简易网络嗅探器的实现

1. 实验目标

网络嗅探器是一种捕捉网络数据的工具，它可以捕捉所有流经本地网卡的数据包。此外，它还可以对捕捉到的数据包进行解析，即分析数据包的通信协议，解码并获得包中的原始信息。

Wireshark 是全世界最常用的嗅探器工具之一。Wireshark 可以显示数据包列表和数据包使用的协议中每一个字段的内容。本实验的目标是参考 Wireshark，实现一个有基本功能的网络嗅探器，完成对网络数据包的捕捉和解析功能。除了基本功能外，本实验还要求实现 GUI 界面。

2. 实验原理

在讨论嗅探器的具体实现之前，我们首先应当了解数据包在互联网上的传播方式。数据包在互联网上是根据 IP 地址进行寻址的。在数据包被转发到目标 IP 地址所在的子网后，子网内的路由器会将这个包进行广播，所以子网内的所有机器均可以收到这个包。除 IP 地址外，数据包中还包含了目标结点的 MAC 地址。一般来说，计算机中的网卡默认处在直接模式，此模式下，网卡会查看数据帧中的目的 MAC 地址，若与自己的不匹配，则丢弃。而网络嗅探器则会将网卡置于混杂模式，此时网卡会将所有的数据包都保存下来，从而达到嗅探的目的。

3. 实验环境

本实验中，GUI 和网络抓包逻辑均适用 Python 语言实现。本来考虑过用 JAVA 实现 GUI 设计，但因为我没有混合编程经验，担心上手比较慢，所以整个项目全部用 Python 实现了。

环境配置：macOS + Python 3.10 + PyQt5 + scapy

3.1 图形界面 GUI

使用 Python 语言实现。主要借助 PyQt 库+Pycharm。

可以完成 GUI 设计的 Python 库比较多，综合考虑下来，PyQt 库使用最方便。PyQt 不仅有强大的功能和内置函数，本身就拥有视化界面，可以通过鼠标操作来替代一些写代码的工作。

以下是安装和使用中的经验和技巧：

（1） 安装

```
$ pip install pyqt5
$ pip install pyqt5-tools
```

注意，由于从 Python3.5 开始，Qt Designer 从 PyQt5 中转移到了 PyQt5-tools 中，因此需要单独安装 pyqt5-tools！

这一点在很多较老的教程里都没有提到，会导致后面 pyuic 无法正常使用。

（2） 添加到 PyCharm 中

在设计图形化页面时，我们主要用到的是 PyQt5 中的两个工具：Qt Designer

和 PyUIC5。Qt Designer 的功能是，将用户绘制的图形页面自动转化为.ui 文件。pyuic5 的功能是，将上一步生成的.ui 文件自动转化为.py 文件。

为了方便使用，我们可以将这两个工具添加到 PyCharm 中的 External Tools 中。这一步在网上有很多清晰的教程，在此不赘述。

（3） 函数接口

在设计页面时，我们用到的是 Qt Designer 的窗口/组件模式，但在使用内置的功能函数时，我们需要切换到信号/槽模式。和窗口模式类似，在此模式下，只需要用鼠标连线，即可实现组件之间的函数调用。

注意，假如发现某个调用关系错误，需要在 Qt Designer 页面上修改，不可以直接在 pyuic 生成的.py 文件中修改，否则下次执行 pyuic 时，会将此次修改覆盖。

使用 PyQt5 生成的文件为 mainpage.py，只要在 main.py 中 import mainpage，即可给页面上的按钮设计自定义的处理函数。同上，调用 mainpage 组件接口的程序语句务必写在 main.py 文件中。

3.2网络抓包与解析

使用 Python 语言实现。主要借助 Scapy 库。Scapy 库的功能非常强大，它允许用户发送、嗅探、分析和伪造网络包。在本实验中，主要是用其中的嗅探和分析功能。

最初我尝试使用 python libpcap 库，因为这个库底层绑定了 c 语言的 libpcap 库，所以抓包效率非常高（参考博客）。但由于 libpcap 更适用于 c 语言，有关 python 实现的参考资料非常有限，且对数据包解析和过滤方面有所欠缺。综上考虑，最后改用了更流行的 Scapy 库。

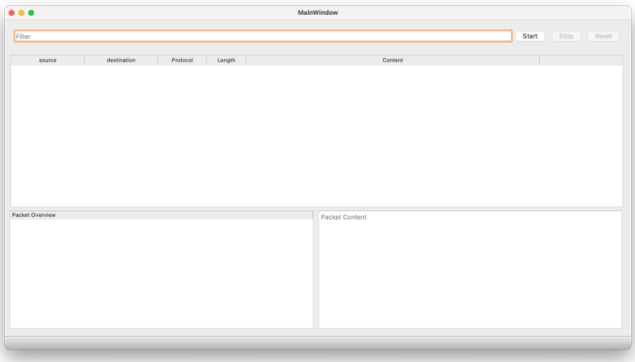
Scapy 库的使用指南非常多，因此不再一一列举各函数的用法。但在使用过程中曾遇到报错“找不到/dev/bpf*文件”。此问题可能是 macOS 系统独有，且较难复现，所以相关资料较少，花了很长时间才查到解决方法。此问题可能与访问权限有关。解决方法：

```
$ sudo chgrp admin /dev/bpf*
$ sudo chmod g+rw /dev/bpf*
```

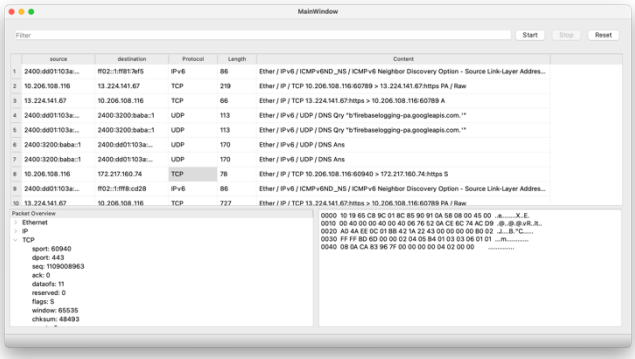
此方法来源为此篇博客。

4. 实现效果

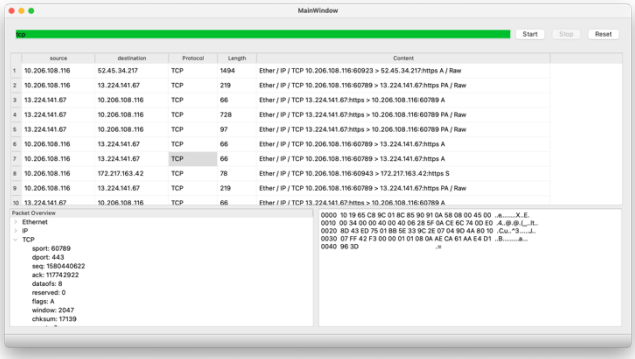
页面布局：



开始嗅探：



添加过滤条件的嗅探：



5. 思考与总结

在本实验中，我实现了一个拥有基础功能的网络嗅探器，可以完成网络数据包的解析和捕获。本项目后续的优化方向为：

1. 交换式局域网的监听（考虑使用 ARP 欺骗）；
2. 支持选择网口；
3. 提升软件稳定性。

6. 参考

除上述提到的两篇博客外，还需感谢 ShenTiao。