Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет Безопасности Информационных Технологий

Дисциплина
«Управление мобильными устройствами»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
«Обработка и тарификация трафика NetFlow»
Вариант 5

Выполнила:
Студентка группы N3350
Шкарева Алена Дмитриевна

Проверил:
доцент ФБИТ,
Федоров Иван Романович

Санкт-Петербург
2020

Цель работы:

Реализация правил тарификации трафика NetFlow.

Задачи:

Сформировать файл для тарификации;
Провести тарификацию
Построить график зависимости объема трафика от времени.

Ход работы:

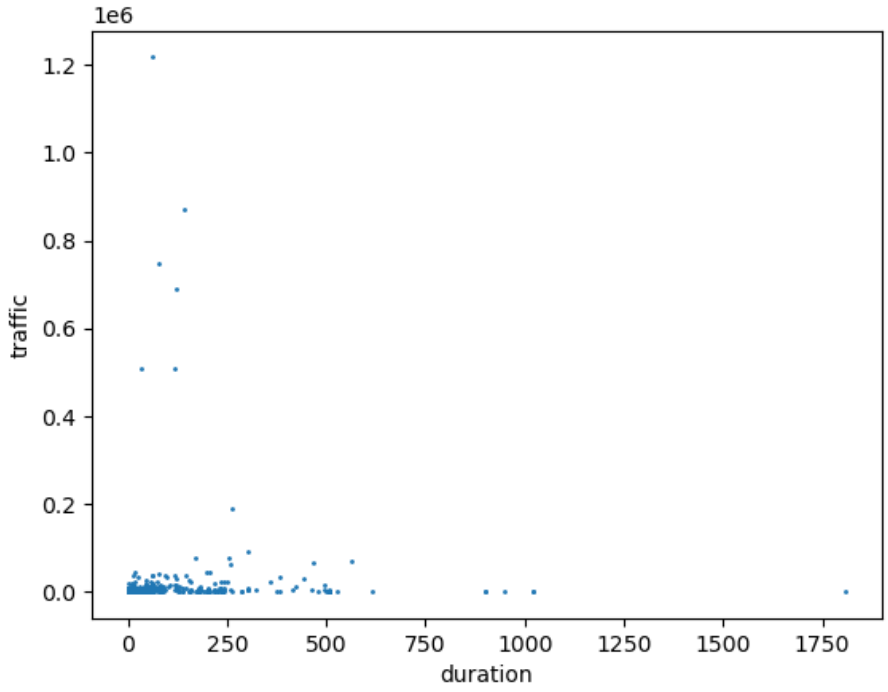
Для приведения файла с трафиком NetFlow к удобному формату использовалась утилита nfdump:
nfdump -r nfcapd.202002251200 -o extended -o csv > file.csv
Из файла были убраны последние строки summary.

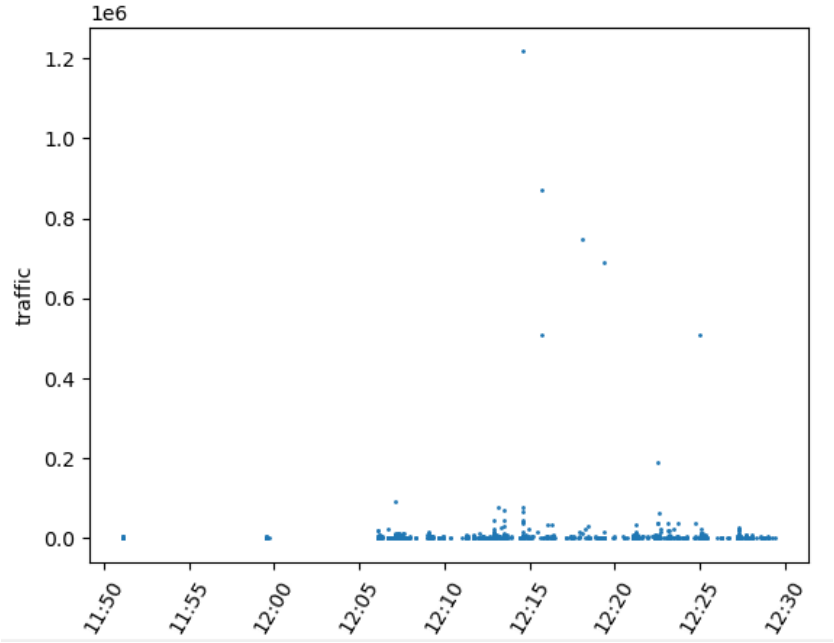
Реализация:

Для реализации программы был выбран язык Python.
Из сторонних библиотек используются: pandas, matplotlib.
Выполнение программы:

```
C:\Users\Gammilen\projects\labs\умy\2>python -m script
Оплата услуги "Интернет": 7027.8 руб.
График связи объема трафика и длительности обращения
График связи объема трафика и времени обращения
```

Полученные графики





Вывод:

В результате выполнения лабораторной работы был изучен принцип работы протокола NetFlow.

Исходный код:

https://github.com/gammilen/mobile_device_management

script.py

```
import pandas as pd
from plots import *
import os

class Charging:
    def __init__(self, ip):
        self._payment = 0
        self.init_df(ip)
        self.base_rule = None
        self.add_rule = None

    def init_df(self, abonent_ip):
        #load file
        df =
pd.read_csv(os.path.join(os.path.dirname(os.path.abspath(__file__)),
"file.csv"))
        #filter columns
        df = df[["ts", "te", "td", "sa", "ibyt"]]
        #filter source address
        df = df[df["sa"]==abonent_ip]
        self.df = df

    def init_rules(self, base, add):
        #base rule
        self.base_rule = dict(k=base)
```

```
        #add rule
        add = add[0]
        self.add_rule = dict(k=add[0], limit=add[1])

    def _get_amount(self):
        return self.df["ibyt"].sum()/1024/1024

    def calculate(self):
        a = self._get_amount()
        #apply addition rule
        if (self.add_rule):
            tmp = a - self.add_rule["limit"]
            if tmp <= 0:
                #change conditions (Mb to Kb(in Mb)
                a *= 1024
                tmp = a - self.add_rule["limit"]
                if tmp <= 0:
                    self._payment += a * self.add_rule["k"]
                    return
            a = tmp
            self._payment += self.add_rule["limit"] * self.add_rule["k"]
        #apply base rule
        self._payment += a * self.base_rule["k"]

    @property
    def payment(self):
        return round(self._payment, 2)



def prepare_charging():
    abonent_ip = "192.168.250.59"

    c = Charging(abonent_ip)
    c.init_rules(1, [(0, 1000)])
    return c

def get_payment(charging):
    charging.calculate()
    return charging.payment

def main():
    c = prepare_charging()
    print("Оплата услуги \"Интернет\":", get_payment(c), "руб.")

    print("График связи объема трафика и длительности обращения")
    make_duration_traffic_plot(c.df)
    print("График связи объема трафика и времени обращения")
    make_time_traffic_plot(c.df)

if __name__ == "__main__":
    main()
```

plots.py

```
import pandas as pd
import matplotlib.pyplot as plt

def make_time_traffic_plot(df):
      idx = pd.to_datetime(df["ts"])
      df2 = pd.DataFrame(list(df["ibyt"]), index=idx)

      plt.xlabel("time")
```

```
        plt.ylabel("traffic")
        labels = [pd.Timestamp('2020-02-25 11:50:00'), pd.Timestamp('2020-02-25
11:55:00'), pd.Timestamp('2020-02-25 12:00:00'), pd.Timestamp('2020-02-25
12:05:00'), pd.Timestamp('2020-02-25 12:10:00'), pd.Timestamp('2020-02-25
12:15:00'), pd.Timestamp('2020-02-25 12:20:00'), pd.Timestamp('2020-02-25
12:25:00'), pd.Timestamp('2020-02-25 12:30:00')]
        plt.xticks(labels, [str(i)[11:16] for i in labels], rotation=60)
        plt.scatter(df2.index, df2, s=1)
        plt.show()

def make_duration_traffic_plot(df):
        plt.xlabel("duration")
        plt.ylabel("traffic")
        plt.scatter(df["td"], df["ibyt"], s=1)
        plt.show()
```