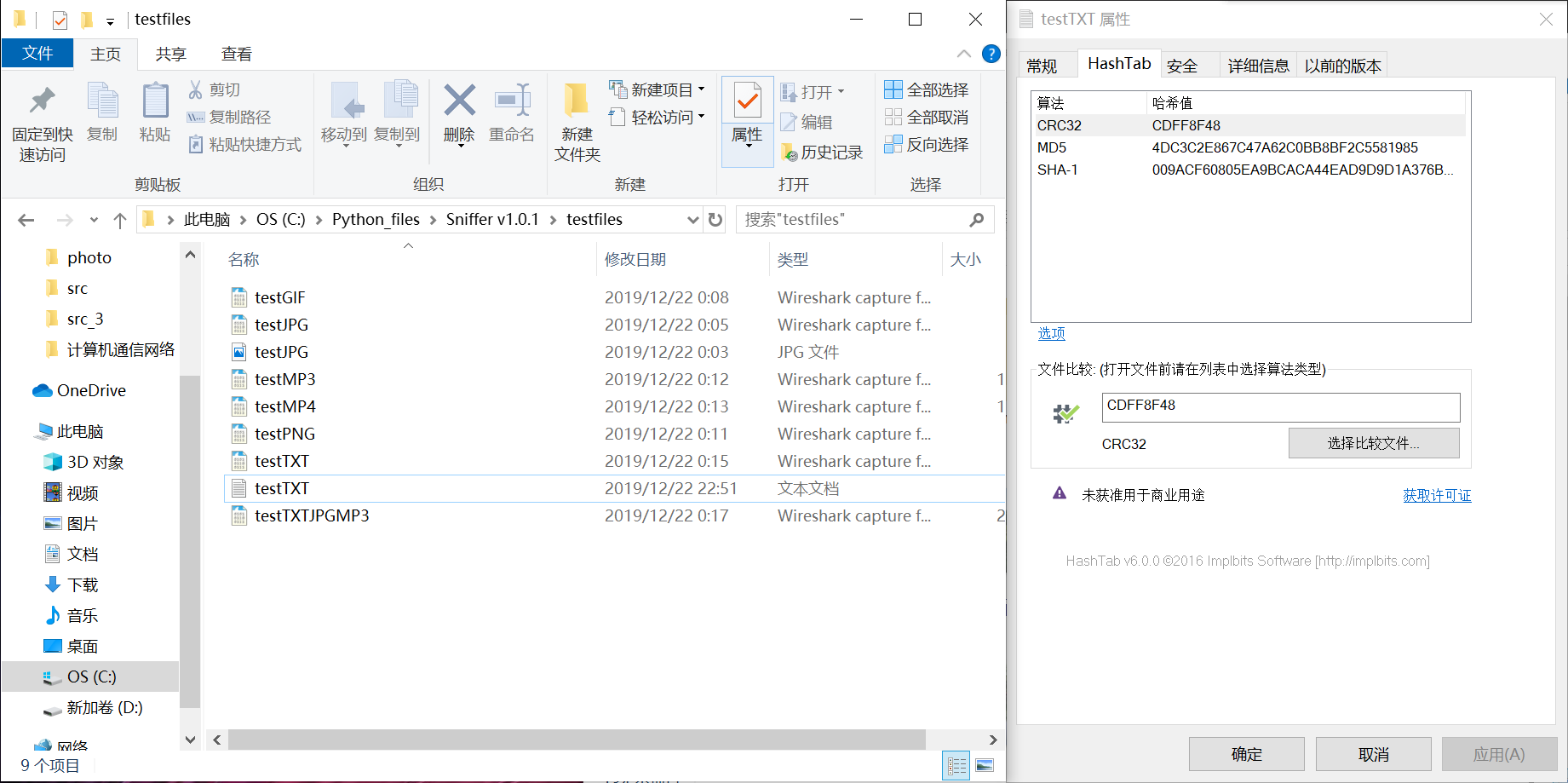
找到所有数据包，在第29、30、31行，实际上寻找文件的数据包并不难，文件一般比较大，需要十几个或者几十个接近MTU的数据包传输，而这些数据包基本都是连在一起的，数据包可以乱序到达，对最后结果不会有影响，寻找到的数据包如下



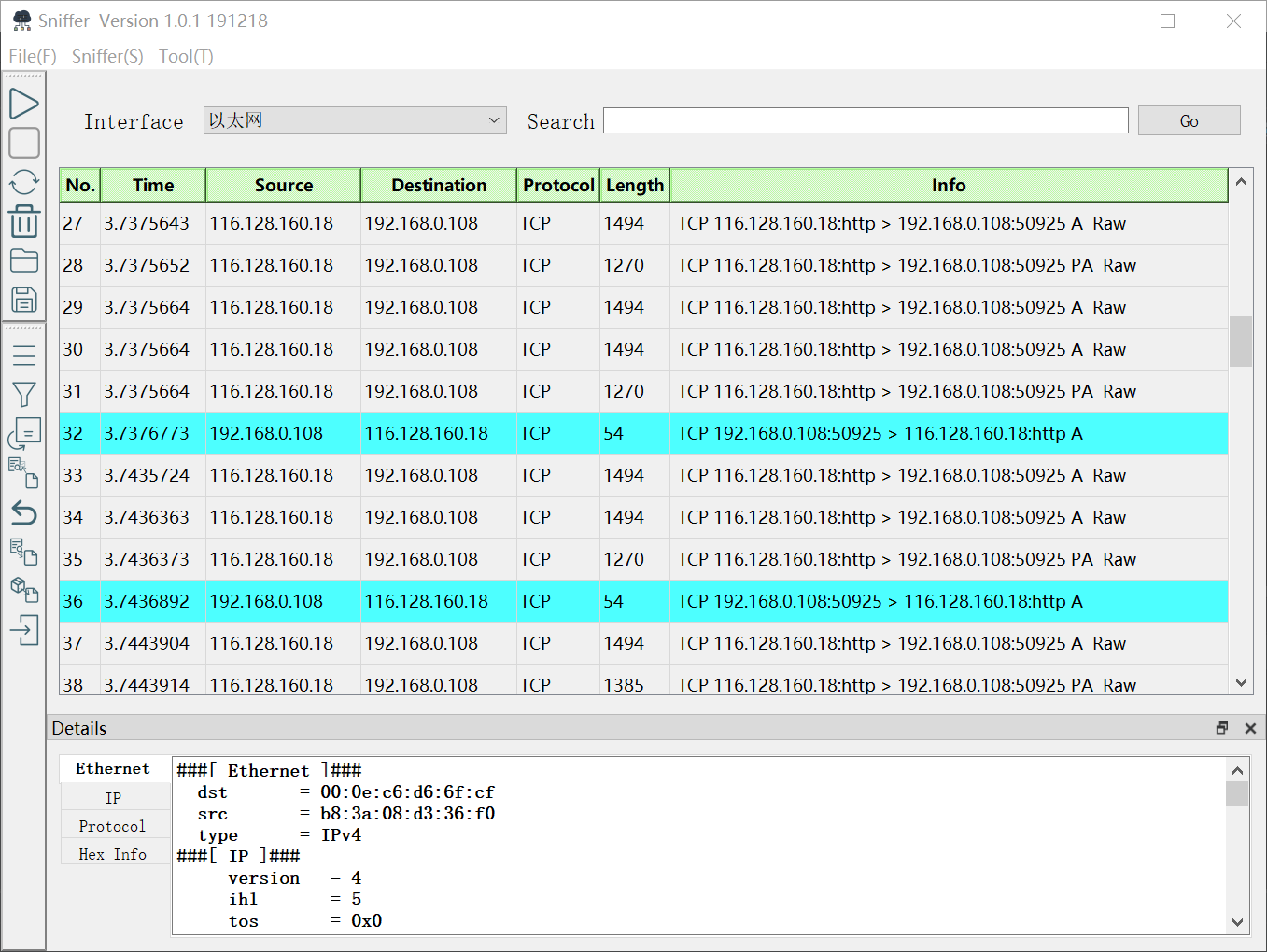
使用TcpToFile按钮重组，生成文件testTXT.txt

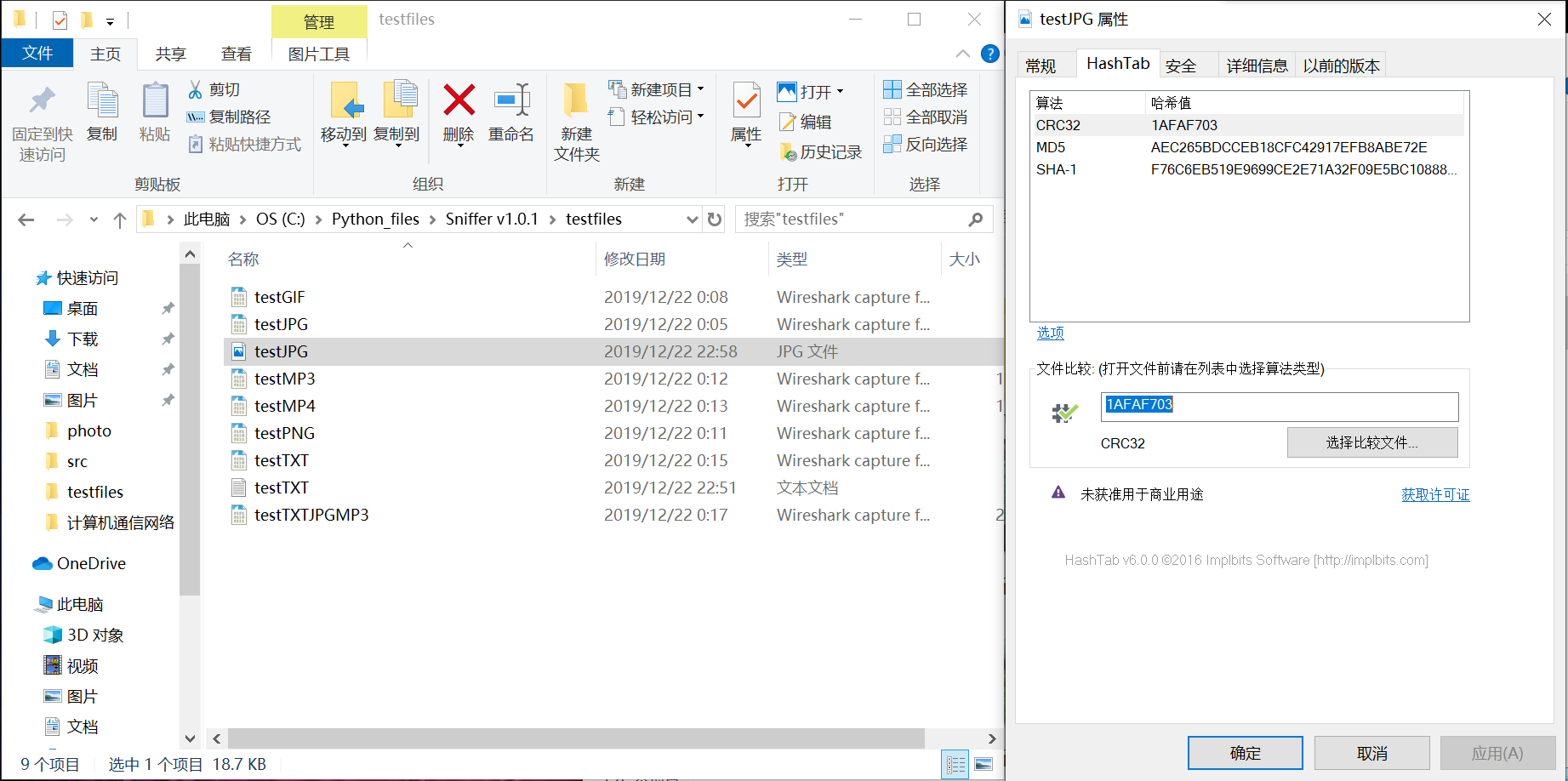


查看生成文件，可以正常打开，并用HashTab进行校验，与原文件一致



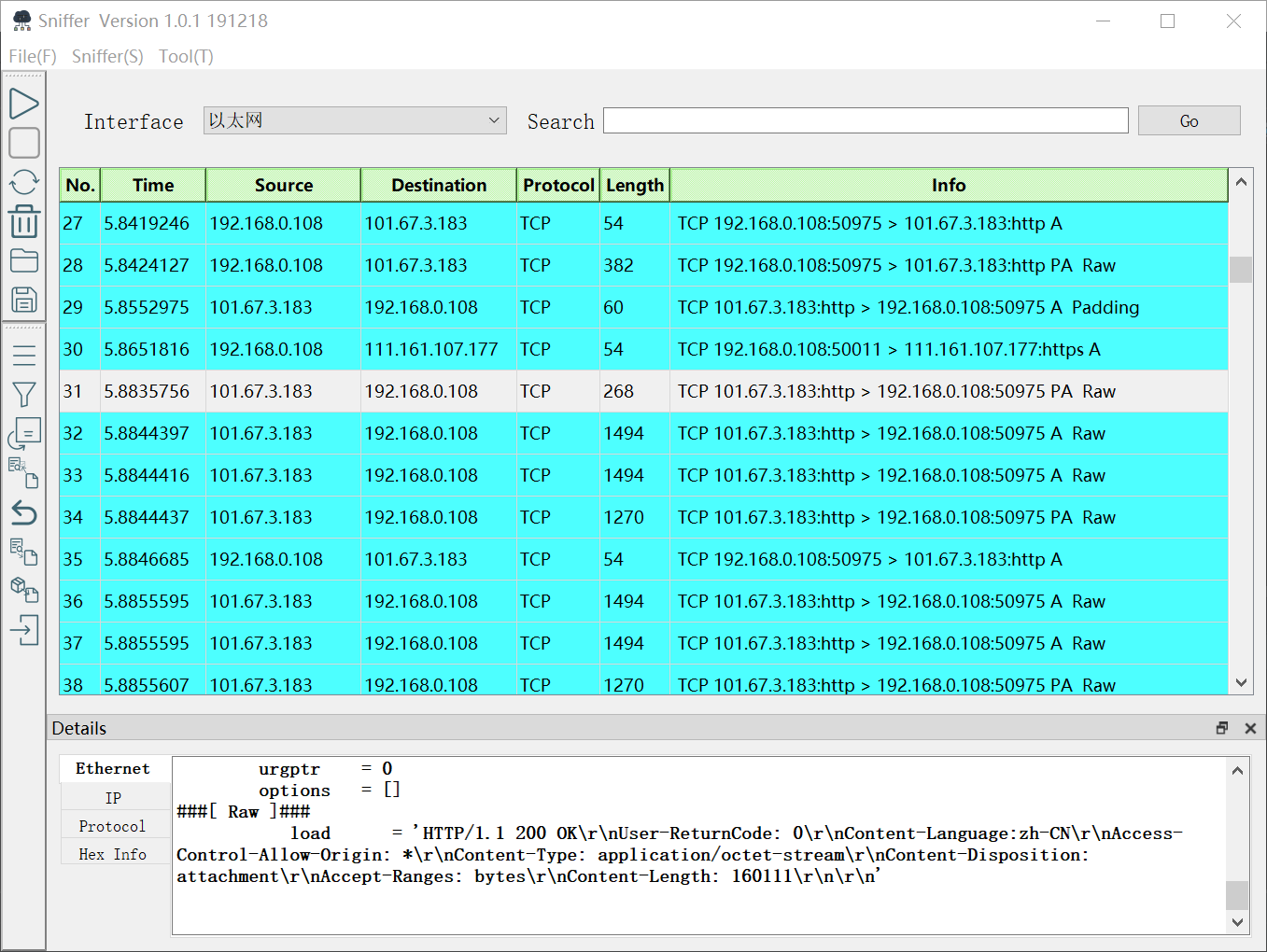
开始JPG文件的测试，同TXT文件一样，先抓包，选中相应的数据包，使用TcpToFile按钮重组，生成testJPG.JPG文件，本处给出选中数据包的截图与生成JPG文件之后使用HashTab进行校验的截图



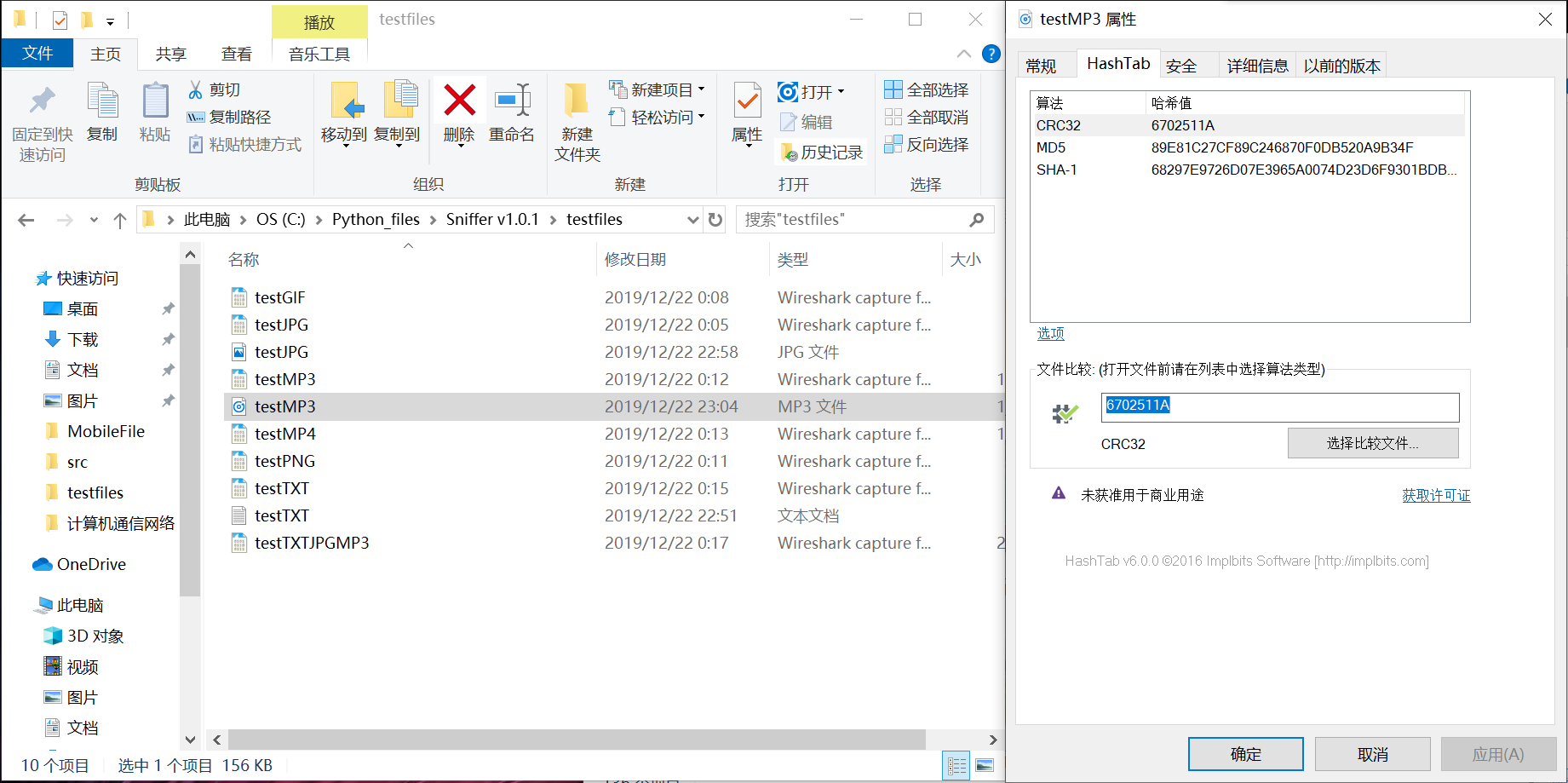


1. 使用b方法测试重组MP3文件与MP4文件

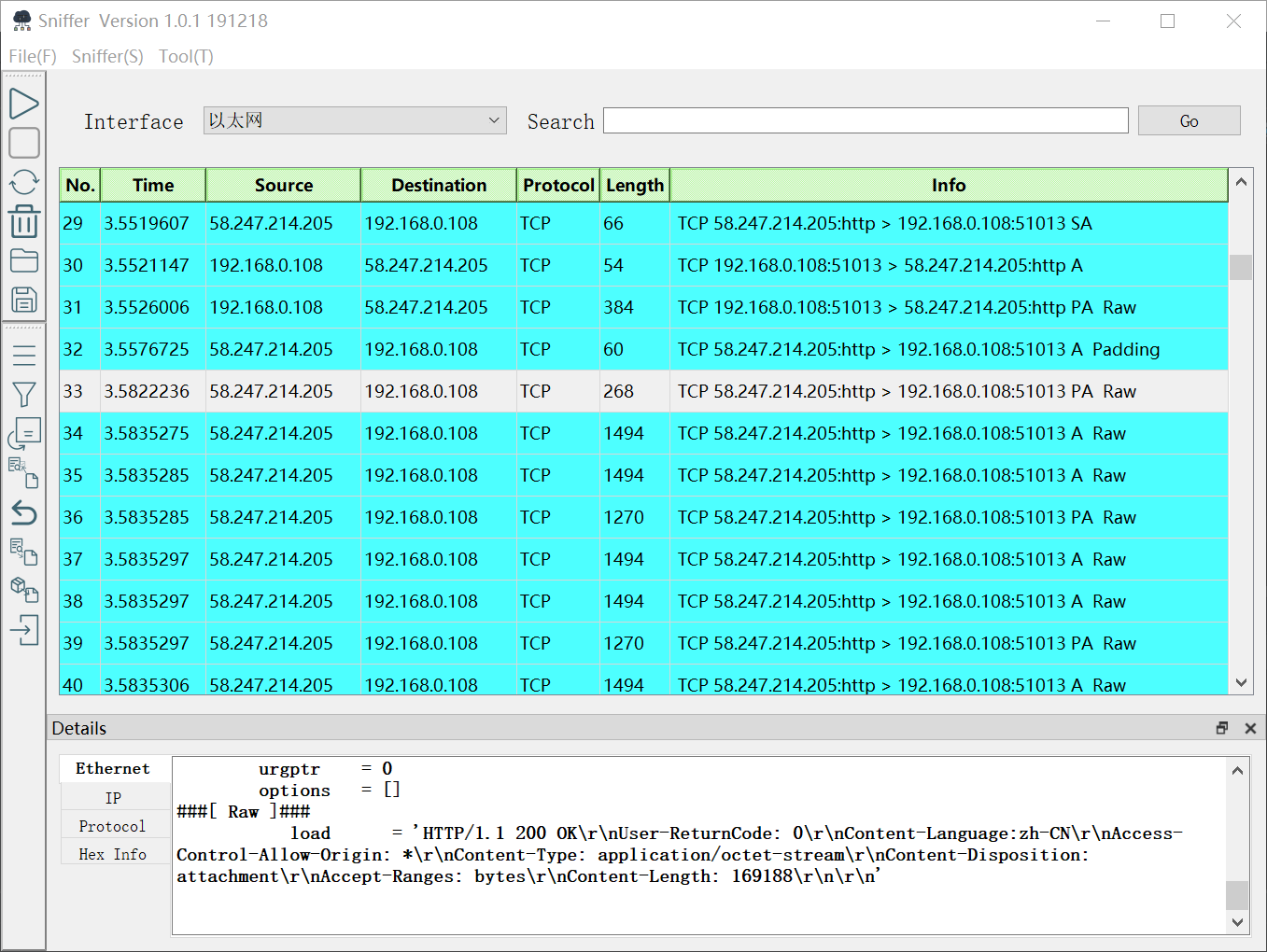
首先，我们通过QQ传输MP3文件，抓包，找到传递文件参数的数据包，如图第31行

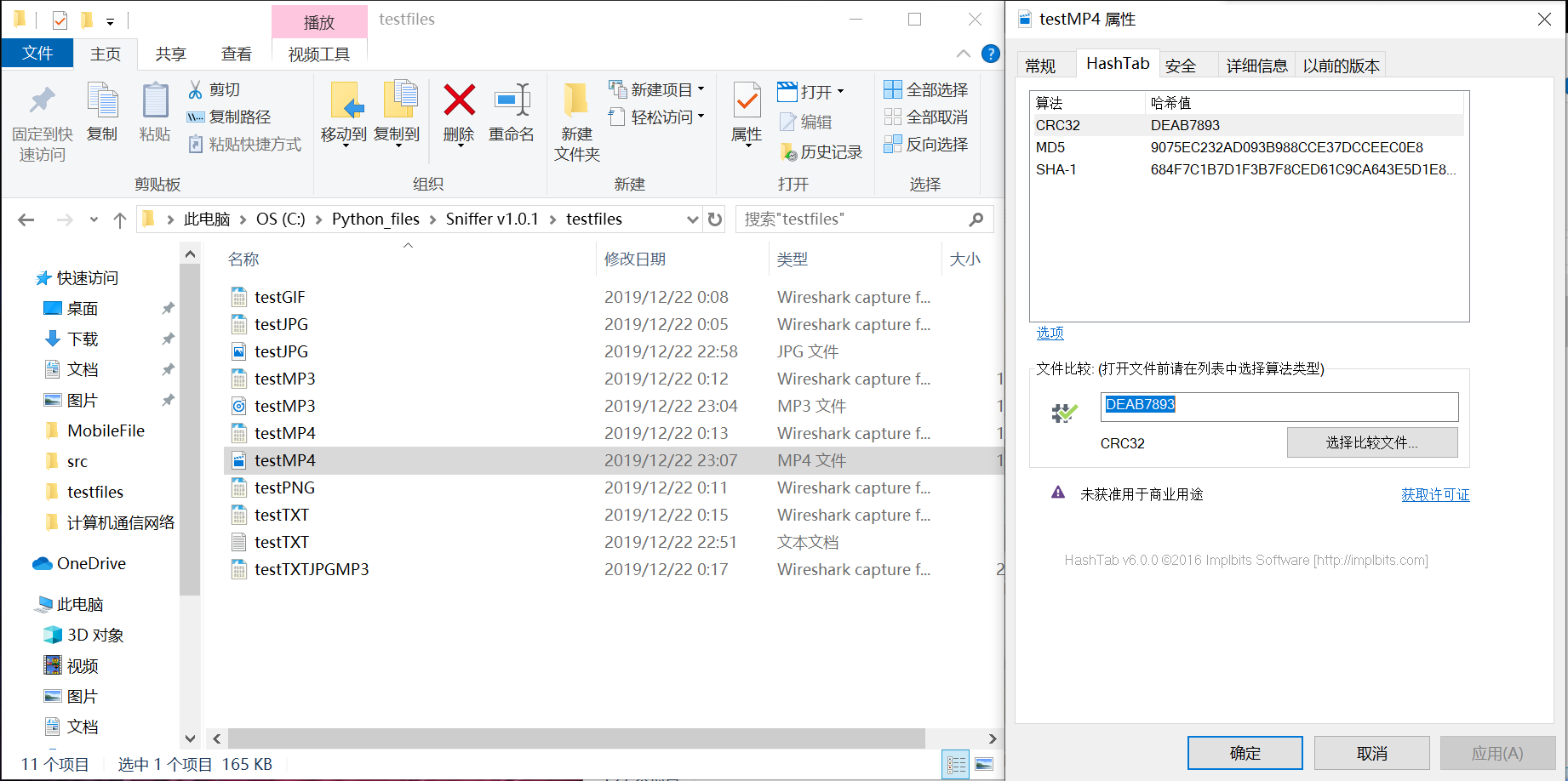


使用SearchToFile按钮重组，生成testMP3.MP3文件，可以正常打开，下面给出使用HashTab对其进行校验的结果



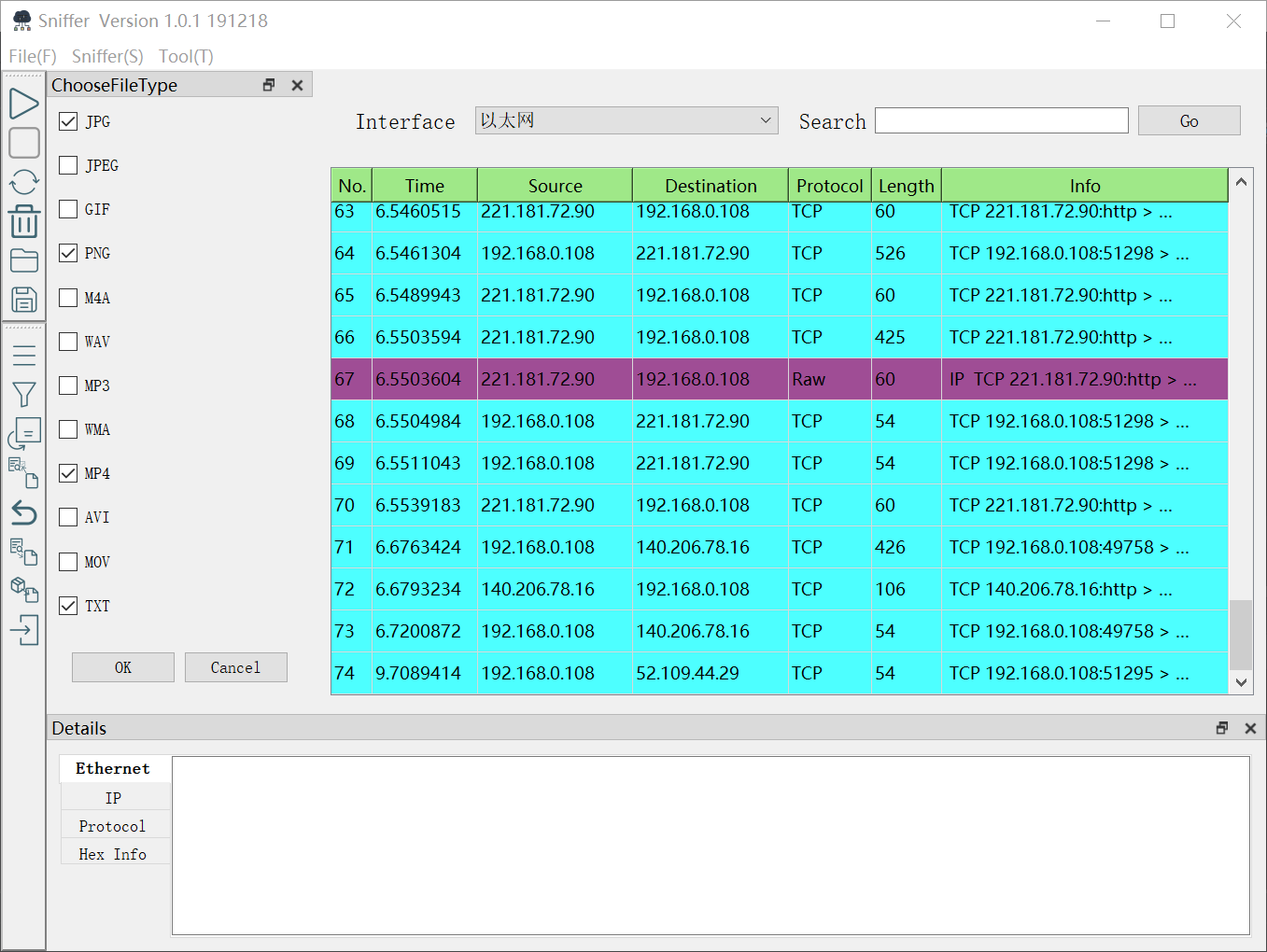
再用QQ传输MP4文件，抓包，找到传递文件参数的数据包，使用SearchToFile按钮重组，生成testMP4.MP4文件，可以正常打开，下面给出数据包截图和使用HashTab对生成的文件进行校验的结果



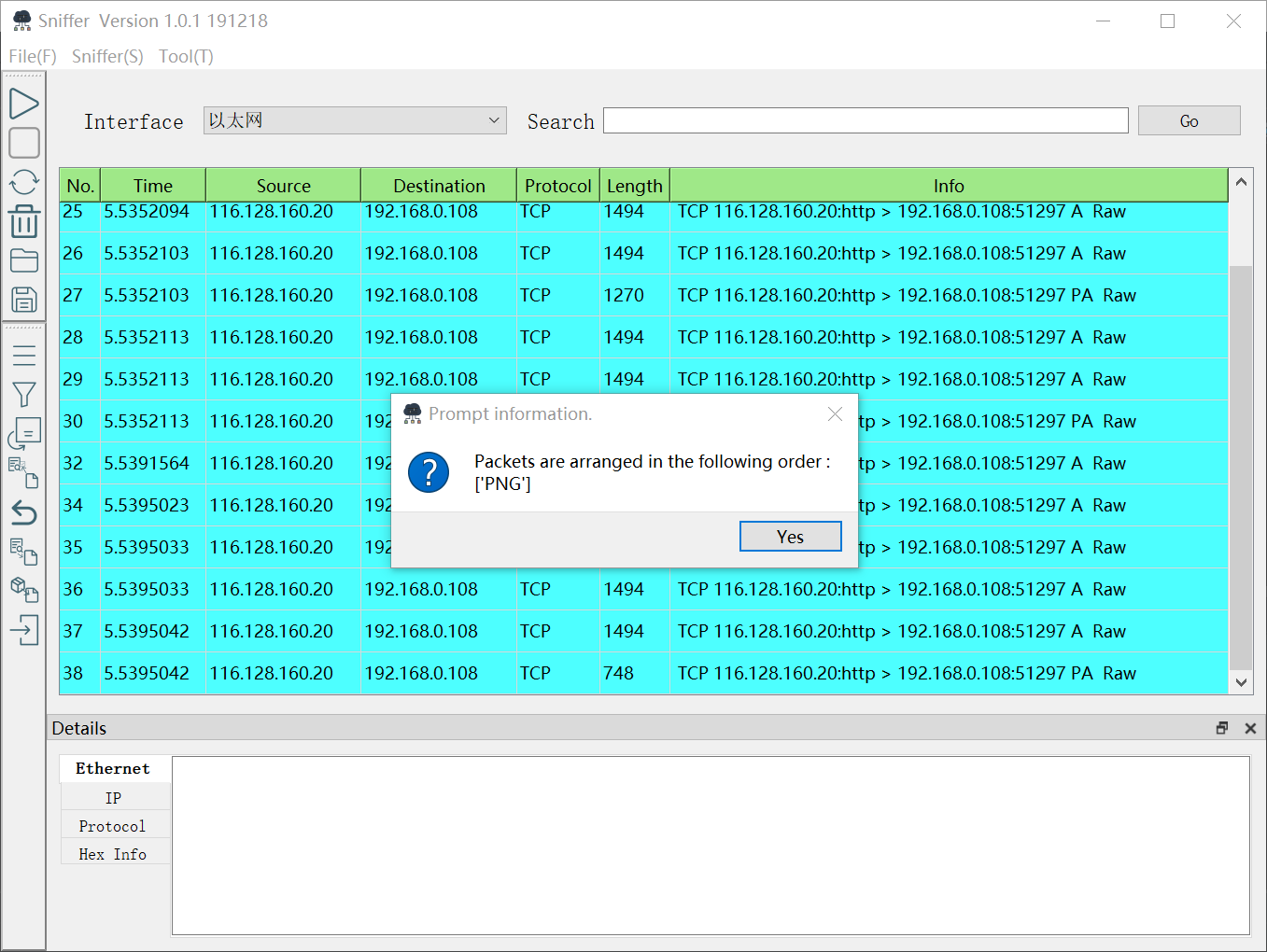


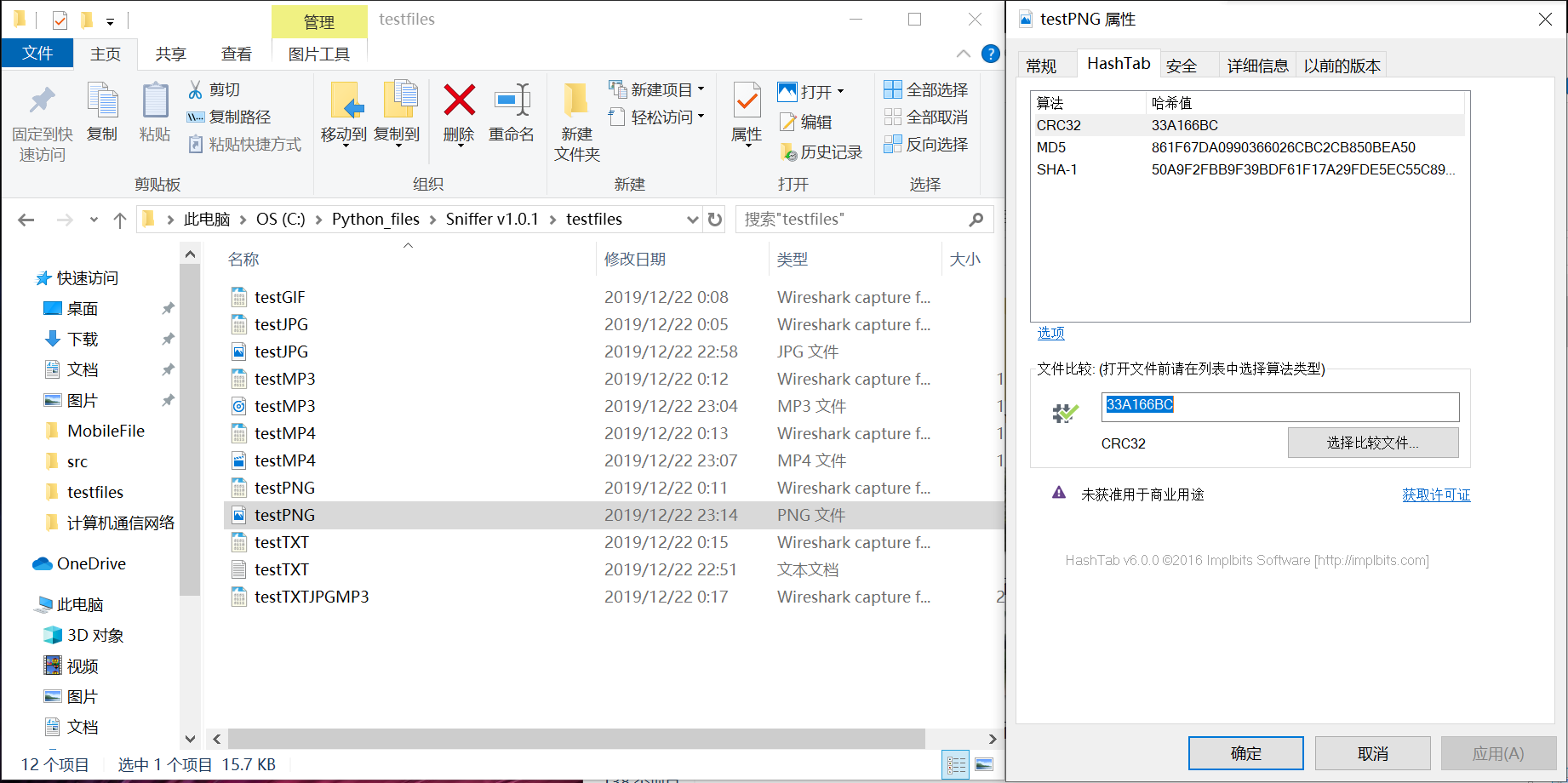
**7.文件过滤**

我们通过QQ传输PNG文件，抓包，在ChooseFileType中勾选包括PNG文件在内的几个文件类型，如下

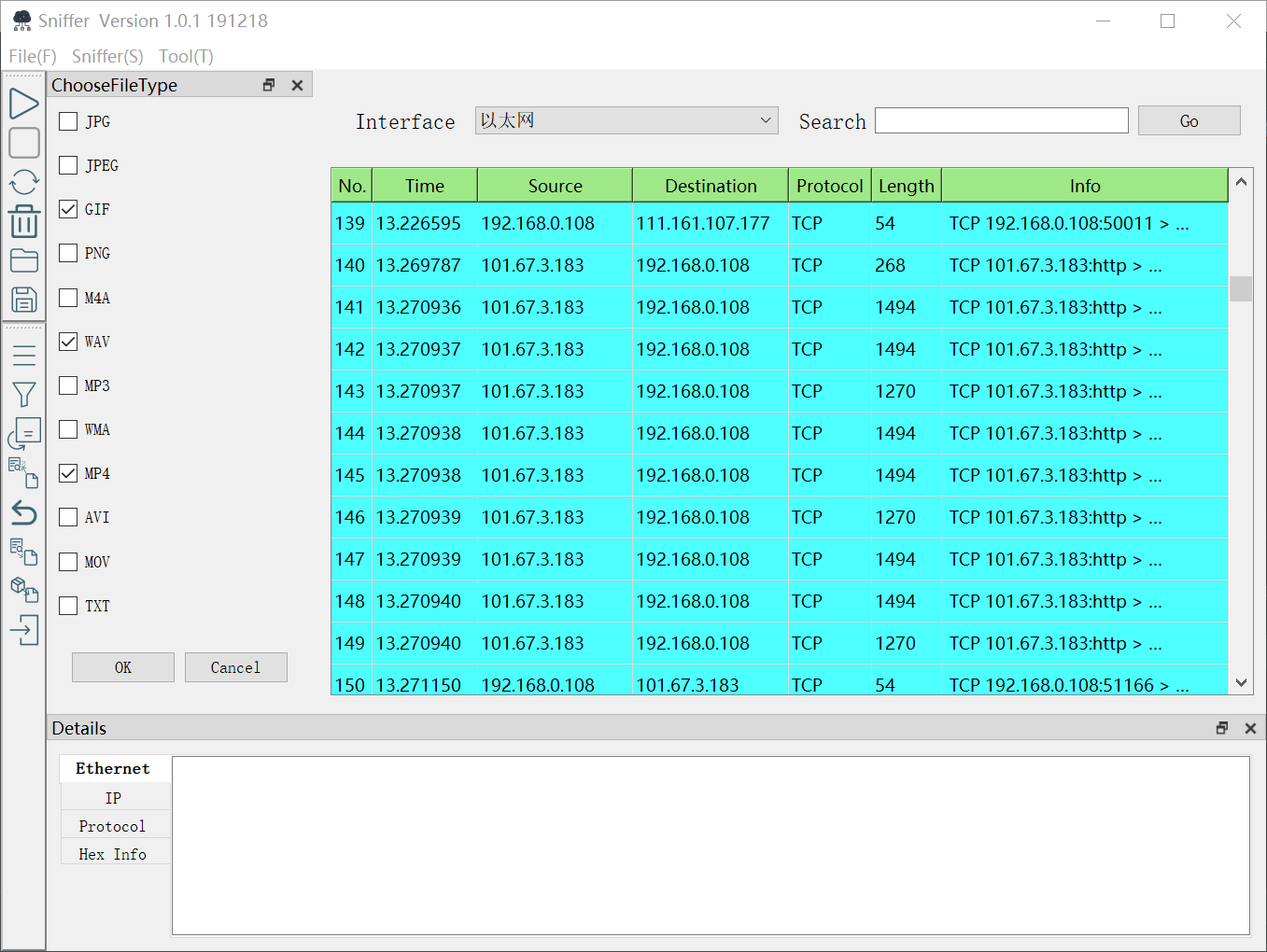


程序检测到PNG文件，并将传递文件参数的数据包以及包含文件数据的数据包显示在PktTable中，我们可以通过文件重组的两种方式之一进行测试，这里使用较为简便的第一种方式，下面给出过滤之后的显示信息与使用文件重组功能重组出testPNG.PNG文件之后使用HashTab对其进行校验的结果

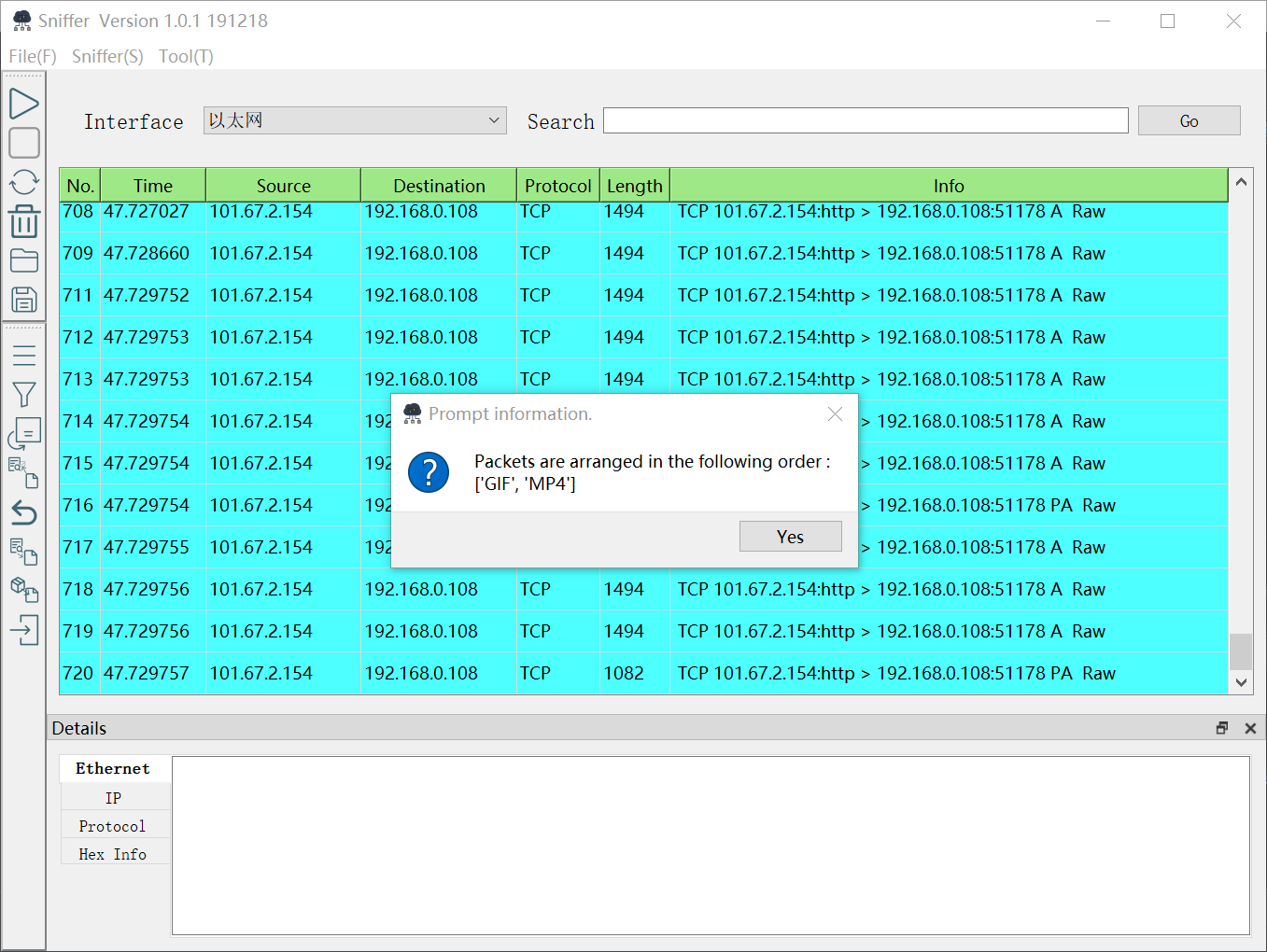




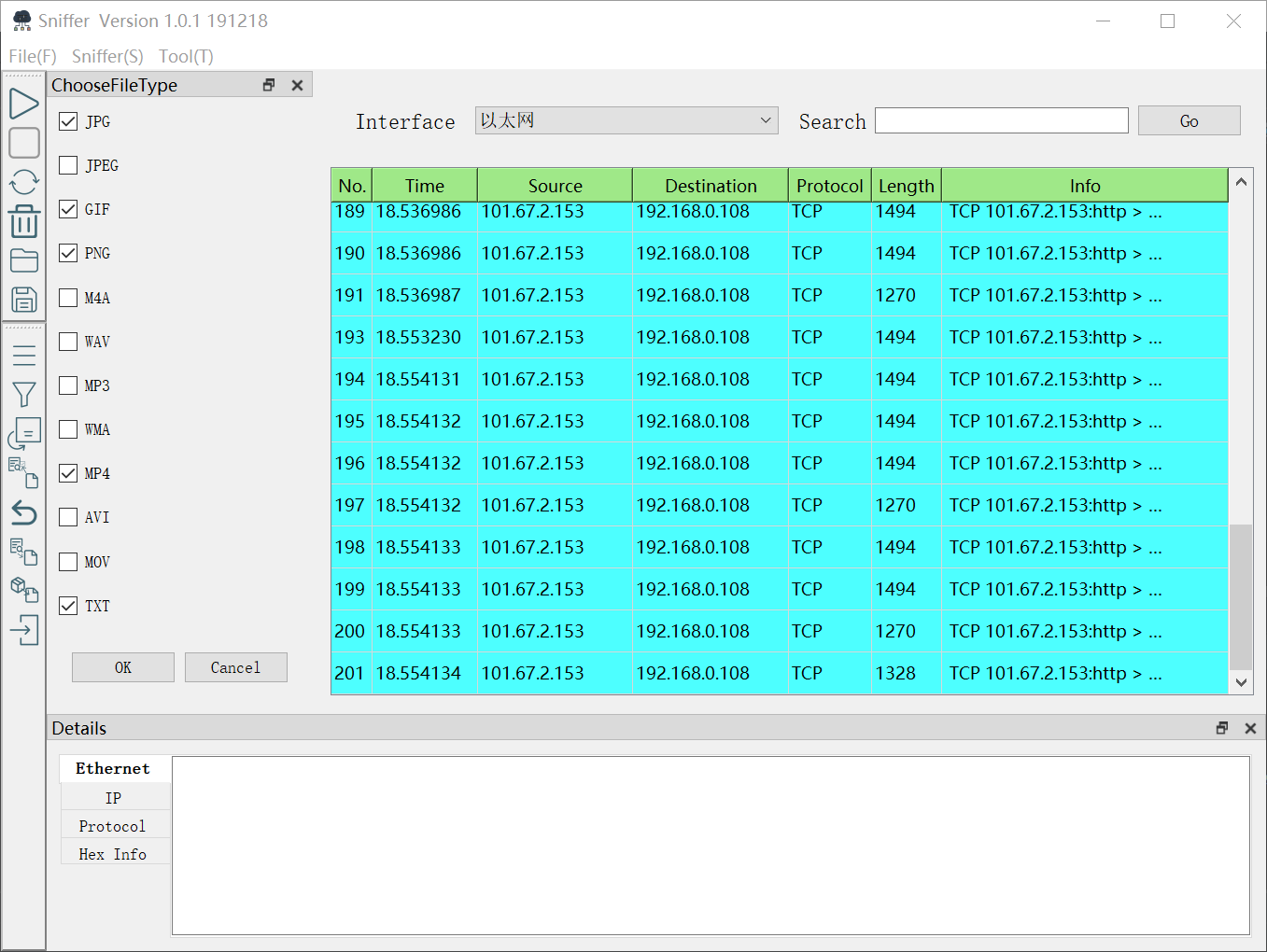
下面测试传输多个文件，使用QQ按顺序传输MP3、GIF和MP4文件，抓包，并勾选除MP3文件之外的文件类型，如下

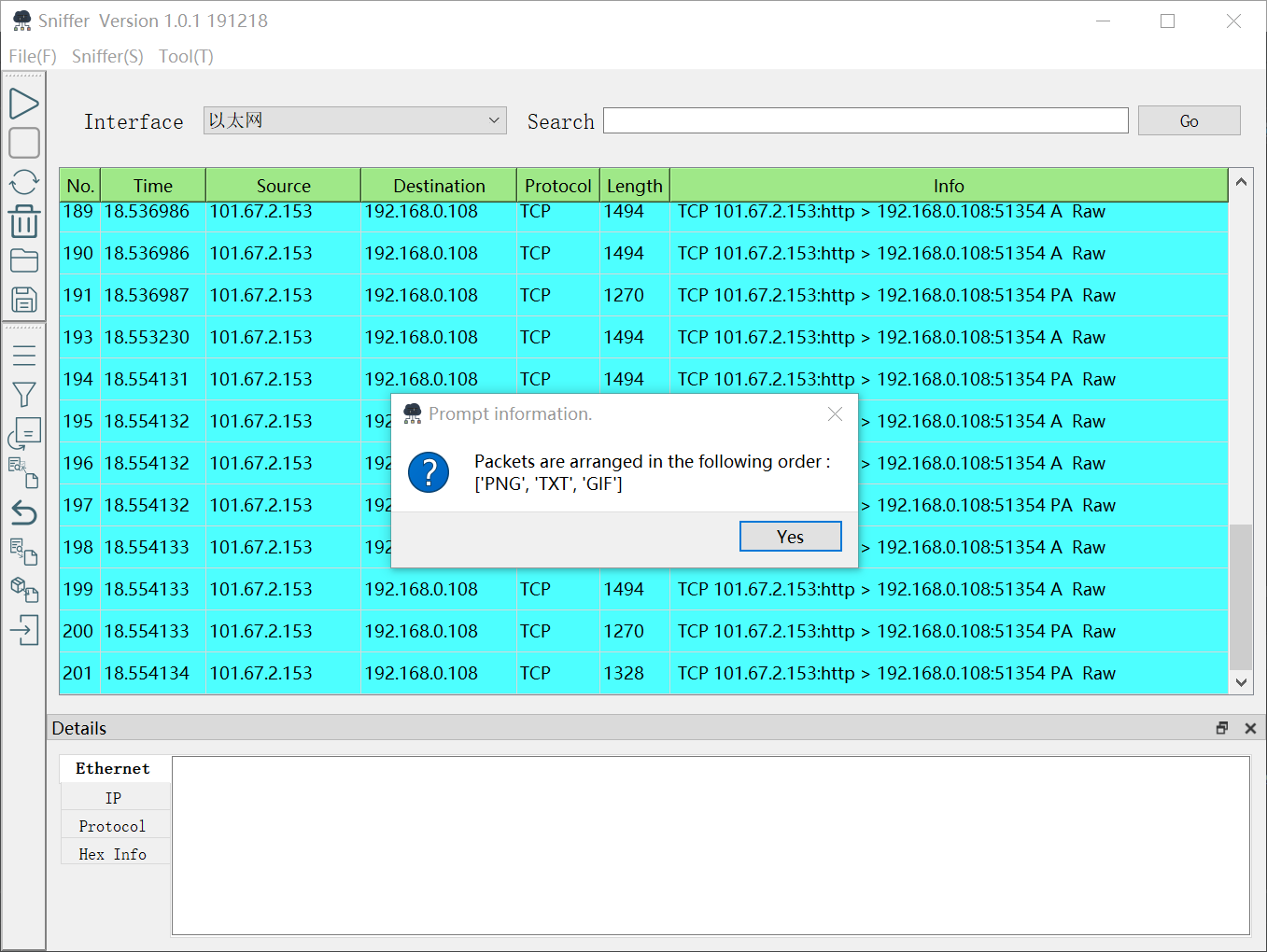


过滤结果如下，程序不仅过滤出了数据包，还按照顺序将过滤出的文件类型显示出来，便于用户之后进行文件重组，经测试，使用这些数据包可以恢复原文件



当然，本程序也可正确过滤出文本文档，我们按顺序传输PNG文件、TXT文件、GIF文件，勾选过滤文件类型，测试结果如下





**五.遇到的问题与解决方法**

**1.选择接口时，出现一些无效的网卡（多数为隧道），无法监听；**

**解决办法：**WireShark中不显示这些隧道，观察发现它们没有MAC地址，于是通过遍历得到的网卡列表，筛去没有MAC地址的网卡。

**2.Filter过滤规则问题；**

**解决办法：**一开始写了一个比较用户友好的过滤器模块，但查资料发现scapy使用BPF语法，如果使用原模块，转成BPF非常麻烦，因为时间有限，我们决定使用BPF语法规范Filter输入。

**3.显示重组之后的数据包信息时会产生卡顿，无法进行下一步操作，但是不报错**

**解决办法：**首先，我们发现此问题可以通过在查看重组报信息时将ReasultTable拖至主界面外解决，这样不会出现卡顿。后来我们发现此问题并不广泛存在，在我们两个人的电脑上运行时，一个会卡顿，一个不会卡顿，我们认为这可能是由于硬件的差异，为了适应更多类型的运行环境，我们将ReassembleWidget单独做成了一个界面。

**4.文件重组与过滤无从下手**

**解决办法：**由于文件传输是一件非常复杂的事情，所以我们一开始对通过数据包重组文件与过滤文件并没有一点思路。后来，我们在在特定环境下实现了这两个功能，其中文件重组的两种方式适应环境不同，a方式在以二进制方式传输数据的情况下都适用，b方式在QQ中适用，在其他环境中不保证一定成功，文件过滤在QQ中使用，在其他环境中不保证一定成功，事实上，文件重组b方式与文件过滤的适应环境相同。通过在特定环境下进行文件重组与过滤。

**5.尝试QProxyStyle重载更改QTabBar格式失败**

**解决办法：**现在看来可能是因为继承类虚函数调用缺参数导致，由于PyQt该类的源码只是调用的接口，我们一时找不到原因，而几个虚函数重载后总显示参数Widget为None无法使用。最后由于时间原因，重载了QTabBar继承类的paintEvent函数，算是比较简单有效的解决办法。

**六.心得体会**

这次的大作业总体来说还是很有难度的，无论是使用PyQt5设计友好的用户界面，还是用Scapy抓包解析包，刚开始时每做一步都要查很长时间的资料，也因此在开始费了很长时间，但是到最后完成所有功能时，看到自己花费了很多精力做出来的东西达到了预期效果，还是有一定的成就感的。

在做完整个项目之后，我们发现这次大作业对于我们理解计算机通信网络的内容有很大帮助，无论是计算机通信的过程还是各种数据包的结构，在实践中我们进一步加深了印象。在逐步完成GUI和将模块一步步串联起来的过程中，我们的速度越来越快，进展越来越顺利。从一开始的到处找资料，到后期对着Qt文档和Python的help函数输出，自己研究使用内置函数和重载基类函数，我们对于PyQt的认识提高了一个层次。而对于Scapy抓包，我们通过实践能够找到理论对应的部分，大作业给予我们的是具象化的认识。尝试完成文件重组等功能，要求我们深入了解报文各部分，知道在哪和怎么使用。

我们有很多想法，关于GUI的架构和内容（重做架构之后随着代码的增加，随着思考设计，我们希望还能进行1次重构，一些功能写得更加人性化一点，增加提示），关于Sniffer本身功能的强化（Info字段的实现现在还是比较简单，希望实现WireShark的那种信息），关于更改阻塞线程机制的问题。在有限的时间内，我们并不能把这些实现，一些实现也很粗糙。期待以后的空闲时间，我们能做一些额外的工作，这也是对我们编程能力的一种锻炼。

最后，感谢老师在课堂上讲了关于计算机通信网络的许多知识，为我们理解Sniffer做了理论基础，也要感谢助教学长和我们讲解了一堂课的大作业，这也为我们指明了方向，避免了很多无用功。