МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ФАКУЛЬТЕТ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«Обработка и тарификация трафика NetFlow»

Дисциплина: Управление мобильными устройствами

Вариант 4

Выполнил: студент

группы N3351

Головко Игорь Никитич

Проверил: инженер ФБИТ,

Университет ИТМО,

Федоров Иван Романович

Санкт-Петербург
2020 г.

Цель работы

Реализовать правило тарификации для услуг типа “Интернет” по количеству переданных байт. Протарифицировать абонента с IP-адресом 192.168.250.59 с коэффициентом k: 0,5руб/Мб до достижения 500Мб, далее 1руб/Мб. Построить график зависимости объёма от времени.

Средства реализации

Для выполнения работы был использован язык программирования Python в среде PyCharm. Данный выбор обоснован широкой распространённостью и мультиплатформенностью языка, а также удобным доступом к консоли, что тяжело осуществимо при использовании стандартного IDLE. Выбранный язык программирования позволяет использовать эффективные высокоуровневые структуры данных, простой в изучении и распространяется без ограничений.

Для построения графиков использовался модуль matplotlib. Принцип построения – зависимость количества байт, переданного через равные промежутки, от пройденного времени с первой хронологически записи в файле. Результат сохраняется в файле png.

Исходный код

1) lab2_graph.py – построение графика

```
import matplotlib.pyplot as plt
import datetime
import getopt
import sys

argv = sys.argv[1:]
opts, args = getopt.getopt(argv, 'x:y:')  # passing args


def main():
    with open(args[0], 'r', newline='') as infile:
        file_content = infile.read().split('\n')
        del file_content[-5:]  # removing last descriptive lines
        file_content.pop(0)  # removing first description line
        # ordered_tmp = sorted(tmp_dict.items(), key=lambda x: x[0])
        tmp_list = [line.split() for line in file_content]
        file_content = sorted(tmp_list, key=lambda x:
datetime.datetime.strptime(x[0]+" "+x[1], "%Y-%m-%d %H:%M:%S.%f"))
        first_row = file_content[0]
        first_datetime = datetime.datetime.strptime(first_row[0]+" "+first_row[1],
"%Y-%m-%d %H:%M:%S.%f")
        for last_row in file_content:
            pass
        last_datetime = datetime.datetime.strptime(last_row[0]+" "+last_row[1], "%Y-
```

```
%m-%d %H:%M:%S.%f")
        # first_datetime and last_datetime are class datetime.datetime
        deltatime = (last_datetime-first_datetime).total_seconds()
        count = datetime.timedelta(minutes=5)
        bytes = 0
        x_time = []  # seconds since first datetime
        y_bytes = []  # traffic intensity during timedelta
        for row in file_content:
            row_datetime = datetime.datetime.strptime(row[0]+" "+row[1], "%Y-%m-%d
%H:%M:%S.%f")
            if row_datetime < count+first_datetime:
                if row[12] == "M":
                    bytes += float(row[11]) * 1024 * 1024
                else:
                    bytes += float(row[11])
            else:
                x_time.append(count.total_seconds()/60)
                count += datetime.timedelta(minutes=5)
                y_bytes.append(bytes/1024)
                bytes = 0
                if row[12] == "M":
                    bytes += float(row[11]) * 1024 * 1024
                else:
                    bytes += float(row[11])
        x_time.append(count.total_seconds()/60)
        y_bytes.append(bytes/1024)
        plt.xlabel("Time from beginning in minutes")
        plt.ylabel("Traffic (Kb)")
        plt.title("Data transition per every 5 minutes")
        plt.plot(x_time, y_bytes)  # build graph
        plt.savefig("Graph.png")


if __name__ == '__main__':  # will not execute whole program if imported
    main()
```

2) lab2.py – тарификация абонента

```
import getopt
import sys

argv = sys.argv[1:]
opts, args = getopt.getopt(argv, 'x:y:')  # passing args


def main():  # operate with args
    with open(args[0], 'r', newline='') as infile:
        money = parse(infile)
        print(money)


def parse(infile):
    subscriber = "192.168.250.59"
    bytes = 0
    file_content = infile.read().split('\n')
    del file_content[-5:]  # removing last descriptive lines
    for index, row in enumerate(file_content):
        if index == 0:  # skip first description line
            continue
        row = row.split()
```

```
            """
            file structure:
            row[0] is date as year-month-day
            row[1] is time as hours:mins:secs.msecs
            row[2] is Event
            row[3] is XEvent
            row[4] is protocol
            row[5] is source ip addr
            row[6] is ->
            row[7] is destination ip addr
            row[8] is xip addr
            row[9] is ->
            row[10] is xip addr
            row[11] is bytes
            row[12] is 0 OR M if Mbytes in 11
            row[13] is 0 if 12 is M
            """
            if subscriber in row[5]:
                if row[12] == "M":
                    bytes += float(row[11]) * 1024 * 1024
                else:
                    bytes += float(row[11])
            if subscriber in row[7]:
                if row[12] == "M":
                    bytes += float(row[11]) * 1024 * 1024
                else:
                    bytes += float(row[11])
        return tariffing(bytes)


    def tariffing(bytes):  # module to import
        if bytes < 500*1024*1024:
            if bytes > 500*1024:
                retval = 500*0.5 + (bytes-500*1024) / 1024
            else:
                retval = bytes / 1024 * 0.5
        else:
            retval = 500*0.5 + (bytes-500*1024*1024) / 1024 / 1024
        return int(retval)


    if __name__ == '__main__':  # will not execute whole program if only tariffing is
    imported
        main()
```

Выводы

Биллинговые системы способны автоматически собирать и обрабатывать трафик NetFlow. Для тарификации каждого отдельного клиента достаточно осуществить парсинг базы данных и провести расчёты по тарифному плану. Обработка трафика также может представлять графическую интерпретацию обезличенных данных, которую возможно использовать для анализа Big Data. Всё это осуществимо в виде отдельных модулей, импортирование которых позволяет создать крупные и гибкие для настройки автоматизированные системы.