电子科技大学信息与软件工程学院

**计 算 机 网 络**

**课程设计报告**

学 号 2017221302009

姓 名 陆圣珩

**电子科技大学教务处制表**

目 录

[1. 课程设计任务及特色 3](#_Toc11350008)

[2. 本课程设计包含的网络知识点 4](#_Toc11350009)

[3. 相关软件 6](#_Toc11350010)

[4. 任务分解及设计(重点) 6](#_Toc11350011)

[5. 课程设计成果展示 9](#_Toc11350012)

[6. 总结及互评 12](#_Toc11350013)

## 课程设计任务及特色

本次课程设计的基本任务为完成本机上的流量检测任务，对用户上网行为进行监控，特别记录用户访问淘宝、京东和进行QQ聊天的日志，统计用户每天、每时段访问淘宝的次数、访问京东的次数、进行QQ聊天的时间，给出图形化的统计结果。

课程设计主要分为三个部分，即后台监测系统的搭建，前端数据可视化展示系统的搭建，以及监测算法的部署。

后端选择 python 作为主要语言，使用 python-flask 作为后端框架，选择 flask 的理由是其轻巧的后端架构，快速的开发流程，与算法的较好的契合度，以及对前后端交互使用 RESTful 的支持。

前端使用 React.js 这一主流前端框架，并使用目前社区建设较为完备的 Echarts 作为数据可视化的载体。这一选择是考虑到 React.js 相比其他框架有着更加成熟的社区支持，以及更加规范化的组件模式，方便进一步添加新的组件以丰富课程设计。Echarts 作为前端最为著名的几个数据可视化框架在几年内已经得到了业界普遍的认可，并拥有一份完整详实并容易上手的官方文档以供使用。

前端部分设计了一个用户管理系统，即只有经过注册授权的电脑用户方可访问检测系统前端，并进行规则的添加、监控网站的添加等操作。以此相对应的，后端将会使用 flask-login 完成一个简单的注册登录方法，并使用 Mysql 完成基础的数据库搭建。

核心算法部分是基于 Scapy 库进行开发，在经过学习，使用，并针对课程设计进行修改与封装之后部署于后端。Scapy是一个功能强大的交互式数据包操作程序。它能够伪造或解码大量协议的数据包，通过线路发送，捕获它们，匹配请求和回复等等。

本次课程的难点在于流量监测部分，即利用 Scapy 记录访问日志，统计访问特定网站的次数，并将获得的数据发送给前端进行展示。

## 本课程设计包含的网络知识点

HTTP简介

HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。

HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于1990年提出，经过几年的使用与发展，得到不断地完善和扩展。目前在WWW中使用的是HTTP/1.0的第六版，HTTP/1.1的规范化工作正在进行之中，而且HTTP-NG(Next Generation of HTTP)的建议已经提出。

HTTP协议工作于客户端-服务端架构为上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。Web服务器根据接收到的请求后，向客户端发送响应信息。

HTTP请求报文

一个HTTP请求报文由请求行（request line）、请求头部（header）、空行和请求数据4个部分组成。

请求行由请求方法字段、URL字段和HTTP协议版本字段3个字段组成，它们用空格分隔。例如，GET /index.html HTTP/1.1。HTTP协议的请求方法有GET、POST、HEAD、PUT、DELETE、OPTIONS、TRACE、CONNECT。

请求头部由关键字/值对组成，每行一对，关键字和值用英文冒号“:”分隔。请求头部通知服务器有关于客户端请求的信息，典型的请求头有：User-Agent：产生请求的浏览器类型。Accept：客户端可识别的内容类型列表。Host：请求的主机名，允许多个域名同处一个IP地址，即虚拟主机。

最后一个请求头之后是一个空行，发送回车符和换行符，通知服务器以下不再有请求头。

  请求数据不在GET方法中使用，而是在POST方法中使用。POST方法适用于需要客户填写表单的场合。

## 相关软件

装有 Windows 10 操作系统的个人计算机，用于存放并部署前后端代码并运行；

Python 3.7

Scapy

JavaScript

React

Echarts

## 任务分解及设计(重点)

任务1：后端 python-flask 框架搭设

任务2：前端 React 框架搭设，Echarts 部署

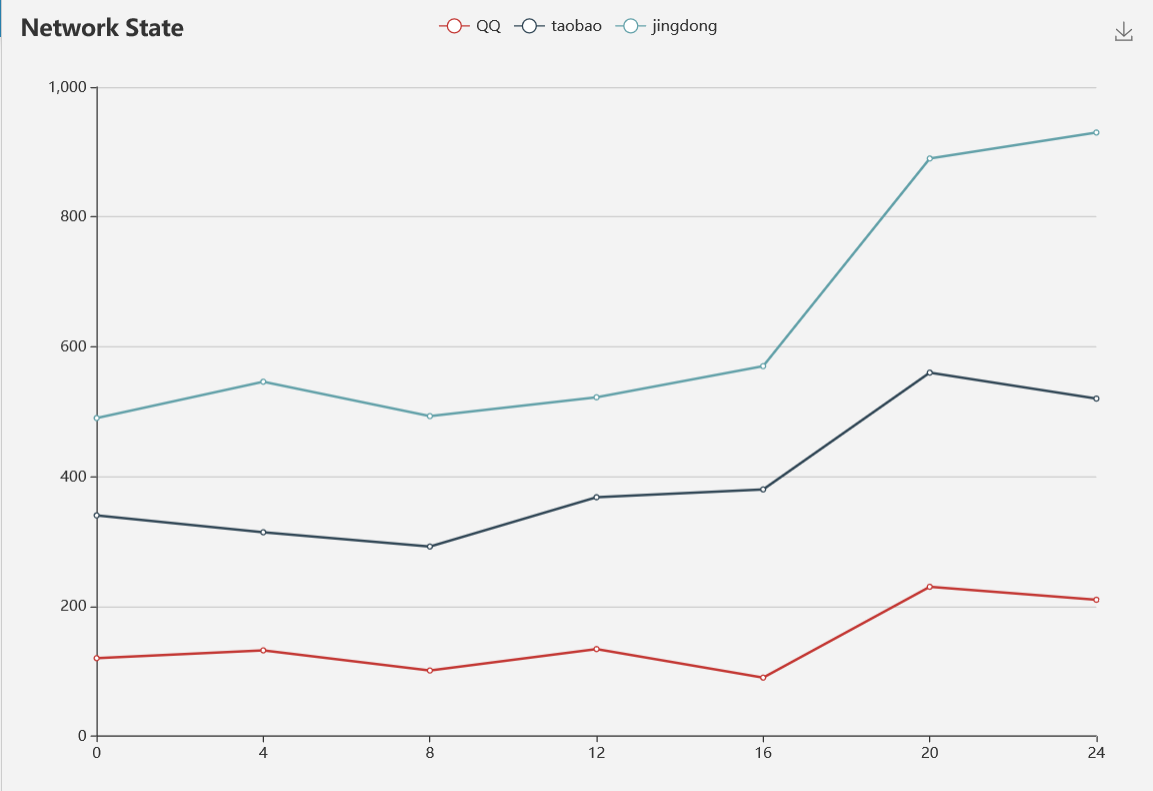
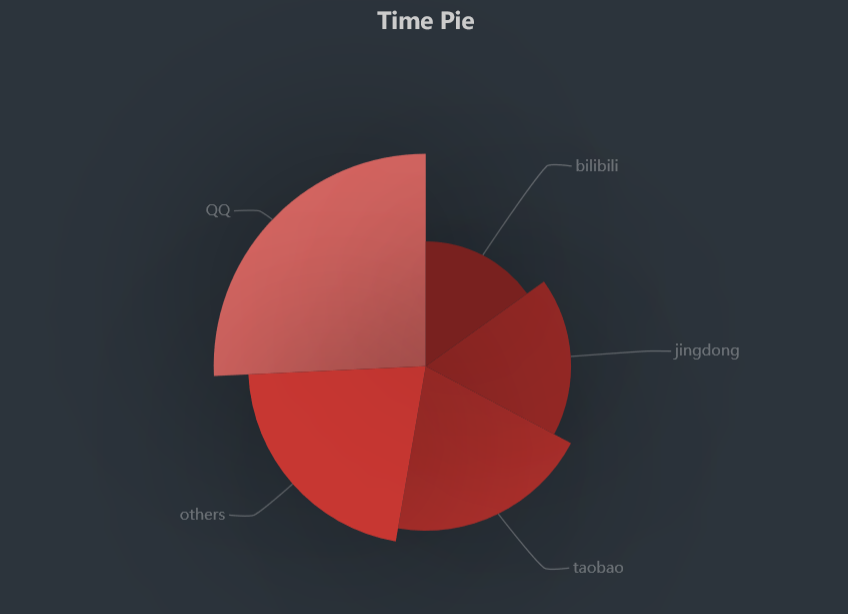
任务3：流量监测算法的实现与部署

**整体设计流程：**

首先完成基础算法的实现，根据算法输入输出设计前后端端口以供数据交互，并设计数据库用以储存算法产生的数据。在完成端口设计后基于需求分析完成前后端框架设计，并进行开发，调试，部署。

**统计数据可视化：**

使用 Echarts 前端框架完成数据可视化，主要的数据可视化内容分为两块，一是用户进行操作的时间占比，二是总报文数的时刻统计图，计划使用饼图完成第一个的数据可视化，利用折线图完成第二个目标的数据可视化，样式如下：

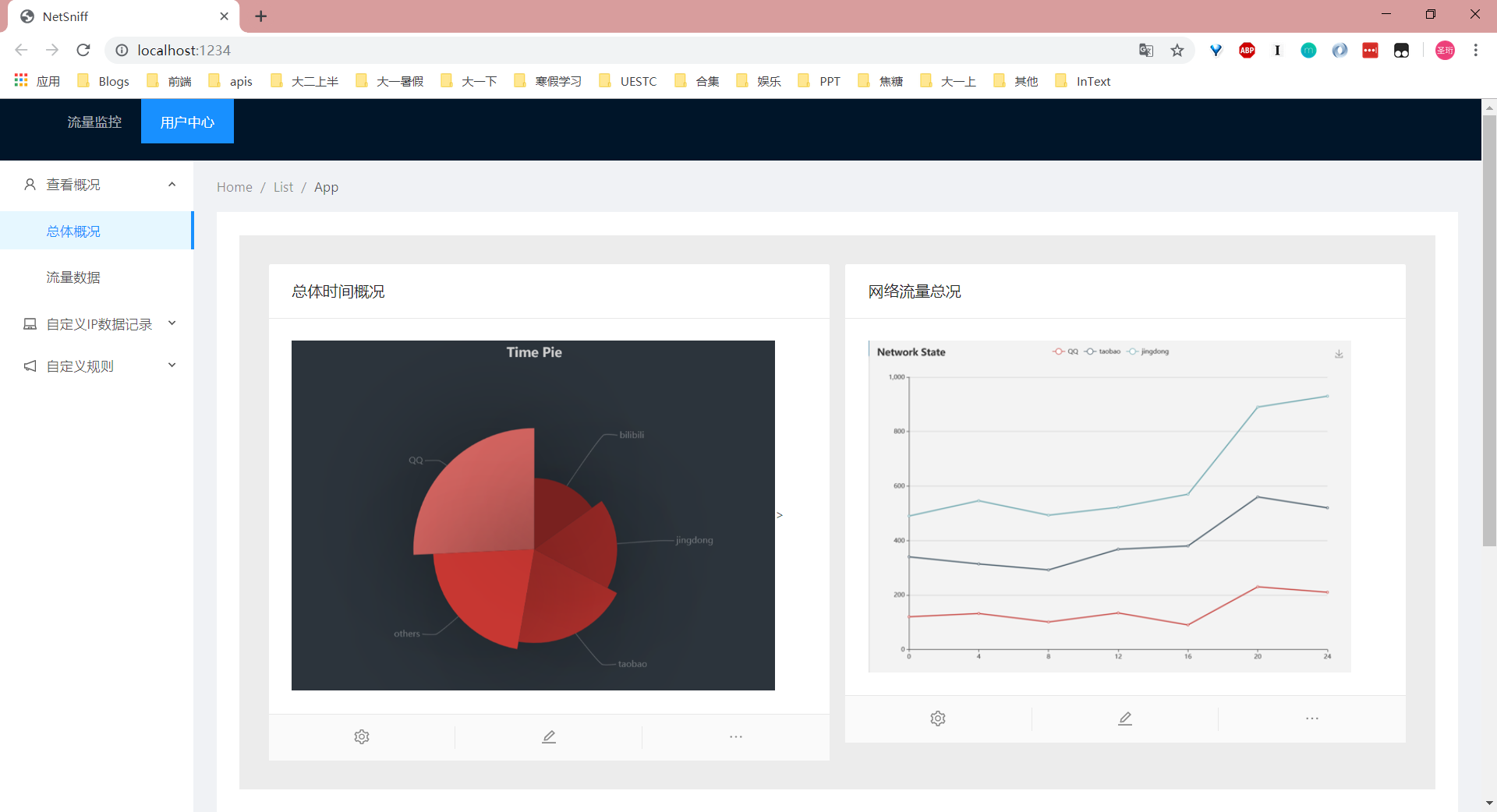
注：此数据为随机数据

**前端页面设计：**

利用Antd 经典 顶部-侧边布局-通栏 这一布局，右边显示数据可视化，左边为选项选择，加入搜索框以检索数据。

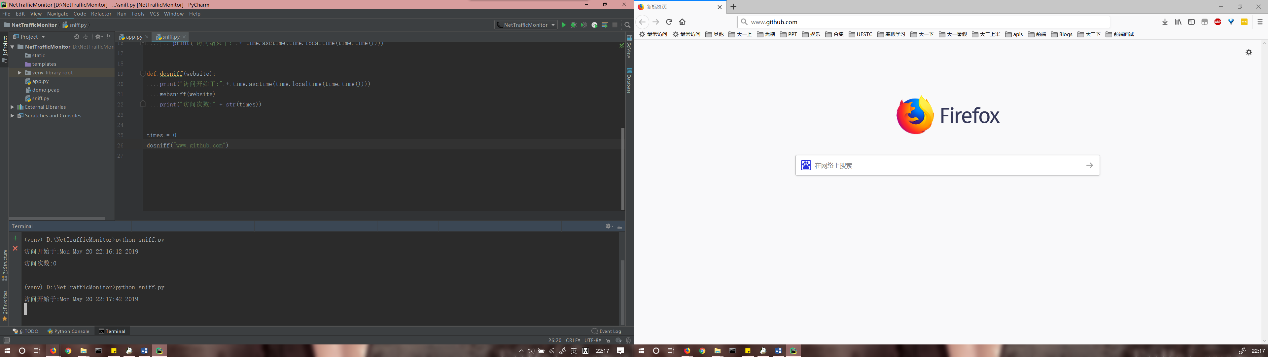
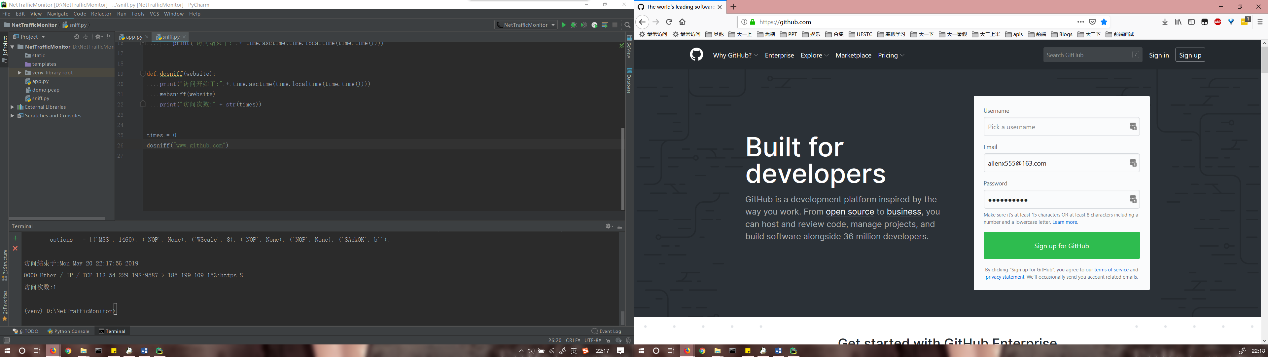
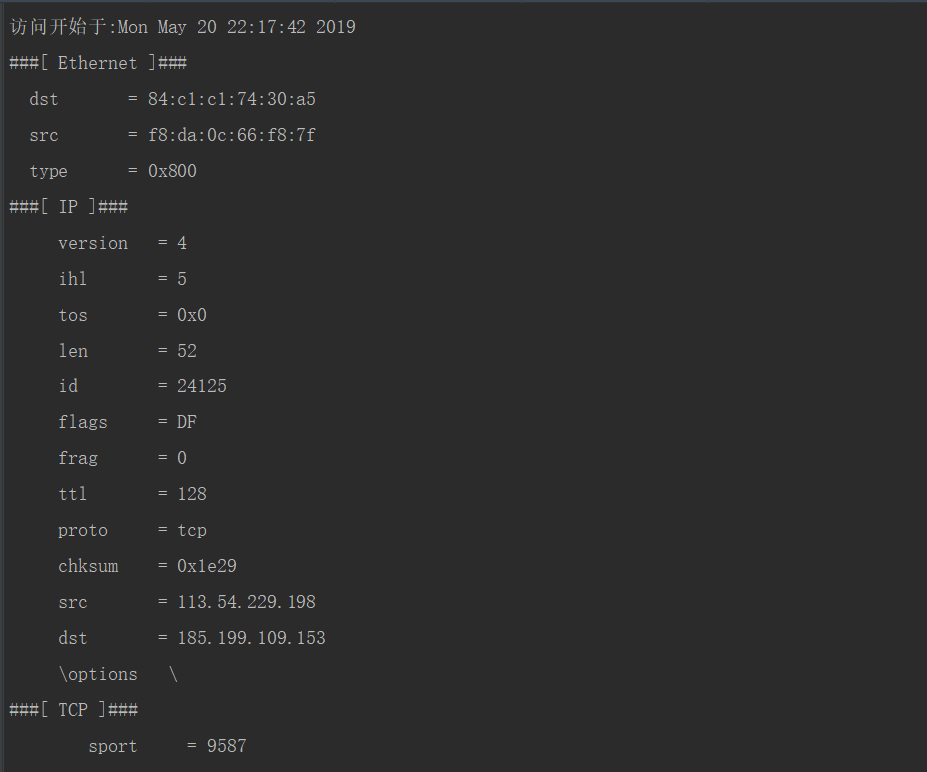
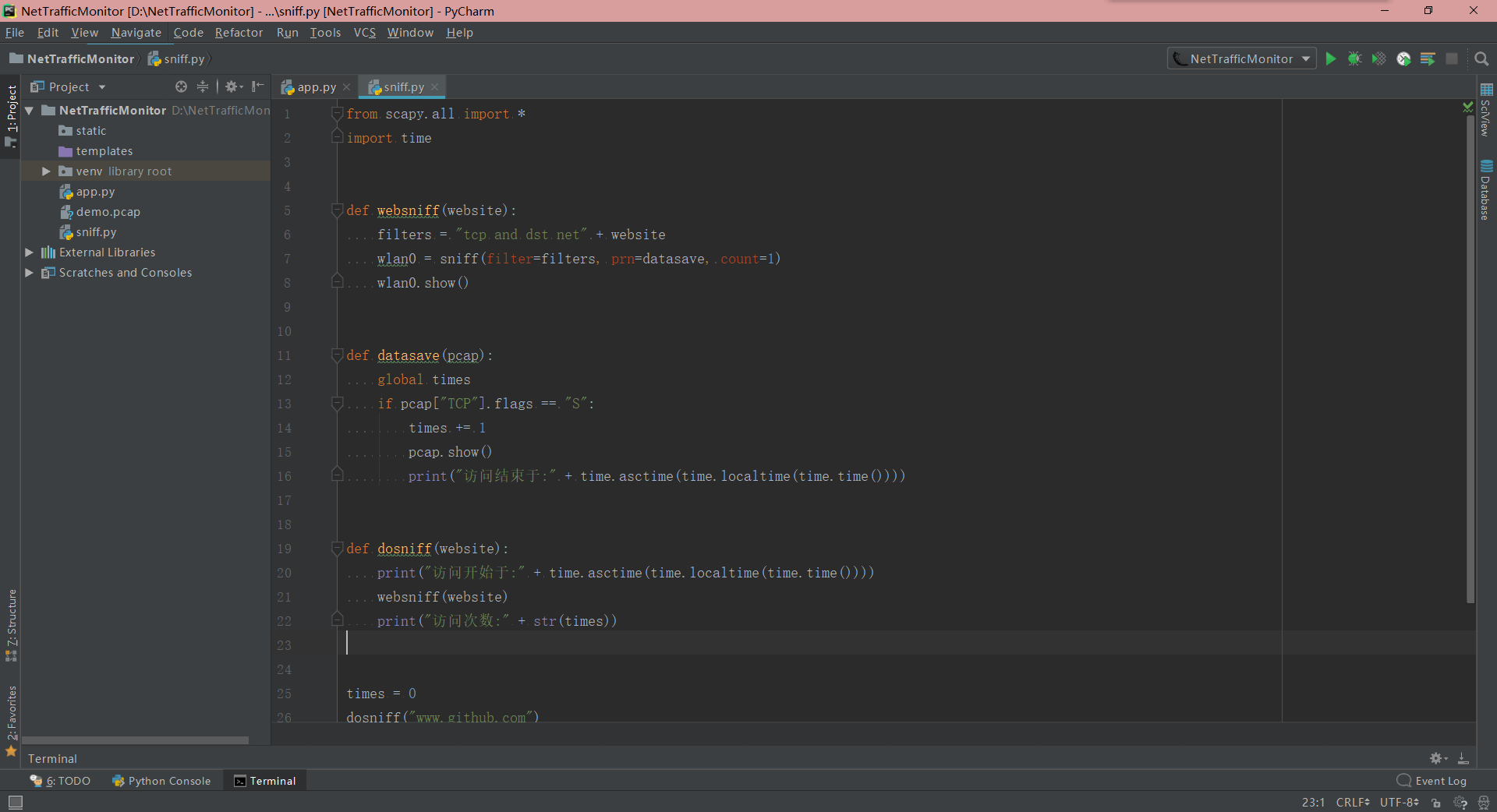
顶部为总应用切换，分别对应主功能流量监控界面与辅助功能用户中心，用户中心存在的目的是为了诸如家长监护或防沉迷等后续计划。

主流量监控页面左侧选项栏如图所示，其中自定义规则为添加新网络以供监控，自定义IP数据记录栏为已添加的网站的数据流量显示。

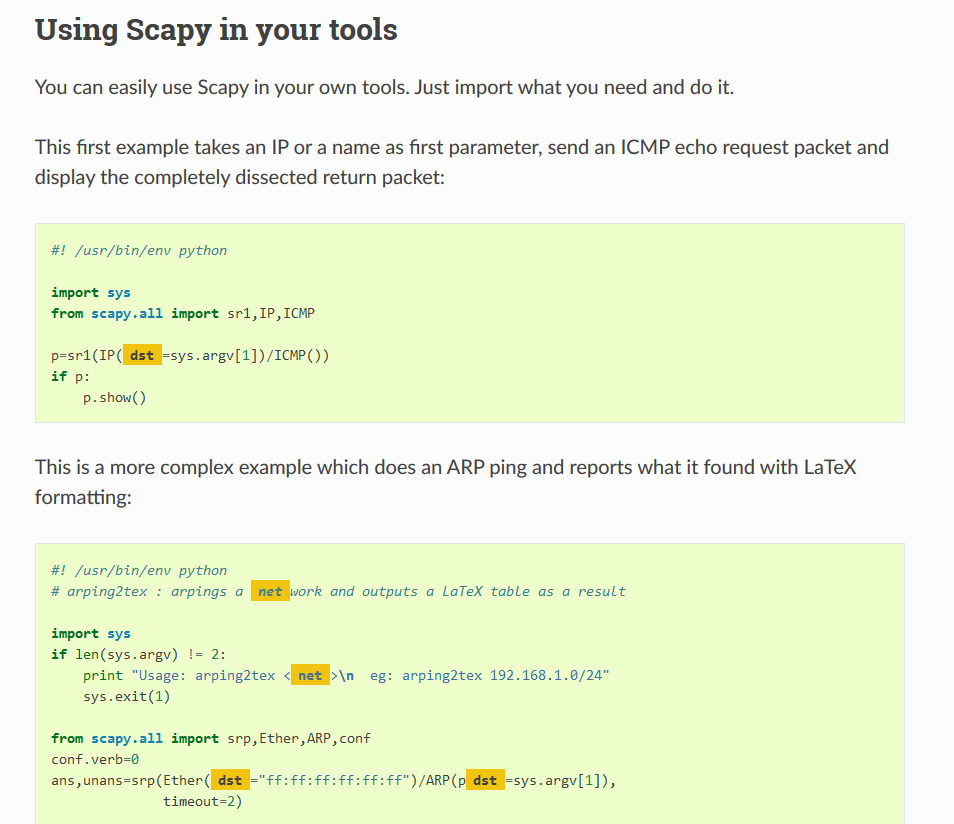


## 课程设计成果展示

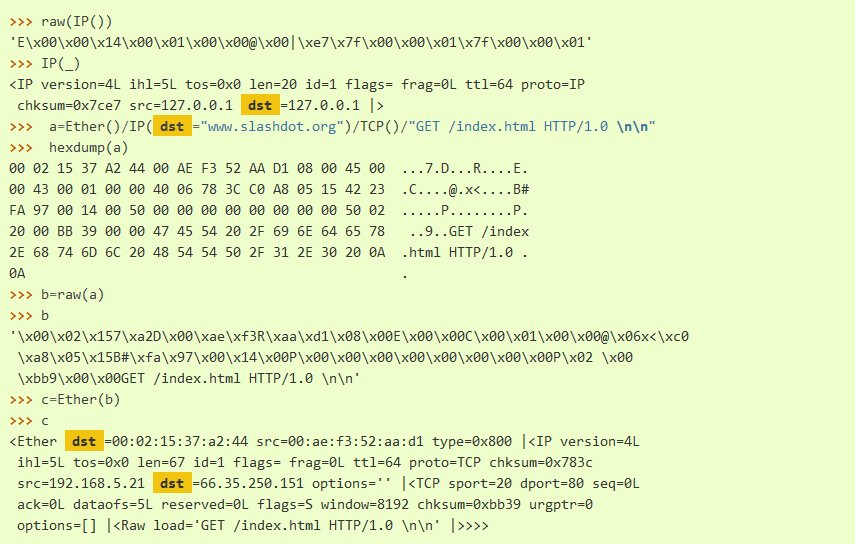
|  |
| --- |
| 核心算法模块代码： |
| from scapy.all import \*  import time  def websniff(website):  filters = "tcp and dst net" + website  wlan0 = sniff(filter=filters, prn=datasave, count=1)  wlan0.show()  def datasave(pcap):  global times  if pcap["TCP"].flags == "S":  times += 1  pcap.show()  print("访问结束于:" + time.asctime(time.localtime(time.time())))  def dosniff(website):  print("访问开始于:" + time.asctime(time.localtime(time.time())))  websniff(website)  print("访问次数:" + str(times))  times = 0  dosniff("www.github.com") |
| 注释：  通过检索 TCP 报文 flags 字段以确认是否创建了新的链接访问，记录时间，并使全局属性 times 增长。 |

其中核心代码 filters = "tcp and dst net" + website 来自 scapy 自带的函数，根据阅读 scapy 官方文档与其源代码，可以追踪到其使用了一个 IP（）函数，如图所示：



而 IP（）为 scapy 自定函数，找到了其示例与一部分实现原理：



可以确定其也是使用了 IP 协议以将网址转换为具体 ip 地址。

## 总结及互评

本次课程设计由本人独立完成，在时间安排上存在一定的问题，没有考虑到过大的工作量导致时间过于紧张。

利用 Scapy 解析消息头，以及如何判断是否在一定时间内建立了新的连接在一开始困扰了一段时间，但是在回头仔细阅读官方文档，并重新审视各个 flags 之后最终想出了这个解决方法。

因为时间和人力的紧张导致 qq 部分只能够实现最基本的抓包，无法更进一步分析其内容。

通过本次学习再一次复习了HTTP协议，UDP和TCP协议等，实践是最好的学习知识的方法，通过这次实验能够对原本稍显模糊的计网的知识有了更加清晰具体的视角，并有了自己的心得体会和经验。