一、背景
二、环境及技术
1、环境
2、技术
三、产品介绍
1、基本功能
①支持代理
已实现功能
计划实现功能
② 获取流量数据
③ 数据库（内含身份验证）
Ⅰ用户信息系统
已实现功能
计划功能：
Ⅱ 网络监管系统
已实现功能：
计划功能：
④ 令牌桶
实现过程
计划实现功能
⑤ 流量数据处理
计划实现功能
⑥ GUI
GUI样式
使用技术
显示内容
使用方式
2、拓展功能
① 完善令牌桶限速功能
② 提取信息
③ 使用大数据知识分布式处理数据（未完全完成，只完成了框架）
四、优势
能获取更详细的数据
操作简单
五、未来发展方向
六、附录
localGui.py
localproxy.py
remoteproxy.py
data_main.py
data_preparation.py
data_extract.py

一、背景

现在，在生活中，无论是学习，工作，还是娱乐，我们几乎都离不开网络，但是能帮助我们个性化处理网络的工具似乎也不是很多，当我们不满足360等安全软件提供的网络数据信息，但又觉得利用抓包软件分析过于麻烦；当我们想像写日记一样记录我们的浏览记录，但又不满足"历史纪录"提供的那少的可怜的信息；当我们想同时处理多个需要用到网络的任务，但又想自己分配他们的带宽...
是的，大众软件不需要配置不需要操作，但它能提供的数据少的可怜。专业软件数据详尽，但可能过于详尽，缺乏统计。我们也许需要一种快捷而精致的软件，让我们快速的获取有关自己网络的细致信息，并且能够配置自己的网络。也许，我们所设计的这款软件，更多更多和我们有相同想法的人，都想要。

二、环境及技术

1、环境

windows

python 3.8

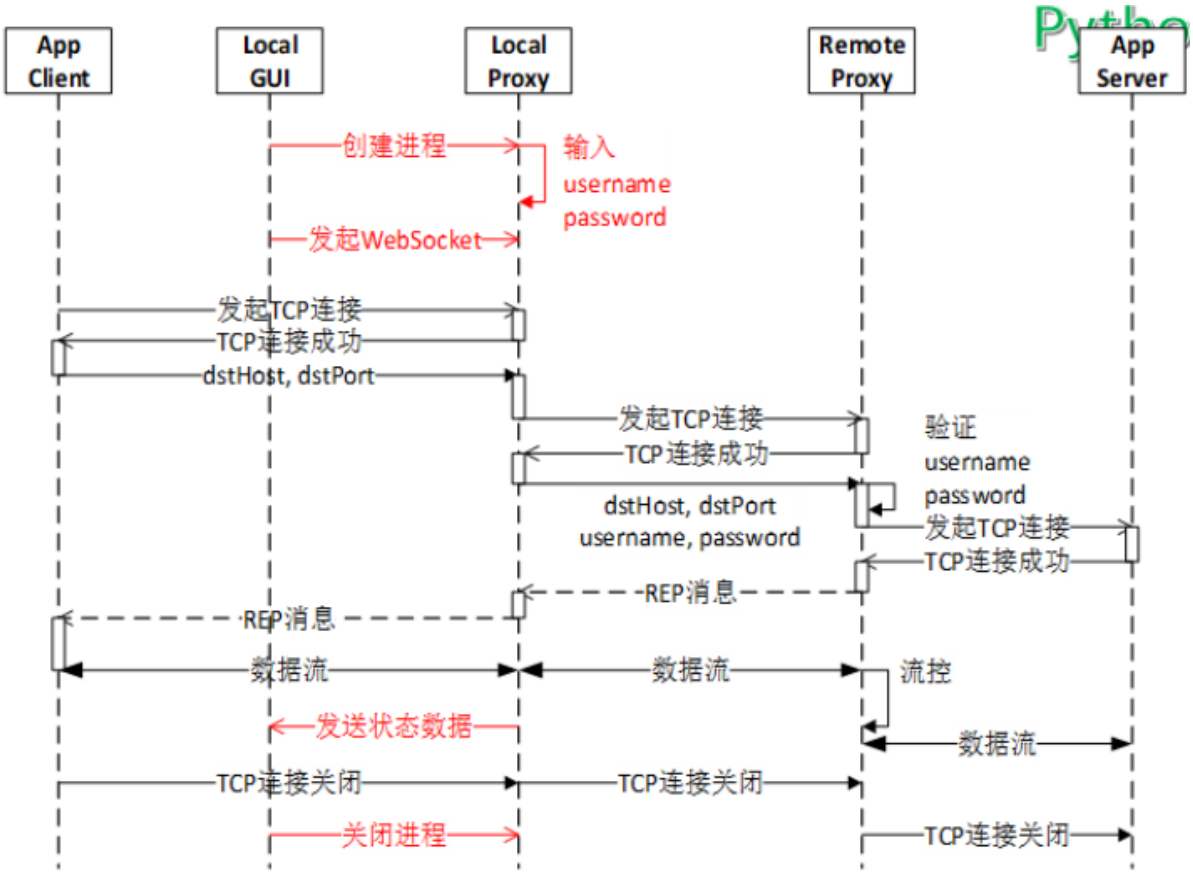
2、技术

协程

sqlite

三、产品介绍

1、基本功能



①支持代理

已实现功能

http和socks5

这是目前我们的软件支持的代理方式，在浏览器设置代理后，启动我们的软件，浏览器在使用的过程中，所有数据将经过我们的软件发送出去，代理方式可自动识别，用户无需做更多操作。在后续的版本中，我们将实现全局代理的方式，使得用户的使用更加方便。

计划实现功能

其他协议的通信

② 获取流量数据

在代理服务器local端进行处理，计算出数据传输的带宽，将传输的带宽数据传输到GUI端口进行显示

③ 数据库（内含身份验证）

Ⅰ用户信息系统

已实现功能

用户将通过用户名和密码登陆账号，用以区别。用户信息将被存储在数据库里面

计划功能：

将实现注册功能，用户登陆自己的账号之后，可以设置一些自己的偏好设置，，同时，网络监管模块将根据不同账号进行更加细致从处理

Ⅱ 网络监管系统

已实现功能：

通过代理，用户通过我们的软件可以获取浏览器接收和发送的所有信息，我们将记录用户访问过的域名及其ip和近期访问次数。

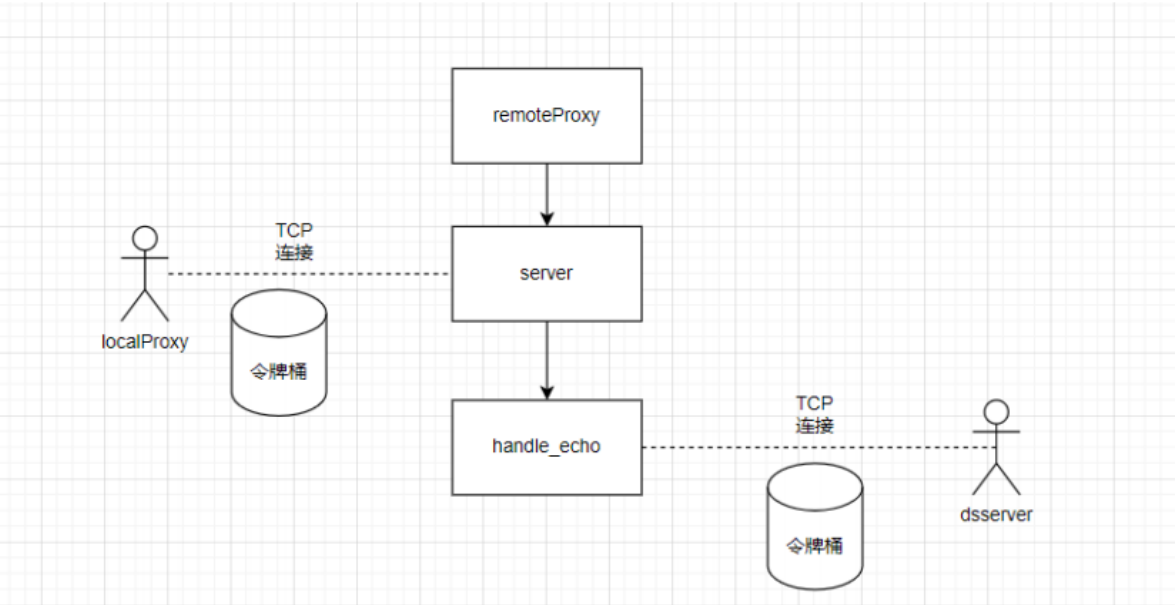
计划功能：

① 我们的软件可以根据近期访问次数，对近期经常访问的网页进行内部加速，网页的储存与替换采用LFU算法。内部加速将使得我们的软件对用户原网速的影响更小，从而在其有较高网速需求的时候，不会因为我们的软件被限制。同时，用户可以通过设置“临界网速”，配置软件，当需要的网速大于临界网速时，我们的软件将会逐渐降低检测力度，直到放弃检测，这将最大限度的在实现我们软件功能的同时，降低我们的软件对用户使用的影响。

② 我们的软件将根据统计信息，生成报表，用户可选择输出方式，比如csv等。

软件将给出访问信息的统计信息，显示主要网站的流量数据，按天，周或者月，统计新出现的网页，和消失的网页。用户在查看时，如果莫名出现陌生网页，则怀疑被攻击的可能性，当即时更改重要密码。同时用户也可以查看自己前一段时间的网络使用情况，类似于日记功能，方便用户的自我管理

④ 令牌桶



实现过程

因为每次发送信息，都会访问一次令牌桶，我们利用这个特性去实现令牌桶算法。

① 访问令牌桶时，加锁。

② 访问令牌桶动都会记录一次当前时刻的时间，用当前时刻的时间减去上一次访问令牌桶的时间，再乘以生成令牌桶的速率，可以得到现有的令牌数。（这个令牌数不能超过令牌桶容量，如果超出了，则令当前令牌数等于令牌桶最大容量）

③ 在访问令牌桶时，会传来一个参数，这个参数是需要消耗的令牌数量，如果当前令牌数量多于需要消耗的令牌数，则相减后得到剩余令牌，然后释放锁；反之，则释放锁之后，等待下一次访问令牌桶，直至当前令牌数量多余需要消耗的令牌数量。

计划实现功能

新增一个设置，用户可以控制是否限速。如果不限速。

⑤ 流量数据处理

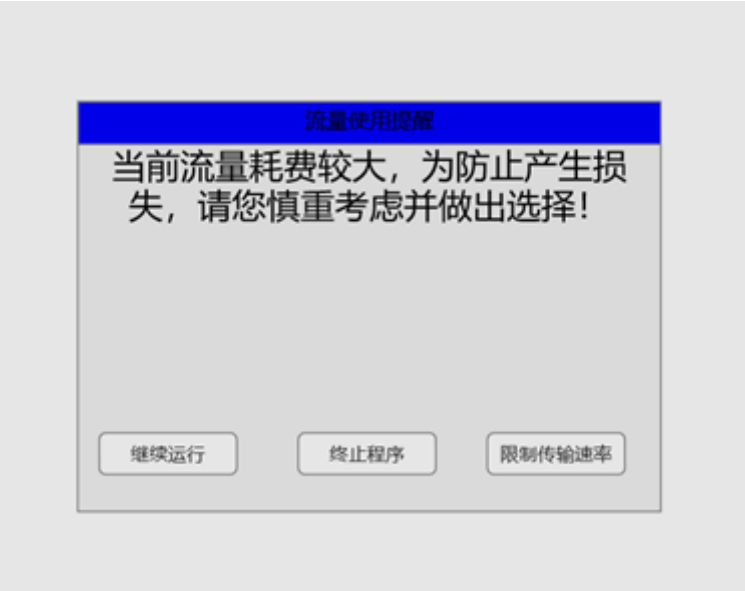
代理服务器的基本功能已经实现，在此基础上，根据对速率的监控，定期统计速率数据并对速率的增长趋势进行分析，若通过代理服务器的数据传输带宽有突发增长并持续一段时间，警示用户，用户可自己进行决策来确定是否继续，此功能的主要用途有两点：

① 防止流量使用过度而对用户带来损失。

② 进入部分网页点击不慎产生的捆绑下载。

计划实现功能

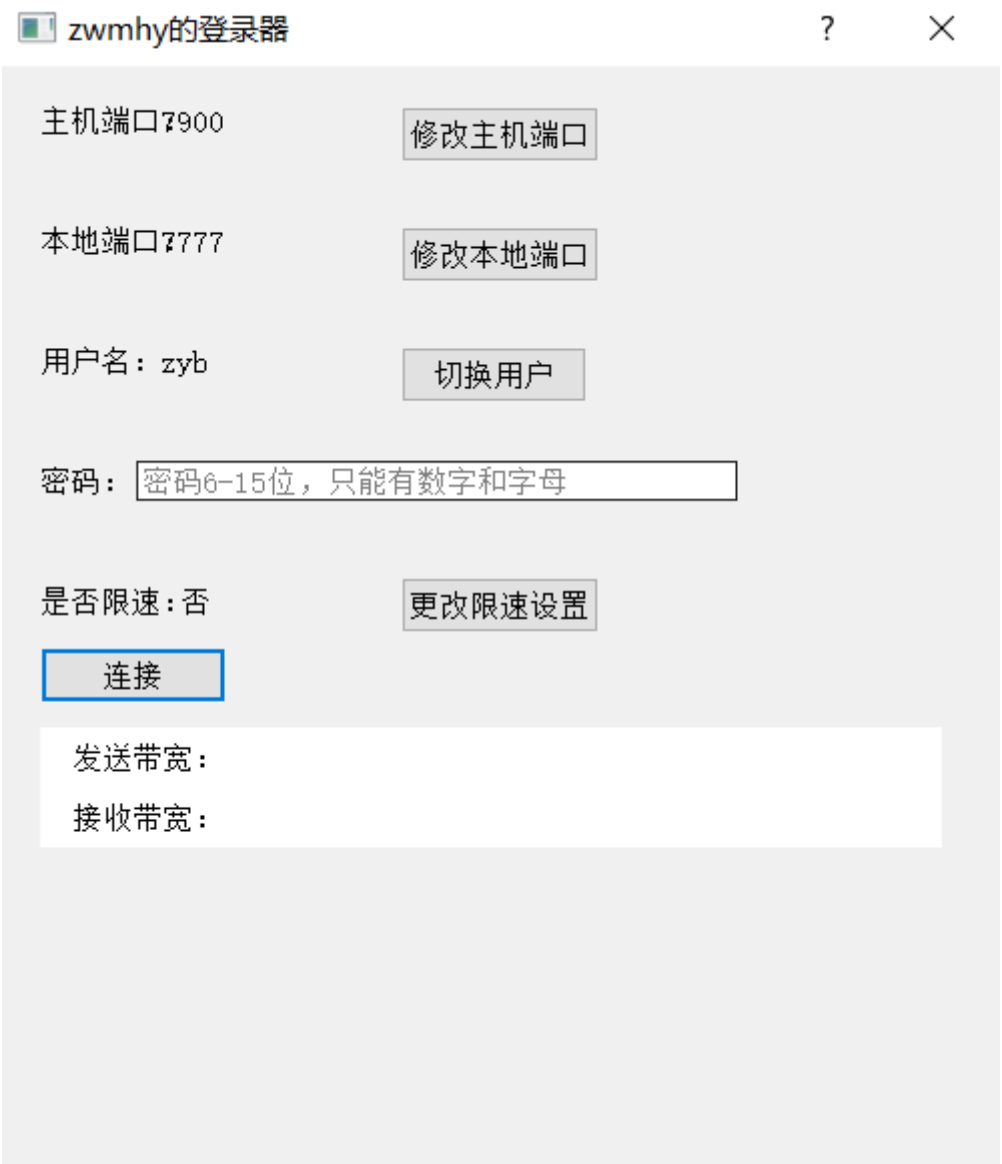
周期性检测传输带宽，用户在使用代理时如果带宽突发变大并且持续一定时间段，弹出警示框提醒用户，界面和选项初步计划如下图：

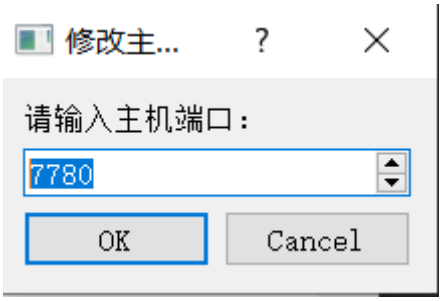


用户可选择继续使用代理传输，或者终止代理，或者通过限制传输速率来降低流量的消耗速度。限速连接另一个扩展功能，用户可以输入期望速率然后将数据传输到限速功能，采用令牌桶算法，对流量数据进行限速。用户可以根据GUI端的速率显示选择合理的限速速率。

⑥ GUI

GUI样式





使用技术

PyQt5

显示内容

① 主机端口

② 本地端口

③ 用户名

④ 密码（用掩码显示）

⑤ 显示带宽

使用方式

① 更改主机端口、本地端口、用户名、密码、是否限速。为了避免冲突，我们设置了本机端口和本地端口的范围。

② 点击连接，GUI将与 local 建立 websocket 连接，可以实时监控流量数据，并将其显示在 GUI 上。

2、拓展功能

① 完善令牌桶限速功能

原来的写法虽然也可以限速，但是本质上是错误的，这个版本将原来的错误修正了。

初始版本：每个用户有独属于自己的令牌数用来限速。

现在的版本：所有用户共用一个令牌数，只有被分配到令牌的用户才能用代理浏览网页。

② 提取信息

在 localproxy 和 remoteproxy 转发数据包的过程中，获取数据包的信息，并且进行数据的提取。

利用正则表达式，获取网址和 HTTP版本

```
第 0 个文件：   tsbrowser.xiangtatech, rwww.baidu
第 1 个文件：   tsbrowser.xiangtatech, rwww.baidu
第 2 个文件：   tsbrowser.xiangtatech, rwww.baidu
第 3 个文件：   tsbrowser.xiangtatech, rwww.baidu
```

```
第 0 个文件：   tsbrowser.xiangtatech.com, rwww.baidu.com
第 1 个文件：   tsbrowser.xiangtatech.com, rwww.baidu.com
第 2 个文件：   tsbrowser.xiangtatech.com, rwww.baidu.com
第 3 个文件：   tsbrowser.xiangtatech.com, rwww.baidu.com
```

```
第 0 个文件：   http/1.1
第 1 个文件：   http/1.1
第 2 个文件：   http/1.1
第 3 个文件：   http/1.1
```

③ 使用大数据知识分布式处理数据（未完全完成，只完成了框架）

环境：

Java：java-1.8.0

Python：python-3.8.5

Scala：scala-2.11.8

Flume：apach-flume-1.9.0

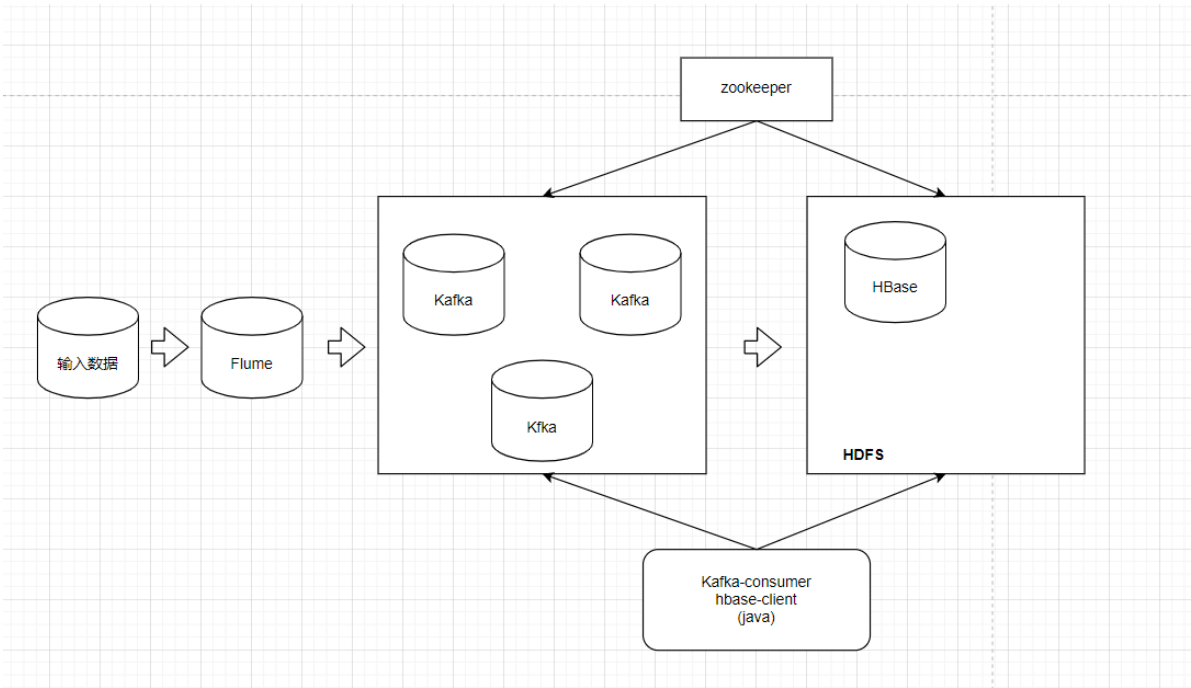
Kafka：kafka_2.11-2.4.1

Hadoop：hadoop-2.10.1

HBase：hbase-2.3.5

Spark：spark-2.4.7

在 linux 配置好相关环境之后，将提取出来的数据信息从flume输入，编写 java 程序，按照下面的流程图将输入的信息存入到HBase的表里。



接下来编写处理数据信息的java程序，将提交到 spark 上运行，进行分布式地处理这些数据，并将结果写入HDFS中。

这样我们可以更快、更准地获取到我们想得到的结果。比如：访问比较频繁的网页、存在安全问题无法访问的网页……

现在这个框架我们已经搭好了，但是因为时间的关系和基础功能出现了一些bug，我们没有继续去做这方面的内容，这个拓展部分只完成到了这个程度，没有去思考我们应该根据提取出来的数据去做什么样的数据处理。

四、优势

能获取更详细的数据

比起一些大众的浏览器自带的服务器，比如360，只可以简单地监控一下网速，而我们地产品，除了网速，还可以提供流量、网页访问次数等一些其他数据。

操作简单

wireshark等抓包软件呢，可以获取更为详细的数据，但是这些软件操作相对麻烦，不适合大众使用。相对的，我们的产品给这些数据提供一些接口，用户可以非常简单地使用。

五、未来发展方向

目前我们的软件主要应用于个人计算机，帮助个人统计网络的使用信息。经过一些修改，它同样可以用于一个局域网的流量管理与监控，我们的软件将具有应用到企业里面的潜力。在后续的版本中，我们将会考虑使用java或则c++重写我们的软件，实现更高的效率，同时优化我们的代码，进一步提高效率。

六、附录

localGui.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'Gui.ui'
#
# Created by: PyQt5 UI code generator 5.15.1
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.

import sys
import logging
import os
import asyncio
import humanfriendly
import time
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import *
from PyQt5.QtNetwork import *
from PyQt5.QtWebSockets import *
from PyQt5.QtGui import *
from PyQt5.QtCore import *

#global gl_username
#global gl_password

# 主窗口
class Ui_Form_1(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(484, 390)
        self.Go = QtWidgets.QPushButton(Form)
        self.Go.setGeometry(QtCore.QRect(190, 170, 93, 28))
        self.Go.setObjectName("Go")
        self.Go.clicked.connect(self.enter)

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.Go.setText(_translate("Form", "Welcome"))

    def enter(self):
        self.hide()
        self.s = My_second_Form()
        self.s.show()

# 一级界面
class Ui_Form_2(object):
    def setupUi(self, Form):
        #global gl_username
```

```
        #global gl_password
        Form.setObjectName("Form")
        Form.resize(615, 406)
        self.username_lable = QtWidgets.QLabel(Form)
        self.username_lable.setGeometry(QtCore.QRect(20, 120, 101, 21))
        self.username_lable.setObjectName("username_lable")
        self.input_username = QtWidgets.QLineEdit(Form)
        self.input_username.setGeometry(QtCore.QRect(110, 120, 113, 21))
        self.input_username.setText("")
        self.password_lable = QtWidgets.QLabel(Form)
        self.password_lable.setGeometry(QtCore.QRect(20, 180, 101, 20))
        self.password_lable.setObjectName("password_lable")
        self.input_password = QtWidgets.QLineEdit(Form)
        self.input_password.setGeometry(QtCore.QRect(110, 180, 113, 21))
        self.input_password.setObjectName("input_password")
        self.input_password.setEchoMode(QLineEdit.Password)
        self.show_text = QtWidgets.QTextBrowser(Form)
        self.show_text.setGeometry(QtCore.QRect(280, 80, 256, 192))
        self.show_text.setObjectName("show_text")
        self.go_btn = QtWidgets.QPushButton(Form)
        self.go_btn.setGeometry(QtCore.QRect(120, 300, 61, 28))
        self.go_btn.setObjectName("go_btn")
        self.exit_btn = QtWidgets.QPushButton(Form)
        self.exit_btn.setGeometry(QtCore.QRect(340, 300, 61, 28))
        self.exit_btn.setObjectName("exit_btn")

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

        self.go_btn.clicked.connect(self.show_msg)
        self.exit_btn.clicked.connect(self.Hide)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.username_lable.setText(_translate("Form", "Username"))
        self.password_lable.setText(_translate("Form", "Password"))
        self.go_btn.setText(_translate("Form", "登录"))
        self.exit_btn.setText(_translate("Form", "退出"))

    # 显示欢迎信息
    def show_msg(self):
        self.username = self.input_username.text()
        self.password = self.input_password.text()
        if self.username!='' and self.password!='':
            self.show_text.setText(f'Welcome, {self.username}')
            self.show_text.show()

            self.go_btn.setText('next')
            self.go_btn.clicked.connect(self.monitor)
        # 显示错误界面
        else:
            self.e = Error_Form()
            self.e.show()

    # 显示监控（图形化）
    def monitor(self):
        self.hide()
```

```
        self.m = My_third_Form()
        self.m.show()

    # 退出
    def Hide(self):
        self.hide()
        self.h = MyMainForm()
        self.h.show()

# 二级界面
class Ui_Form_3(Ui_Form_2):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(762, 529)
        self.scrollArea = QtWidgets.QScrollArea(Form)
        self.scrollArea.setGeometry(QtCore.QRect(99, 76, 561, 341))
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")
        self.scrollAreaWidgetContents = QtWidgets.QWidget()
        self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0, 559, 339))
        self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents")
        self.tableWidget = QtWidgets.QTableWidget(self.scrollAreaWidgetContents)
        self.tableWidget.setGeometry(QtCore.QRect(0, 0, 561, 341))
        self.tableWidget.setColumnCount(4)
        self.tableWidget.setObjectName("tableWidget")
        self.tableWidget.setRowCount(0)
        item = QtWidgets.QTableWidgetItem()
        self.tableWidget.setHorizontalHeaderItem(0, item)
        item = QtWidgets.QTableWidgetItem()
        self.tableWidget.setHorizontalHeaderItem(1, item)
        item = QtWidgets.QTableWidgetItem()
        self.tableWidget.setHorizontalHeaderItem(2, item)
        item = QtWidgets.QTableWidgetItem()
        self.tableWidget.setHorizontalHeaderItem(3, item)
        self.scrollArea.setWidget(self.scrollAreaWidgetContents)
        self.Start_btn = QtWidgets.QPushButton(Form)
        self.Start_btn.setGeometry(QtCore.QRect(160, 430, 93, 28))
        self.Start_btn.setObjectName("Start_btn")
        self.Stop_btn = QtWidgets.QPushButton(Form)
        self.Stop_btn.setGeometry(QtCore.QRect(480, 430, 93, 28))
        self.Stop_btn.setObjectName("Stop_btn")
        self.Exit_btn = QtWidgets.QPushButton(Form)
        self.Exit_btn.setGeometry(QtCore.QRect(310, 480, 93, 28))
        self.Exit_btn.setObjectName("Exit_btn")
        self.Host_label = QtWidgets.QLabel(Form)
        self.Host_label.setGeometry(QtCore.QRect(410, 20, 71, 21))
        self.Host_label.setObjectName("Host_label")
        self.ConsolePort_lable = QtWidgets.QLabel(Form)
        self.ConsolePort_lable.setGeometry(QtCore.QRect(410, 50, 91, 21))
        self.ConsolePort_lable.setObjectName("ConsolePort_lable")
        self.Host_text = QtWidgets.QLineEdit(Form)
        self.Host_text.setGeometry(QtCore.QRect(500, 20, 141, 21))
        self.Host_text.setText("")
        self.Host_text.setObjectName("Host_text")
        self.ConsolePort_text = QtWidgets.QLineEdit(Form)
        self.ConsolePort_text.setGeometry(QtCore.QRect(500, 50, 131, 21))
        self.ConsolePort_text.setObjectName("ConsolePort_text")
        self.username_lable = QtWidgets.QLabel(Form)
```

```
        self.username_lable.setGeometry(QtCore.QRect(100, 20, 72, 21))
        self.username_lable.setObjectName("username_lable")
        self.label = QtWidgets.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(100, 50, 72, 21))
        self.label.setObjectName("label")
        self.username_text = QtWidgets.QLineEdit(Form)
        self.username_text.setGeometry(QtCore.QRect(170, 20, 151, 21))
        self.username_text.setObjectName("username_text")
        self.password_text = QtWidgets.QLineEdit(Form)
        self.password_text.setGeometry(QtCore.QRect(170, 50, 151, 21))
        self.password_text.setObjectName("password_text")
        self.password_text.setEchoMode(QLineEdit.Password)

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

        self.Stop_btn.clicked.connect(self.Stop)
        self.Start_btn.clicked.connect(self.Start)
        self.Exit_btn.clicked.connect(self.Hide)

        # -----使用QProcess类管理localProxy-----
        self.process = QProcess()
        self.process.setProcessChannelMode(QProcess.MergedChannels)
        #self.process.finished.connect(self.process_finished)
        self.process.started.connect(self.process_started)
        self.process.readyReadStandardOutput.connect(self.process_readyread)


    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        item = self.tableWidget.horizontalHeaderItem(0)
        item.setText(_translate("Form", "Time"))
        item = self.tableWidget.horizontalHeaderItem(1)
        item.setText(_translate("Form", "Connect or not"))
        item = self.tableWidget.horizontalHeaderItem(2)
        item.setText(_translate("Form", "SendBandWidth"))
        item = self.tableWidget.horizontalHeaderItem(3)
        item.setText(_translate("Form", "RecvBandWidth"))
        self.Start_btn.setText(_translate("Form", "Start"))
        self.Stop_btn.setText(_translate("Form", "Stop"))
        self.Exit_btn.setText(_translate("Form", "Exit"))
        self.Host_label.setText(_translate("Form", "Host"))
        self.ConsolePort_lable.setText(_translate("Form", "ConsolePort"))
        self.username_lable.setText(_translate("Form", "username"))
        self.label.setText(_translate("Form", "password"))

    # 开始监控
    def Start(self):
        self.host = self.Host_text.text()
        self.consolePort = self.ConsolePort_text.text()
        msg = Ui_Form_2()
        self.username = self.username_text.text()
        self.password = self.password_text.text()
        pythonExec = os.path.basename(sys.executable)

        cmdLine = f'{pythonExec} localproxy.py -u {self.username} -p
{self.password} -c {self.consolePort}'
```

```
        # print(cmdLine)
        logging.debug(f'cmd={cmdLine}')
        self.process.start(cmdLine)

    def process_readyread(self):
        data = self.process.readAll()
        #print(type(data))
        try:
            msg = data.data().decode('utf-8').strip()
            logging.debug(f'msg={msg}')
        except Exception as exc:
            # logging.error(f'{traceback.format_exc()}')
            exit(1)

    def process_started(self):
        # 等同于self.process，使用sender适应性更好
        process = self.sender()
        processId = process.processId()
        logging.basicConfig(filename='example.log', level=logging.DEBUG)
        logging.debug(f'pid={processId}')
        #self.processIdLine = QLineEdit()
        #self.processIdLine.setText(str(processId))

        self.websocket = QWebSocket()
        self.websocket.connected.connect(self.websocket_connected)
        self.websocket.disconnected.connect(self.websocket_disconnected)
        self.websocket.textMessageReceived.connect(self.websocket_message_rec)
        self.websocket.open(QUrl(f'ws://127.0.0.1:
{self.ConsolePort_text.text()}/'))

    def websocket_connected(self):
        self.websocket.sendTextMessage('secret')

    def websocket_disconnected(self):
        self.process.kill()

    def websocket_message_rec(self, msg):
        # print(msg)
        logging.debug(f'msg={msg}')
        send_Bandwidth, recv_Bandwidth, *_ = msg.split()
        self.nowTime = QDateTime.currentDateTime().toString('hh:mm:ss')
        self.sendmsg_input = f'{humanfriendly.format_size(int(send_Bandwidth))}'
        self.recvmsg_input = f'{humanfriendly.format_size(int(recv_Bandwidth))}'
        row = self.tableWidget.rowCount()   # 返回当前行数
        self.tableWidget.insertRow(row)     # 尾部插入一行新行表格
        col = self.tableWidget.columnCount()# 返回当前列数
        self.tableWidget.setItem(row, 0, QTableWidgetItem(self.nowTime))
        self.tableWidget.setItem(row, 1, QTableWidgetItem('connect'))
        self.tableWidget.setItem(row, 2, QTableWidgetItem(self.sendmsg_input))
        self.tableWidget.setItem(row, 3, QTableWidgetItem(self.recvmsg_input))

    # 进程停止
    def Stop(self):
        self.process.kill()

    # 退出
    def Hide(self):
        self.hide()
```

```
        self.h = My_second_Form()
        self.h.show()

# 错误界面
class Error_Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(400, 124)
        self.error_lable = QtWidgets.QLabel(Form)
        self.error_lable.setGeometry(QtCore.QRect(60, 40, 291, 41))
        self.error_lable.setFrameShadow(QtWidgets.QFrame.Plain)
        self.error_lable.setObjectName("error_lable")

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.error_lable.setText(_translate("Form", "error: username or password
is null"))

class MyMainForm(QMainWindow, Ui_Form_1):
    def __init__(self, parent=None):
        super(MyMainForm, self).__init__(parent)
        self.setupUi(self)

class My_second_Form(QDialog, Ui_Form_2):
    def __init__(self, parent=None):
        super(My_second_Form, self).__init__(parent)
        self.setupUi(self)

class My_third_Form(QDialog, Ui_Form_3):
    def __init__(self, parent=None):
        super(My_third_Form, self).__init__(parent)
        self.setupUi(self)

class Error_Form(QDialog, Error_Ui_Form):
    def __init__(self, parent=None):
        super(Error_Form, self).__init__(parent)
        self.setupUi(self)

def ui_main():
    app = QApplication(sys.argv)
    window = MyMainForm()
    window.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    ui_main()
```

localproxy.py

```
import argparse
import asyncio
import websockets
import logging
import time
from struct import unpack,pack
import os
path = 'data'
files = os.listdir(path)

SendBand = 0
RecvBand = 0
SendBandwidth = 0
RecvBandwidth = 0

write_data_num=0
write_data=[]
write_max=0
write_loop=0

read_data_num=0
read_data=[]
read_max=0
read_loop=0

data11=''
data22=''

async def localConsole(ws, path):
    global SendBandwidth
    global RecvBandwidth
    try:
        while True:
            await asyncio.sleep(1)
            # print('SendBandwidth, RecvBandwidth ', SendBandwidth, ' ',
RecvBandwidth)
            msg = await ws.send(f'{int(SendBandwidth)} {int(RecvBandwidth)}')
            SendBandwidth = 0
            RecvBandwidth = 0
    except websockets.exceptions.ConnectionClosedError as exc:
        logging.error(f'{exc}')
    except websockets.exceptions.ConnectionClosedOK as exc:
        logging.error(f'{exc}')
    except Exception:
        # logging.error(f'{traceback.format_exc()}')
        exit(1)

async def readfile():
    global read_data,read_loop,read_data_num,data11
    # print(data11)
    if data11 != '':
        # print('read_loop:', read_loop)
        if (read_loop == 0):
            read_data.append(data11)
            read_data_num = read_data_num + 1
```

```
            # print('read_data_num, read_max:', read_data_num, read_max)
            if (read_data_num > read_max):
                read_data_num = 0
                read_loop = 1

                read_data1 = ''.join('%s' %a for a in read_data)
                filename = open(path + '\\' + "read_data1" + ".txt", "a+",
encoding='utf-8')
                # filename = open("read_data1.txt", "a+")
                filename.write(read_data1 + "\n")
                filename.close()

        else:
            read_data[read_data_num] = data11
            read_data_num = read_data_num + 1
            if (read_data_num > read_max):
                read_data_num = 0
                read_data1 = ''.join('%s' %a for a in read_data)
                filename = open(path + '\\' + "read_data2" + ".txt", "a+",
encoding='utf-8')
                # filename = open("read_data2.txt", "a+")
                filename.write(read_data1 + "\n")
                filename.close()
    data11 = ''

async def writefile():
    global write_data, write_loop,write_data_num,data22
    if data22 != '':
        if (write_loop == 0):
            write_data.append(data22)
            write_data_num = write_data_num + 1
            if (write_data_num > write_max):
                write_data_num = 0
                write_loop = 1

                write_data1 = ''.join('%s' %a for a in write_data)
                filename = open(path + '\\' + "write_data1" + ".txt", "a+",
encoding='utf-8')
                # filename = open("write_data1.txt", "a+")
                filename.write(write_data1 + "\n")
                filename.close()
        else:
            write_data[write_data_num] = data22
            write_data_num = write_data_num + 1
            if (write_data_num > write_max):
                write_data_num = 0

                write_data1 = ''.join('%s' %a for a in write_data)
                filename = open(path + '\\' + "write_data2" + ".txt", "a+",
encoding='utf-8')
                # filename = open("write_data2.txt", "a+")
                filename.write(write_data1 + "\n")
                filename.close()
    data22 = ''

async def transport(reader, writer, addr):
    global SendBand
    global RecvBand
```

```
    global data11, data22
    while reader.at_eof:
        try:    # 从reader接收外部报文
            data = await reader.read(1000)
            RecvBand += len(data)

            data11 = data

            if not data:
                writer.close()
                break
        except (ConnectionAbortedError, ConnectionRefusedError) as e:
            writer.close()
            print(f'{addr}异常退出，{repr(e)}')
            break
        try:    # 向writer转发报文
            SendBand += len(data)
            writer.write(data)
            await writer.drain()

            data22 = data
            # asyncio.run(writefile(data))

        except (ConnectionAbortedError, ConnectionRefusedError) as e:
            writer.close()
            print(f'{addr}异常退出，{repr(e)}')
            break
    print(f'{addr}正常退出')

async def count_width():
    global SendBand
    global RecvBand
    global SendBandwidth
    global RecvBandwidth
    time.sleep(1)
    SendBandwidth = SendBand/1
    SendBand = 0
    # print('SendBandwidth, SendBand: ', SendBand, ' ', SendBandwidth)
    print('SendBandwidth:', SendBandwidth)
    RecvBandwidth = RecvBand/1
    RecvBand = 0
    print('RecvBandwidth:', RecvBandwidth)

async def handle_echo(reader, writer):
    data = await reader.read(5000) #将第一个信息读至EOF
    data1 = data.decode()
    httpdata = data1.split(' ')
    # 客户端发送的协商版本和认证方法请求长度不小于3B
    if len(data) < 3:
        print('VER and CMD of is error')
        print('connecte failed')
        return

    a = (args.username, args.password)
    usermas = ' '.join(a).encode()

    # 判断协议的类型(socks5协议或者http协议)
    # socks5协议
```

```
    if data[0] == 5:
        # 解读客户端发来的第一个信息包
        peername = writer.get_extra_info('peername')
        print(f"Create TCP connection with {peername!r}")
        # 向客户端回复版本号5(支持socks5)，并告知采用无验证需求(CMD='00')
        writer.write(b"\x05\x00")
        await writer.drain()     # 等待客户端发送请求
        # 客户端发送请求细节，服务器接收
        data = await reader.read(1024)  #将第二个信息读至EOF
        header = unpack('!BBBB', data[:4])  # 读取前面四个基本信息 VER/CMD/RSV/ATYP
        # 判断socks版本号和需要实现的功能(本代理服务器只支持socks5协议，且只能实现connect功
能)
        if header[0] == 5 and header[1] == 1:
            if header[3] == 1:  #IPv4 X'01'
                try:
                    dsreader, dswriter = await
asyncio.open_connection('127.0.0.5', 1085)   #向远端代理发起连接请求

                    dswriter.write(usermas)
                    await writer.drain()
                    VER = await dsreader.read(50000)
                    if VER[0] != 1:
                        writer.close()

                    dswriter.write(data)   #将数据包发给remoteproxy
                    await dswriter.drain()
                    REP = await dsreader.read(1024) #从远端代理处接收应答包
                    writer.write(REP)   #将应答包转发给客户端
                    await writer.drain()
                    print(f'connect success with 127.0.0.5 and 1085 !')
                except (TimeoutError, ConnectionRefusedError) as e:
                    print(f'connect failed with 127.0.0.5 and 1085 !')
                    print(f'{repr(e)}')
                    writer.close()
                    return
                #并发转发数据包
                await asyncio.gather(transport(reader, dswriter, '127.0.0.1'),
transport(dsreader, writer, '127.0.0.1'), count_width(), readfile(),
writefile())
            if header[3] == 3: #域名 X'03'
                try:
                    dsreader, dswriter = await
asyncio.open_connection('127.0.0.5', 1085)

                    dswriter.write(usermas)
                    await writer.drain()
                    VER = await dsreader.read(50000)
                    if VER[0] != 1:
                        writer.close()

                    dswriter.write(data)   #将数据包发给remoteproxy
                    await dswriter.drain()
                    REP = await dsreader.read(1024) #从远端代理处接收应答包
                    writer.write(REP)   #将应答包转发给客户端
                    await writer.drain()

                    print(f'connect success with 127.0.0.5 and 1085 in SOCKS5!')
                except (TimeoutError, ConnectionRefusedError) as e:
```

```
                    print(f'connect failed with 127.0.0.5 and 1085 in SOCKS5!')
                    print(f'{repr(e)}')
                    writer.close()
                    return
                #并发转发数据包
                await asyncio.gather(transport(reader, dswriter, '127.0.0.5'),
transport(dsreader, writer, '127.0.0.5'), count_width(), readfile(),
writefile())
    # http协议
    elif httpdata[0] == 'CONNECT': #只处理http的connect包，其余包暂时不处理
        try:
            dsreader, dswriter = await asyncio.open_connection('127.0.0.5',
1085)  # 与remoteproxy建立连接

            #账号、密码认证
            dswriter.write(usermas)
            await dswriter.drain()

            VER = await dsreader.read(50000)
            if VER[0] != 1:
                writer.close()

            dswriter.write(data)   #将数据包发给remoteproxy
            await dswriter.drain()
            REP = await dsreader.read(1024) #从远端代理处接收应答包
            writer.write(REP)    #将应答包转发给客户端
            await writer.drain()
            print(f'connect success with 127.0.0.5 and 1085 in HTTP!')
        except (TimeoutError, ConnectionRefusedError) as e:
            print(f'connect failed with 127.0.0.1 and 1085 in HTTP!')
            print(f'{repr(e)}')
            writer.close()

            return
        await asyncio.gather(transport(reader, dswriter, '127.0.0.5'),
transport(dsreader, writer, '127.0.0.5'), count_width(), readfile(),
writefile())
    else:
        print("we can't handle this type of request")
        writer.close()
        return


async def main():
    print(args.username, args.password)
    if args.consolePort:
        ws_server = await websockets.serve(localConsole, '127.0.0.1',
args.consolePort)
        logging.info(f'CONSOLE LISTEN {ws_server.sockets[0].getsockname()}')

    #asyncio.create_task(calcBandwidth())

    server_1 = await asyncio.start_server(handle_echo, '127.0.0.1', 1080)
    #print('1')

    addr = server_1.sockets[0].getsockname()
    print(f'Serving on {addr}')
```

```
    async with server_1:
        await server_1.serve_forever()

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("-u", "--username", help="please input your username")
    parser.add_argument("-p", "--password", help="please input your password")
    parser.add_argument("-c", "--consolePort", help="please input the
consolePort", default=0 )
    args = parser.parse_args()

    asyncio.run(main())
```

remoteproxy.py

```
import asyncio
import aiosqlite
import argparse
import operator
import time
import sys
from struct import unpack,pack

capacity = 10000000 #桶容量
#Rate = 0  #速率
class TokenBucket:
    '''
        令牌桶算法：出
            令牌出桶速率恒定，当桶中无剩余令牌，则不能取出
    '''
    def __init__(self, rate, capacity):
        '''
            rate:出桶速率
            volume: 最大容积
            current：桶中现有令牌
            times：计时
        '''
        self._rate = rate
        self._capacity = capacity
        self._current_amount = 0
        self._last_consume_time = int(time.time())
        self.tokenSemaphore = asyncio.BoundedSemaphore(1)

    async def consume(self, token_amount):
        #上锁
        await self.tokenSemaphore.acquire()
        # 从上次发送到这次发送，新取出的令牌数量
        increment = (int(time.time()) - self._last_consume_time) * self._rate
        # 桶中当前剩余令牌数量
        self._current_amount = min(increment + self._current_amount,
self._capacity)
        print(self._current_amount, increment, token_amount,
self._last_consume_time, int(time.time()))
        # 如果需要的令牌超过剩余的令牌，则不能发送数据
        if token_amount > self._current_amount:
            self.tokenSemaphore.release()
```

```
            return False
        self._last_consume_time = int(time.time())
        # 可以取出令牌，取出后桶中剩余令牌数量
        self._current_amount -= token_amount
        #解锁
        self.tokenSemaphore.release()
        return True

async def transport(reader, writer, addr, TB):
    while reader.at_eof:
        try:    # 从reader接收外部报文
            data = await reader.read(1000)
            data_len = sys.getsizeof(data)   #数据大小
            if not data:
                writer.close()
                break
        except (ConnectionAbortedError, ConnectionRefusedError) as e:
            writer.close()
            print(f'{addr}异常退出，{repr(e)}')
            break

        try:    # 向writer转发报文
            #指令桶
            if TB.consume(data_len):
                writer.write(data)
                await writer.drain()
        except (ConnectionAbortedError, ConnectionRefusedError) as e:
            writer.close()
            print(f'{addr}异常退出，{repr(e)}')
            break
    print(f'{addr}正常退出')

async def handle_echo(reader, writer):
    data = await reader.read(1000) #将信息读至EOF
    message = data.decode()
    usermessage1 = message.split(' ')
    usermessage = tuple(usermessage1)
    flag = 0

    async with aiosqlite.connect('user.db') as db:
        async with db.execute("SELECT username, password, rate FROM user_table")
as uap:
            async for u in uap:
                if usermessage[0]==u[0] and usermessage[1]==u[1] and flag==0:
                    Rate = u[2] #速率
                    flag = 1
                    print("认证成功!!!")
                    writer.write(b'\x01')
                    await writer.drain()

    if flag==0:
        writer.close()
        print("认证失败!!!")

    data = await reader.read(50000) #将信息读至EOF
    if data[0] == ord('C'):
        data1 = data.decode()
        httpdata = data1.split(' ')
```

```
    # 判断协议的类型(socks5协议或者http协议)
    # socks5协议
    if data[0] == 5:
        # 解读客户端发来的信息包
        peername = writer.get_extra_info('peername')
        print(f"Create TCP connection with {peername!r}")
        header = unpack('!BBBB', data[:4])  # 读取前面四个基本信息 VER/CMD/RSV/ATYP
        # 判断socks版本号和需要实现的功能(本代理服务器只支持socks5协议，且只能实现connect功
能)
        if header[0] == 5 and header[1] == 1:
            if header[3] == 1:  #IPv4 X'01'
                ip = '.'.join([str(i) for i in unpack('!BBBB', data[4:8])]) #获取
IP地址，IPv4的地址4个字节
                port = unpack('!H', data[8:10])[0]  #获取端口号，2字节
                print(f'ip:{ip}, port{port}')
                try:
                    dsreader, dswriter = await asyncio.open_connection(ip, port)
                    writer.write(b'\x05\x00\x00' + data[3:11])  #将信息重新打包发回
给localproxy，告知客户端代理服务器与目标服务器连接成功
                    await writer.drain()
                    print(f'connect success with {ip} and {port}!')
                except (TimeoutError, ConnectionRefusedError) as e:
                    print(f'connect failed with{ip} and {port}!')
                    print(f'{repr(e)}')
                    writer.close()
                    return
                #指令桶
                TB = TokenBucket(Rate, capacity)
                #并发转发数据包
                await asyncio.gather(transport(reader, dswriter, ip, TB),
transport(dsreader, writer, ip, TB))
            if header[3] == 3: #域名 X'03'
                Hostlen = unpack('!B', data[4:5])[0] #域名长度
                Host = data[5: Hostlen+5].decode('utf-8') #解析域名
                port = unpack('!H', data[5+Hostlen: Hostlen+7])[0] #获取端口号，2字
节
                print(f'Hostlen:{Hostlen}, Host:{Host}, port:{port}')
                try:
                    dsreader, dswriter = await asyncio.open_connection(Host,
port)
                    writer.write(b'\x05\x00\x00'+data[3: Hostlen+7])  #将信息重新打
包发回给locoalproxy，告知客户端：代理服务器与目标服务器连接成功
                    await writer.drain()
                    print(f'connect success with {Host} and {port} in SOCKS5!')
                except (TimeoutError, ConnectionRefusedError) as e:

                    print(f'connect failed with{Host} and {port} in SOCKS5!')
                    print(f'{repr(e)}')
                    writer.close()
                    return

                #指令桶
                TB = TokenBucket(Rate, capacity)
                #并发转发数据包
                await asyncio.gather(transport(reader, dswriter, Host, TB),
transport(dsreader, writer, Host, TB))
    # http协议
    elif httpdata[0] == 'CONNECT': #只处理http的connect包，其余包暂时不处理
```

```
        try:
            httpdata1 = httpdata[1].split(':')
            Host = httpdata1[0] # 获取主机地址
            port = httpdata1[1] # 获取端口号
            dsreader, dswriter = await asyncio.open_connection(Host, port)  # 与
目的服务器建立连接
            writer.write(b'HTTP/1.1 200 Connection established\r\n\r\n')    # 将
应答包发给localproxy，告知客户端：代理服务器与目标服务器连接成功
            await writer.drain()
            print(f'connect success with {Host} and {port} in HTTP!')
        except (TimeoutError, ConnectionRefusedError) as e:
            print(f'connect failed with {Host} and {port} in HTTP!')
            print(f'{repr(e)}')
            writer.close()
            return

        #指令桶
        TB = TokenBucket(Rate, capacity)
        #并发转发数据包
        await asyncio.gather(transport(reader, dswriter, Host, TB),
transport(dsreader, writer, Host, TB))
    else:
        print("we can't handle this type of request")
        writer.close()
        return


async def main():

    async with aiosqlite.connect('user.db') as db:
        #await db.execute("DROP TABLE user_table") #删除表
        await db.execute("CREATE TABLE if not exists user_table ( username
VARCHAR(50) primary key, password VARCHAR(50), rate INT)")
        #await db.execute("DELETE FROM user_table")#删除表中的数据
        #await db.execute("INSERT INTO user_table ( username, password) VALUES
('R52125', 'R100214')")
        #await db.execute("INSERT INTO user_table ( username, password) VALUES
('F444627549', 'XJBRCBR')")

        await db.execute("DELETE FROM user_table where username = 1")
        await db.execute("INSERT INTO user_table (username, password, rate)
VALUES (?, ?, ?)" ,(args.username, args.password, args.rate))
        await db.execute("DELETE FROM user_table where username = 1")
        await db.commit()

        async with db.execute("SELECT username, password, rate FROM user_table")
as uap:
            async for u in uap:
                print(f'f1={u[0]} f2={u[1]} f3={u[2]}')

    server = await asyncio.start_server(handle_echo, '127.0.0.5', 1085)

    addr = server.sockets[0].getsockname()
    print(f'Serving on {addr}')

    async with server:
        await server.serve_forever()
```

```
parser = argparse.ArgumentParser()
parser.add_argument("-u", "--username", help="please input the username you want
to add, but it can't be '1'.", default=1)
parser.add_argument("-p", "--password", help="please input the password you want
to add, but it can't be '1'.", default=1)
#parser.add_argument("-pp", "--ppassword", help="please define the password
again")
parser.add_argument("-r", "--rate", help="please input rate", default=0)
args = parser.parse_args()

asyncio.run(main())
```

data_main.py

```
from data_extract import *
from data_preparation import *

if __name__ == '__main__':
    print("加载文件...")
    raw_text_list = get_text_list()
    print("准备完成.")
    print("一共有",len(raw_text_list),"篇文档")
    dic={"1":"网址1",
         "2": "网址2",
         "3": "http版本",
         }

    while True:
        print("可供选择抽取信息如下: \n"
              "1:网址1 2:网址2 3:http版本\n"
              "请输入抽取信息序号,或输入0退出")
        extracNum = input("> ")

        if extracNum == '0':
            print('Bye')
            exit(0)
        print("检测到的结果: ", dic[extracNum])
        result = find(raw_text_list,int(extracNum))
```

data_preparation.py

```
import os
import re

path = 'data'
files = os.listdir(path)


def get_text_list():
    text_list = []

    for i in files:
        f = open(path + '\\' + "write_data1.txt", encoding='utf-8')
```

```
        text = f.read()
        text_list.append(text)
    return text_list
```

data_extract.py

```
import re


def extract_1(data):

    res=re.findall(r'[A-Za-z]+\.[A-Za-z]+',data,0)

    return res


def extract_3(data):

    res=re.findall(r'http/[^\\]*',data,0)

    return res

def extract_2(data):

    res=re.findall(r'[A-Za-z]+\.[A-Za-z]+\.[A-Za-z]+',data,0)

    return res

def extract_4(data):

    res=re.findall(r'',data,0)

    return res

def extract_5(data):

    res=re.findall(r'',data,0)

    return res

def extract_6(data):

    res=re.findall(r'',data,0)

    return res


def extract_7(data):

    res=re.findall(r'[A-Za-z0-9]+\.[A-Za-z0-9]+\.[A-Za-z0-9]+',data,0)

    return res




def choose_part(choice,temp):
```

```
    if choice == 1:
        part = extract_1(temp)
    elif choice == 2:
        part = extract_2(temp)
    elif choice == 3:
        part = extract_3(temp)
    elif choice == 4:
        part = extract_4(temp)
    elif choice == 5:
        part = extract_5(temp)
    elif choice == 6:
        part = extract_6(temp)
    elif choice == 7:
        part = extract_7(temp)
    return part


def find(datas,choice):
    check(datas, choice)
    data_len=len(datas)
    for i in range(0,data_len):
        temp=datas[i]
        part=choose_part(choice,temp)
        if(standard(part)!="NULL"):
            print("第",i,"个文件：  ",standard(part))
    check(datas, choice)

def check(datas,choice):
    if choice >7:
        return
    data_len = len(datas)
    if_exist=[]
    for i in range(0, data_len):
        temp = datas[i]
        part=choose_part(choice,temp)
        res=standard(part)
        if res != "NULL":
            if_exist.append(i)
    print("在文件",if_exist)
    print("中存在，共",len(if_exist),"个文件")


def standard(sentence):
    temp=str(sentence)
    temp=temp.replace('(',"")
    temp = temp.replace(')', "")
    temp = temp.replace('[', "")
    temp = temp.replace(']', "")
    temp = temp.replace('\'', "")
    if(len(temp)<1):
        temp="NULL"
    return temp
```