电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

课程设计报告

课设题目	基于 Scapy 和 PyEcharts 的可视化网络嗅探监督系统
学　　号	2017221102011/2017221102021
姓　　名	程允锐/彭子为
授课教师	刘梦娟

目 录

1、课程设计任务及特色

本报告属于《计算机网络基础》课程的综合课程设计。课题要求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、每时段访问淘宝网站的次数（时间）、访问京东网站的次数（时间）、进行 QQ 聊天的时间，给出图形化的统计结果。

本课程设计完成了基于 Python 环境下的对应功能脚本。针对不同类型的网站（Bilibili，CSDN，京东，淘宝，腾讯 QQ 等）的访问，统计生成了一个网页报告，其中包含了**报文活跃柱状图，报文请求-时间柱状图，报文总数-时间折线关系**以及抓获所有**包的类型分类**。我们对所有报文的类型（['TCP', 'UDP', 'ICMP', 'Other']、['IP', 'IPv6']）都进行了统计，可以参见结果演示，以饼状图的形式给出。这些数据由 Scapy 提供的直接返回结果统计而来。

我们根据抓获报文的通信 IP 地址来确定该报文是否是上述网站的通信报文。因为 IP 地址可能动态变化，我们根据多次访问获取的 IP 地址做了一个映射（在 Python 中使用的字典），将已经确定的 IP 地址映射到相应域名中。另外，对于部分服务可能使用连续的 IP 地址段，将该 IP 地址段加入映射。具体操作可见代码。在实际实践中，我们发现根据数据包的数量来判断一个人的访问习惯是极不准确的。在视频网站和电子商务网站中，持续的浏览会产生大量的报文；而相比之下，CSDN 之类的博客网站以文字居多，数据量小，对应的报文数量也小了很多。如果根据报文数量来定义用户正在进行的浏览目标的确定很不严谨。所以我们采用了根据时间抽样的方法来进行统计。

基于上述考虑，我们设计了如下参数：Round（采集报文的轮数）、Cnt（每轮采集的报文数）、breaktime（每轮采集的间隔）。在程序运行的过程中，每一轮之间会中断 breaktime 时长，用于保证处理机的性能。在 breaktime 之后，采集报文 Cnt 个，如果其中包含了映射中存在的地址的报文，那么认为当前时间用户正在访问该网站。为了减小误差，我们设置较大参数，令 Round=300，Cnt=1000，breaktime=500ms。

本表获得的统计结果中的各个图示的含义如下：

- 报文活跃柱状图：抽象时间内的报文活跃数，指多次抽样中各个网站的访问次数
- 报文请求-时间柱状图：单次抓包后的各个网站的报文数量
- 报文总数-时间折线图：抓包累计的各个网站的报文数量
- 抓获所有包的类型分类：['TCP', 'UDP', 'ICMP', 'Other']、['IP', 'IPv6']。

在控制界面中，我们使用了 Tkinter 库，提供了对已有分类规则以及各项参数的查看和修改操作。通过加入新规则，我们可以检测和嗅探远远不限于以上五个网站的报文。

2、本课程设计包含的网络知识点

以太网（Ethernet）具有共享介质的特征，信息是以明文的形式在网络上传输，当网络适配器设置为监听模式（混杂模式，Promiscuous）时，由于采用以太网广播信道争用的方式，使得监听系统与正常通信的网络能够并联连接，并可以捕获任何一个在同一冲突域上传输的数据包。IEEE802.3 标准的以太网采用的是持续 CSMA 的方式，正是由于以太网采用这种广播信道争用的方式，使得各个站点可以获得其他站点发送的数据。运用这一原理使信息捕获系统能够拦截的我们所要的信息，这是捕获数据包的物理基础。

以太网是一种总线型的网络，从逻辑上来看是由一条总线和多个连接在总线上的站点所组成各个站点采用上面提到的 CSMA/CD 协议进行信道的争用和共享。每个站点（这里特指计算机通过的接口卡）网卡来实现这种功能。网卡主要的工作是完成对于总线当前状态的探测，确定是否进行数据的传送，判断每个物理数据帧目的地是否为本站地址，如果不匹配，则说明不是发送到本站的而将它丢弃。如果是的话，接收该数据帧，进行物理数据帧的 CRC 校验，然后将数据帧提交给 LLC 子层。

本课程设计使用了 Winpcap 和 Scapy 对数据包进行抓取，使用了 PyEcharts 进行绘图与展示。Winpcap 是由伯克利分组捕获库派生而来的分组捕获库，它是在 Windows 操作平台上来实现对底层包的截取过滤。Winpcap 是 BPF 模型和 Libpcap 函数库在 Windows 平台进行**网络数据包捕获和网络状态分析**的一种体系结构，这个体系结构是由一个核心的包过滤驱动程序，一个底层的动态连接库 Packet.dll 和一个高层的独立于系统的函数库 Libpcap 组成。底层的包捕获驱动程序实际为一个协议网络驱动程序，通过对 NDIS 中函数的调用为 Win95、Win98、WinNT、和 Win2000 提供类似于 UNIX 系统下 Berkeley Packet Filter 的捕获和发送原始数据包的能力。Packet.dll 是对这个 BPF 驱动程序进行访问的 API 接口，同时它有一套符合 Libpcap 接口（UNIX 下的捕获函数库）的函数库。

Winpcap 包括三个部分：第一个模块 NPF(Netgroup Packet Filter)，是一个虚拟设备驱动程序文件。它的功能是过滤数据包，并把这些数据包原封不动地传给用户态模块，这个过程中包括了一些操作系统特有的代码。第二个模块 packet.dll 为 win32 平台提供了一个公共的接口。不同版本的 Windows 系统都有自己的内

核模块和用户层模块。Packet.dll 用于解决这些不同。调用 Packet.dll 的程序可以运行 在不同版本的 Windows 平台上，而无需重新编译。

图 1 Scapy

Scapy 是一个 Python 程序，使用户能够发送，嗅探和剖析并伪造网络数据包。此功能允许构建可以探测，扫描或攻击网络的工具。本次课程设计的 Scapy 基于 Winpcap 运行。换句话说，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。Scapy 可以轻松处理大多数经典任务，如扫描，跟踪路由，探测，单元测试，攻击或网络发现。它可以取代 hping，arpspoof，arp-sk，arping，p0f 甚至是 Nmap，tcpdump 和 tshark 的某些部分。

Tkinter 是 Python 的标准 GUI 库。Python 使用 Tkinter 可以快速的创建 GUI 应用程序。简单的构造，多平台，多系统的兼容性，能让它成为让你快速入门定制窗口文件的好助手。

图 2 PyEcharts

PyEcharts 是一个用于生成 Echarts 图表的类库。Echarts 是百度开源的一个数据可视化 JS 库。用 Echarts 生成的图可视化效果非常棒，为了与 Python 进行对接，方便在 Python 中直接使用数据生成图，几位开发者写了这个项目。

3、相关软件

- Anaconda（Python 3.5）with
  - Scapy (Winpcap)
  - PyEcharts
  - Tkinter
- Any Browser

4、任务分解及设计(重点)

- 任务 1：初始化数据并进行 IP 地址的映射表构建
- 任务 2：直接的抓包操作，包内数据提取与处理，单次抓包内时间间隔
- 任务 3：统计数据可视化，绘图展示
- 任务 4：界面与跳转设计，加入额外的规则

整体设计流程：

任务一：引用，初始化数据并进行 IP 地址的映射表构建

```
from scapy.base_classes import *
from scapy.config import *
from scapy.dadict import *
from scapy.data import *
from scapy.error import *
from scapy.themes import *
from scapy.arch import *
from scapy.plist import *
from scapy.fields import *
from scapy.packet import *
from scapy.asn1fields import *
from scapy.asn1packet import *
from scapy.utils import *
from scapy.route import *
from scapy.sendrecv import *
from scapy.sessions import *
from scapy.supersocket import *
from scapy.volatile import *
from scapy.as_resolvers import *
from scapy.ansmachine import *
```

```
from scapy.automaton import *
from scapy.autorun import *
from scapy.main import *
from scapy.consts import *
from scapy.compat import raw  # noqa: F401
from scapy.layers.all import *
from scapy.asn1.asn1 import *
from scapy.asn1.ber import *
from scapy.asn1.mib import *
from scapy.pipetool import *
from scapy.scapypipes import *
import scapy_http.http as http

if conf.ipv6_enabled:  # noqa: F405
    from scapy.utils6 import *  # noqa: F401
    from scapy.route6 import *  # noqa: F401

import re
import time
from pyecharts import Pie, Page, Bar, Line
import webbrowser

v2 = [0, 0]
v1 = [0, 0, 0, 0]
record = [0, 0, 0, 0, 0]
timerecord = [[0], [0], [0], [0], [0]]
timesingle = [[0], [0], [0], [0], [0]]
timetime = [0, 0, 0, 0, 0]
tr = []
dict = {}
Round = 3
breaktime = 0.5
Cnt = 1000


# IP mapping
def make_link_ip():
    dict['58.205.217.1'] = 0  # jd
    dict['120.52.148.118'] = 0
    dict['111.231.211.246'] = 1  # bilibili
    dict['119.27.176.150'] = 1
    dict['120.24.248.50'] = 1
    dict['140.143.82.138'] = 1
```

```
    dict['111.231.212.88'] = 1
    dict['120.92.162.180'] = 1
    dict['47.95.164.112'] = 2  # CSDN
    dict['58.205.221.214'] = 3  # Taobao
    dict['58.205.221.253'] = 3
    dict['140.205.94.189'] = 3
    dict['140.205.94.193'] = 3
    dict['203.119.215.107'] = 3
    # dict['121.51']=4 Tencent QQ
    # dict['210.41']=4
     # dict['111.231']=1

print(time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time())))
tr.append(time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(time.time())))
make_link_ip()
```

任务 2：直接的抓包操作，包内数据提取与处理，单次抓包内时间间隔

```
for t in range(0, Round):

    # 嗅探抓包
    wlan = sniff(iface='WLAN', count=Cnt)
    s = str(wlan)
    print(wlan)
    print(wlan.show())
    # wrpcap('packet.cap', wlan)

    # 提取数据
    v3 = re.findall(r"\d+\.?\d*", s)
    for i in range(0, len(v3)):
        v1[i] += int(v3[i])
    for i in range(0, len(wlan)):
        try:
            if 'IPv6' in wlan[i]:
                v2[1] += 1
            else:
                v2[0] += 1
            if wlan[i].payload.dst in dict.keys():
                record[dict[wlan[i].payload.dst]] += 1
            elif wlan[i].payload.src in dict.keys():
                record[dict[wlan[i].payload.src]] += 1
            # else:
            #    record[0] += 1
```

```
                elif ('121.51' in wlan[i].payload.dst) or ('121.51' in
wlan[i].payload.src) or \
                        ('210.41' in wlan[i].payload.dst) or ('210.41' in
wlan[i].payload.src):
                    record[4] += 1
                elif ('111.231' in wlan[i].payload.dst) or ('111.231' in
wlan[i].payload.src):
                    record[1] += 1
                # print(wlan[i].show())
            except:
                pass
            # print(hexdump(p))

        # 数据处理
        for i in range(0, len(timerecord)):
            timerecord[i].append(record[i])
            timesingle[i].append(record[i] - timerecord[i][t])
            timetime[i] += min(record[i] - timesingle[i][t], 1)
        tr.append(time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())))
        print('this is the %dth round, sleeping for %f second(s).' % (t + 1,
breaktime))
        time.sleep(breaktime)

    # For Debug Use
    print(timerecord)
    print(tr)
```

任务 3：绘图与展示

```
    # 作图
    page = Page()
    attr = ['JD', 'bilibili', 'CSDN', 'Taobao', 'Tencent QQ']
    bar = Bar('报文活跃柱状图')
    bar.add('按抽样时间分类',
            attr,
            timetime,
            # is_convert=True,
            is_more_utils=True  # 设置最右侧工具栏
            )
    page.add_chart(bar)
    bar = Bar('报文请求-时间柱状图')
    for i in range(0, len(timerecord)):
        bar.add(attr[i],
```

```
            tr[1:],
            timesingle[i][1:],
            is_datazoom_show=True,
            # is_convert=True,
            is_more_utils=True  # 设置最右侧工具栏
            )
page.add_chart(bar)
line = Line("访问报文数量-时间折线图")
for i in range(0, len(timerecord)):
    line.add(
        attr[i],
        tr,
        timerecord[i],
        is_datazoom_show=True,
        is_fill=True,
        line_opacity=0.2,
        area_opacity=0.4
    )
page.add_chart(line)
pie = Pie('网络-IP 类型饼状图', title_pos='left')
attr = ['TCP', 'UDP', 'ICMP', 'Other']
pie.add(
    '', attr, v1,  # '': 图例名（不使用图例）
    radius=[50, 75],  # 环形内外圆的半径
    is_label_show=True,  # 是否显示标签
    label_text_color=None,  # 标签颜色
    legend_orient='vertical',  # 图例垂直
    legend_pos='right'
)
attr = ['IP', 'IPv6']
pie.add(
    '', attr, v2,
    radius=[15, 35],
    is_label_show=True,
    label_text_color=None,
    legend_orient='vertical',
    legend_pos='right'
)
page.add_chart(pie)

# 保存
page.render('./page.html')
```

```
    # 打开
    chromepath = 'C:\\Program Files
(x86)\\Google\\Chrome\\Application\\chrome.exe'
    webbrowser.register('chrome', None, webbrowser.BackgroundBrowser(chromepath))
    webbrowser.get('chrome').open('page.html')
```

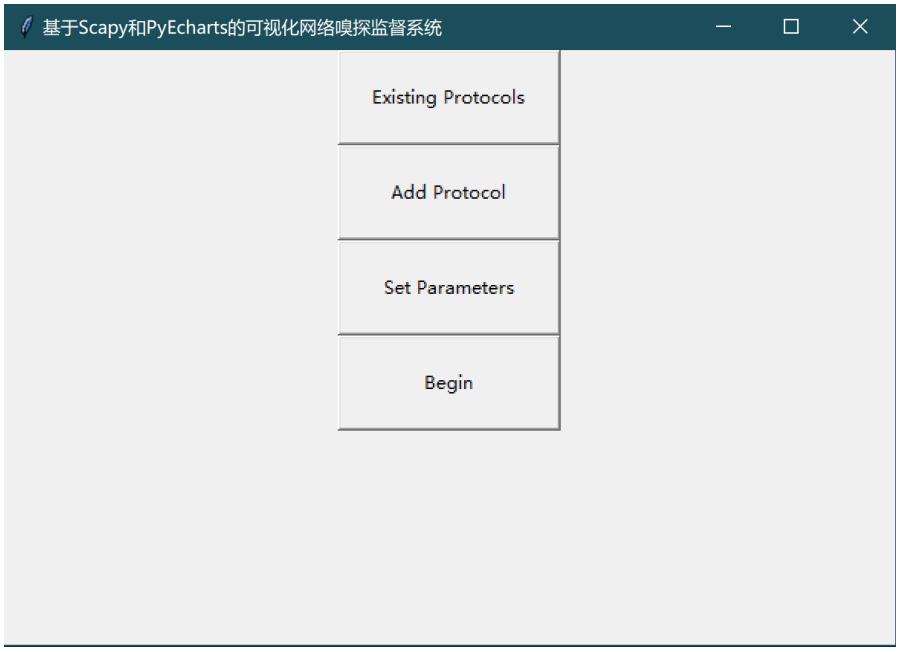
任务 4：界面与跳转设计，加入额外的规则

```
    def GUI():
        def func1():
            win1 = tkinter.Tk()
            win1.title("Existing Protocols and Parameters")
            text = tkinter.Text(win1, width=50, height=20)
            text.pack()
            str = 'Round=' + repr(Round) + '\nCnt=' + repr(Cnt) + '\nBreaktime=' +
repr(breaktime)
            str += '''\ndict['121.51']=Tencent QQ\ndict['210.41']=Tencent
QQ\ndict['111.231']= bilibili'''
            for key in dict:
                str += '\ndict[' + key + ']=' + attr[dict[key]]
            text.insert(tkinter.INSERT, str)
            win1.mainloop()

        def func2():
            def funcadd():
                dict[entry1.get()] = int(entry2.get())

            win1 = tkinter.Tk()
            win1.title("Add Protocol")
            tkinter.Label(win1, text="IP Address").grid(row=0)
            tkinter.Label(win1, text="Value").grid(row=1)
            e = tkinter.Variable()
            e2 = tkinter.Variable()
            entry1 = tkinter.Entry(win1, textvariable=e)
            entry2 = tkinter.Entry(win1, textvariable=e2)
            entry1.grid(row=0, column=1)
            entry2.grid(row=1, column=1)
            button5 = tkinter.Button(win1, text="Add", command=funcadd, width=20,
height=1)
            button5.grid(row=2)
            win1.mainloop()

        def func3():
            def funcadd():
                global Round, Cnt, breaktime
```

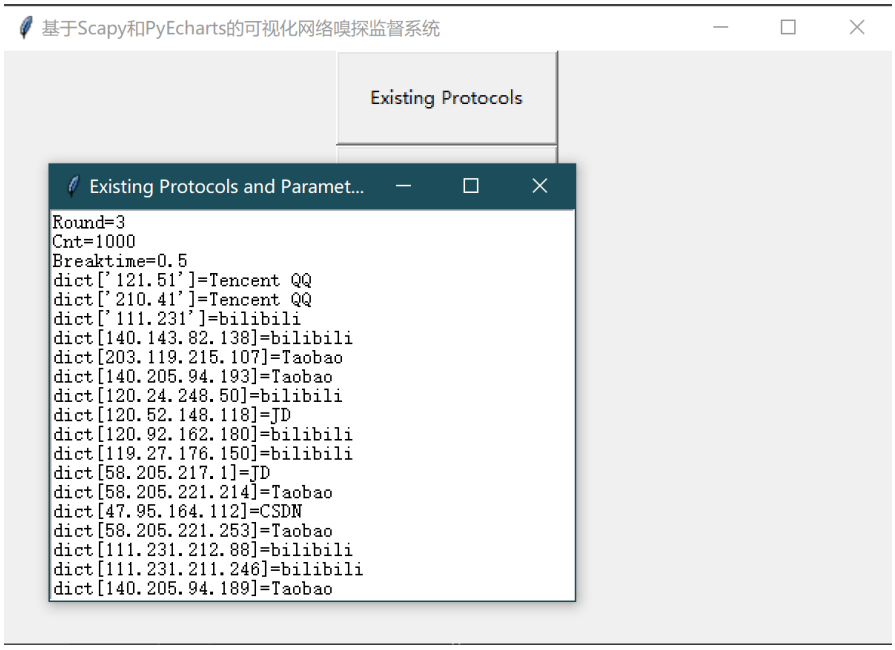
```
                Round = int(entry1.get())
                Cnt = int(entry2.get())
                breaktime = int(entry3.get())

            win1 = tkinter.Tk()
            win1.title("Set Parameters")
            tkinter.Label(win1, text="Round").grid(row=0)
            tkinter.Label(win1, text="Cnt").grid(row=1)
            tkinter.Label(win1, text="Breaktime").grid(row=2)
            e = tkinter.Variable()
            e2 = tkinter.Variable()
            e3 = tkinter.Variable()
            entry1 = tkinter.Entry(win1, textvariable=e)
            entry2 = tkinter.Entry(win1, textvariable=e2)
            entry3 = tkinter.Entry(win1, textvariable=e3)
            entry1.grid(row=0, column=1)
            entry2.grid(row=1, column=1)
            entry3.grid(row=2, column=1)
            button6 = tkinter.Button(win1, text="Add", command=funcadd, width=20,
height=1)
            button6.grid(row=3)
            win1.mainloop()

        def func4():
            main()

        make_link_ip()

        win = tkinter.Tk()
        win.title("基于Scapy和PyEcharts的可视化网络嗅探监督系统")
        win.geometry("600x400+200+50")

        button1 = tkinter.Button(win, text="Existing Protocols", command=func1,
width=20, height=3)
        button1.pack()

        button2 = tkinter.Button(win, text="Add Protocol", command=func2, width=20,
height=3)
        button2.pack()

        button3 = tkinter.Button(win, text="Set Parameters", command=func3,
width=20, height=3)
        button3.pack()
```

```
        button4 = tkinter.Button(win, text="Begin", command=func4, width=20,
height=3)
        button4.pack()

        win.mainloop()
```

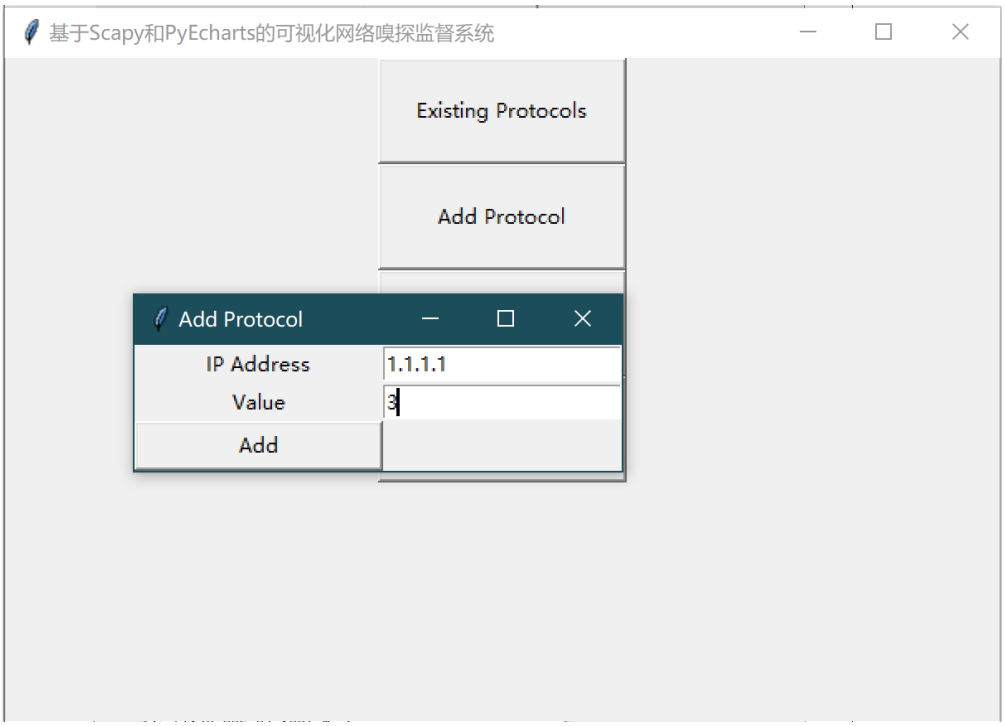
GUI 界面



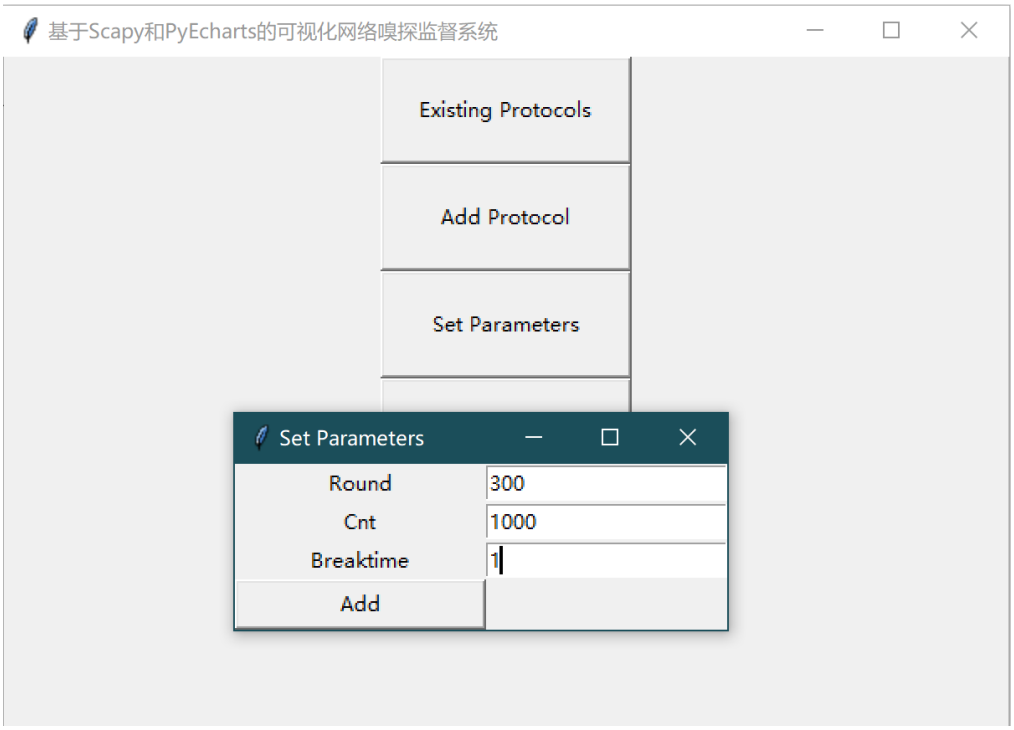
第一项按钮我们可以查看已有的抓包规则；初始化时我们已有一套规则，如有需要可做添加和修改。



第二项按钮我们可以加入新的抓包规则：



第三个按键可以设置参数：



第四个按钮就开始了整个过程，在经过了由参数设置的时间后，会生成 page.html 报告，可以由浏览器打开。

报告示例内容可参见下节。

5、课程设计成果展示

（1）功能测试用例及截图

测试了从 **2019-05-12 09:58:47** 到 **2019-05-12 10:31:31** 之间的报文。累计截获了共计 **300*1000=300000** 份报文并进行了分析。结果如图 3-图 6 所示。各图的含义在“课程设计任务及特色”中已做说明。

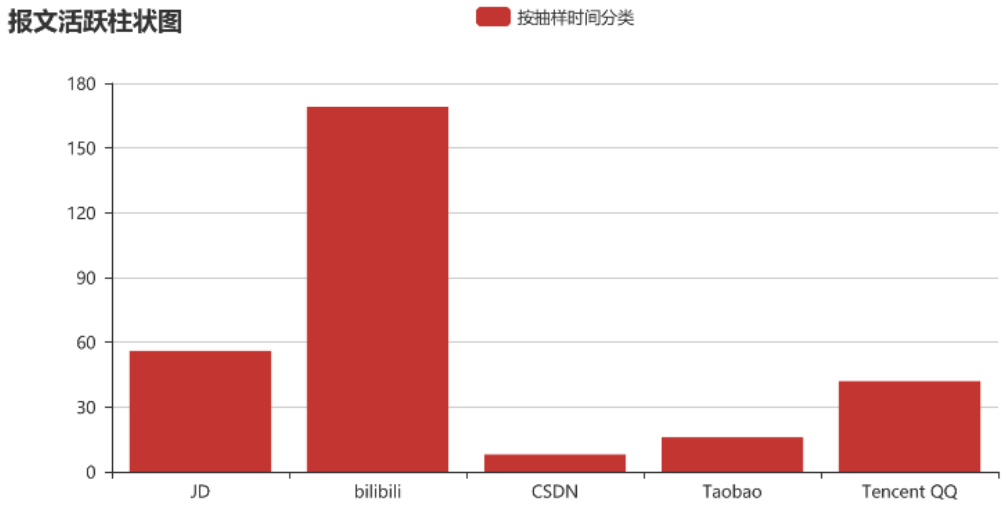


图 3 报文活跃柱状图

图 3 可以看出我在该时段内各个应用的使用时间，与此约半个小时的访问实际情况相一致。

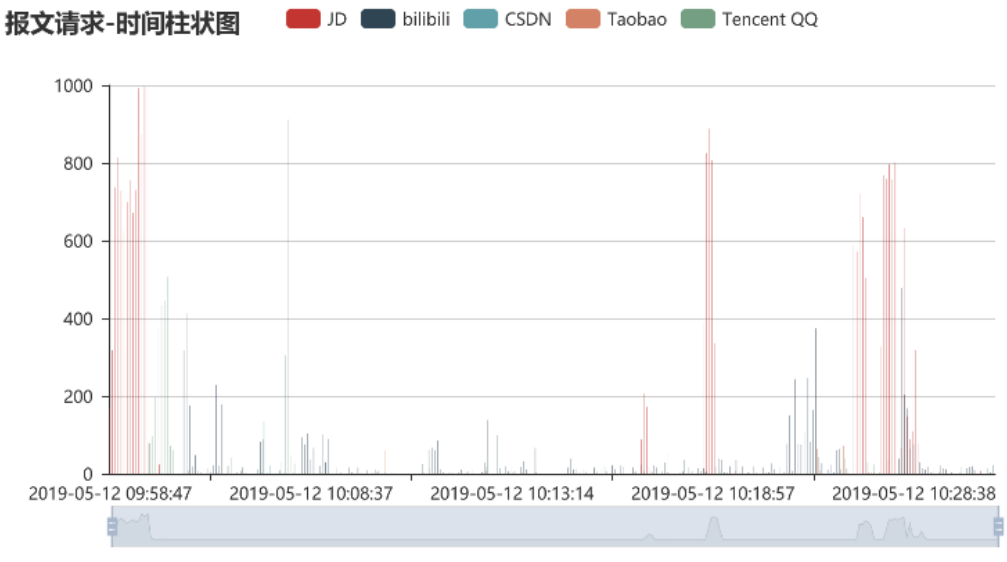


图 4 报文请求-时间柱状图

图 4 可以看出我在该时段内各个时刻的浏览状况，与实际情况相一致。在 HTML 中，可以选择只显示某一应用或某些应用的调查结果。

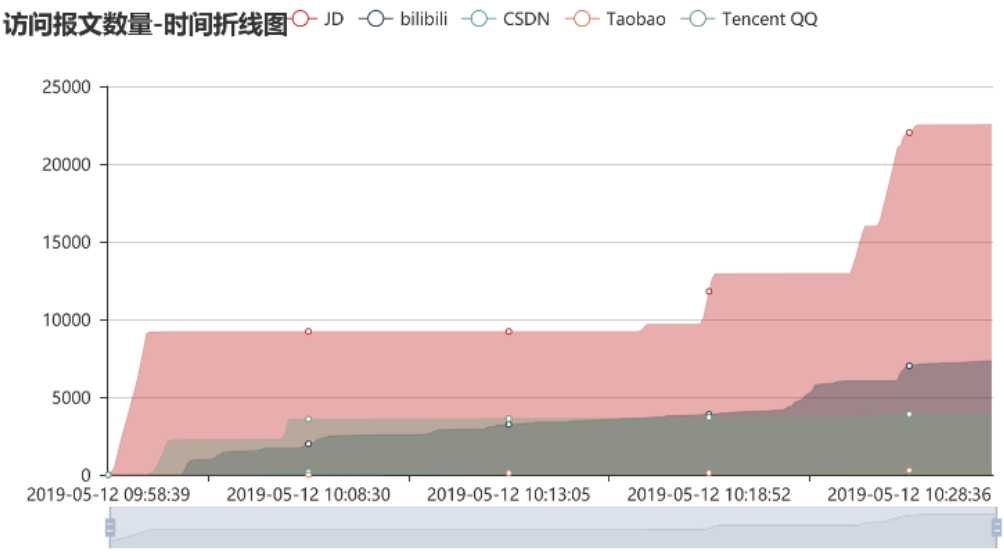


图 5 访问报文数量-时间折线图

图 5 是对图 4 进行累加的结果，可以看出总报文数量的变化。

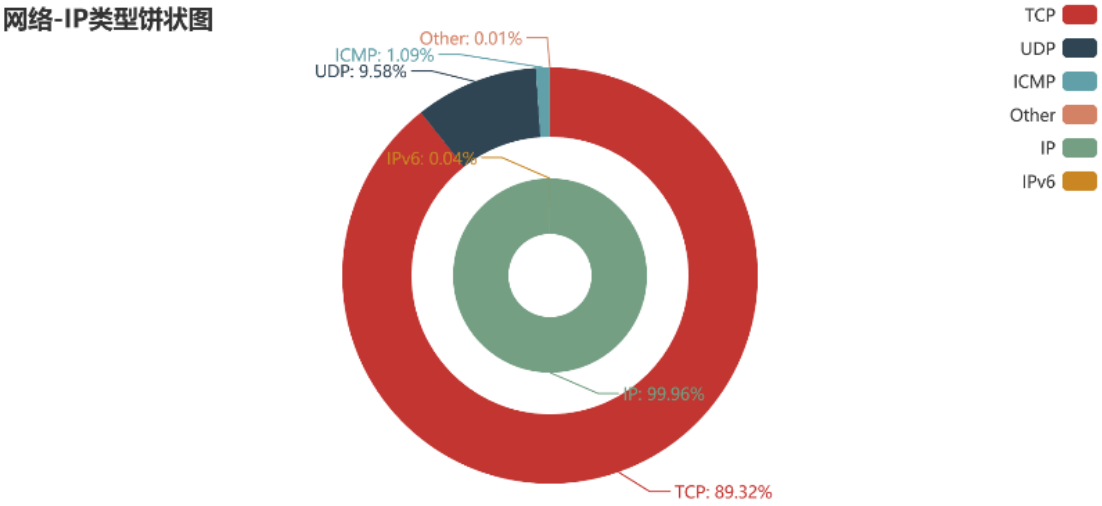


图 6 网络-IP 类型饼状图

图 6 则显示了一些基础数据，包括了各个报文的种类及其比例。

（2）性能测试用例及截图

鉴于本身设计的抽样与等待机制，该程序的运行并不会很大影响系统性能。如图所示，在当前的主流配置下，该程序是否在后台运行对计算机的影响并不大，只与其他程序保持相同的水平，并未占用大量的资源。

如果该程序占用了较多资源，还可以在保证抽样合理性的范围内适当调整参数，使抽样的等待时间延长，或减少抽样个数，都可以理论上再提高效率。

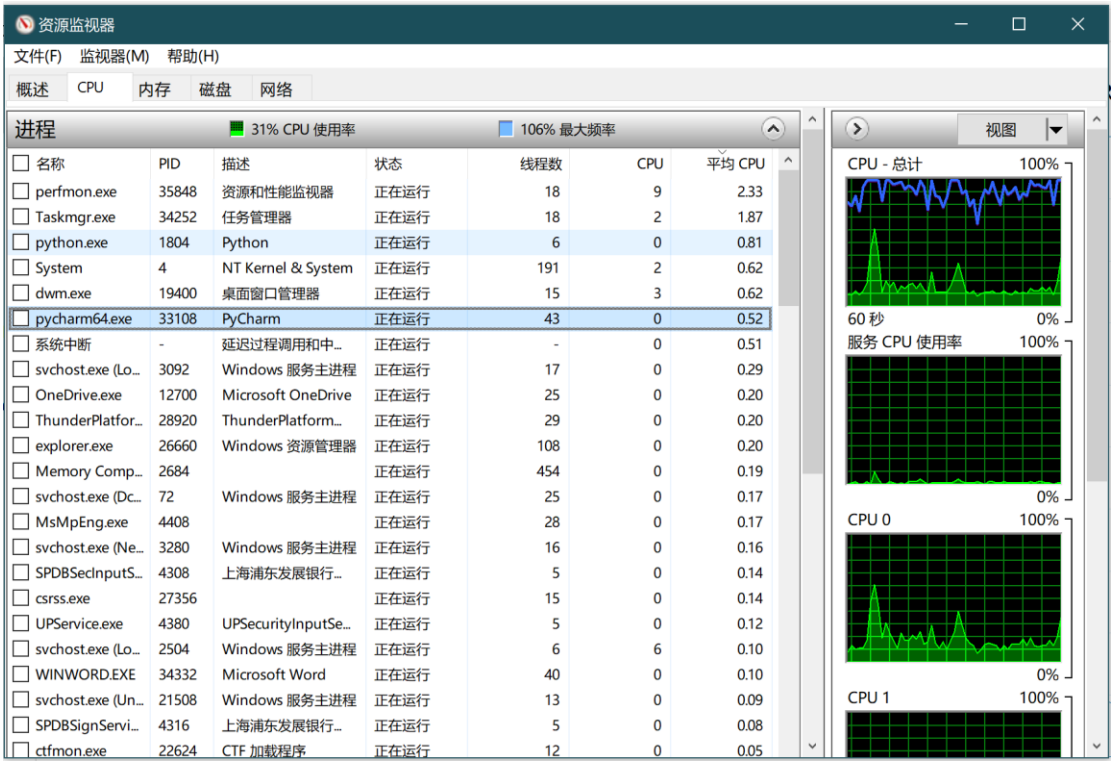


图 7 资源监视

6、总结及互评

以上涉及的第三方库都是初次使用，很难顺利地直接进行代码编写。为此，我们阅读了官方文档，并从官方文档与各大博客中寻找解决方案，并进行了讨论与沟通。最终，我们对统计的机制进行了设计，并完成了这样一个软件。

单个网站的 IP 地址可能发生变化，且单个域名或者同一网站下的不同功能可能使用不同的 IP 地址，这样我们的映射需要实时更新，这是一个庞大的工程。我们考虑如果在运行程序前先进行 Ping 程序进行校准，实时地学习目标域名与 IP 地址地映射关系可以解决这个问题，但因为时间有限我们并没有完成。我们发现，网上存在收费的第三方接口可供使用，提供了更详细的 IP 信息展示，如果再次对程序进行改进，可以尝试购买第三方接口并加入程序。

分工：

彭子为：方法设计，代码编写，文档编写

程允锐：抓取网站地址，代码编写，文档修改