Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет Безопасных Информационных Технологий

Управление мобильными устройствами

Лабораторная работа №2

Выполнил:

студент группы N3348

Колесников Н. Д.

Проверил:

Федоров И. Р.

_____

Санкт-Петербург

2020

Цель работы

В данной работе необходимо обработать трафик NetFlow v5 из файла nfcapd.202002251200. Для чего изначально требуется привести данный файл в читабельный вид (проще всего это сделать с помощью утилиты nfdump), после чего нужно сформировать собственный файл для тарификации любого формата, с которым удобно работать (в соответствии с вариантом работы), после чего необходимо построить график зависимости объема трафика от времени (любым удобным образом) и наконец требуется протарифицировать трафик в соответствии с вариантом задания.

Средство реализации

Для реализации мною был выбран язык программирования Python 3.x , т. к. я считаю что для выполнения данной задачи, он является наиболее подходящим вариантом (Так как присутствует большое количество модулей, используемых для работы с данными и их визуализации).

Для работы была выбрана среда Jupyter Notebook (также известный как IPython Botebook), которая крайне удобна в случае, если ты используешь Python (Так как весь код разделён на секции, что 1. упрощает процесс отладки программы 2. позволяет при необходимости выводить содержимое данных в любом месте программы).

Исходный код

https://github.com/LordKzoth/MOBILE_MANAGEMENT/blob/master/LW_2/LW_2.py

Ход работы (Вариант 6)

Задание:

Вариант 6

Протарифицировать абонента с IP-адресом 192.168.250.1 с коэффициентом k: 0,5руб/Мб (ЗАМЕНА: 0,5руб/Кб) первые 500Мб (ЗАМЕНА: 500Кб), после каждых последующих 500Мб (ЗАМЕНА: 500Кб) k увеличивается на 0,5руб

Данные замены в задание были произведены согласно примечанию №2. Так как в последующем было выяснено, что общий трафик для данного IP приблизительно равен 6 Мб, что значительно меньше 500 Мб.

Ход работы:

Изначально с помощью команды «nfdump -r nfcapd.202002251200» был сформирован файл «nfcapd.txt».

Следующим шагом этот файл необходимо было обработать таким образом, чтобы после этого его можно было бы прочитать с помощью модуля pandas.

Обработка файла

In [2]:
```
text = []
with open('nfcapd.txt', 'r', encoding='utf_8') as f:
    text = f.readlines()
```

In [3]:
```
spaces = [' '*n for n in range(20, 2, -1)]
```

In [4]:
```
for i in range(len(text)):
    text[i] = text[i].replace('->', '')

words = ['INVALID', 'XEvent', 'Ignore']
for word in words:
    for i in range(len(text)):
        text[i] = text[i].replace(word, ' ' + word + ' ')

for space in spaces:
    for i in range(len(text)):
        text[i] = text[i].replace(space, '  ')

for i in range(len(text)):
    text[i] = text[i].replace('  ', ',')

text[:10]
```

Описание данного блока:

1) Содержимое файла «nfcapd.txt» записывается в переменную text;
2) Далее это содержимое обрабатывается следующим образом:
    a. Очищается от символов «–>»;
    b. Добавляются дополнительные пробелы, там где располагаются одинарные (это необходимо для последующей очистки);
    c. Текст очищается от множественных пробелов, которые заменяются на двойные (это необходимо для того, чтобы можно было понять где стоит обыкновенный пробел, а где стоит пробел-разделитель);
    d. Двойные пробелы заменяются на запятые.

Следующим шагом отредактированный текст записывается в новый файл «new_nfcapd.txt», который на следующем шаге обрабатывается с помощью метода pandas.read_csv(). Результат записывается в переменную df.

In [5]:
```
with open('new_nfcapd.txt', 'w', encoding='utf_8') as f:
    f.writelines(text)
```

In [6]:
```
df = pd.read_csv('new_nfcapd.txt', sep=',')
df
```

Out[6]:

	Date first seen	Event	XEvent	Proto	Src IP Addr:Port	Dst IP Addr:Port	X-Src IP Addr:Port	X-Dst IP Addr:Port	In Byte	Out Byte
0	2020-02-25 11:21:06.190	INVALID	Ignore	TCP	192.168.250.3:80	23.226.231.226:3682	0.0.0.0:0	0.0.0.0:0	572	0.0
1	2020-02-25 11:28:30.860	INVALID	Ignore	TCP	192.168.250.50:61137	40.114.211.99:443	0.0.0.0:0	0.0.0.0:0	2241	0.0
2	2020-02-25 11:29:30.210	INVALID	Ignore	TCP	192.168.250.3:80	23.226.231.226:28857	0.0.0.0:0	0.0.0.0:0	308	0.0
3	2020-02-25 11:30:01.860	INVALID	Ignore	UDP	192.168.250.62:58474	192.168.250.1:123	0.0.0.0:0	0.0.0.0:0	152	0.0
4	2020-02-25 11:30:01.860	INVALID	Ignore	UDP	192.168.250.1:123	192.168.250.62:58474	0.0.0.0:0	0.0.0.0:0	152	0.0

Следующим шагом из всех записей, были выбраны только те, в которых фигурирует IP=192.168.250.1

```
In [7]: total_traffic = df[(df['Src IP Addr:Port'].str.contains('192.168.250.1'))
                         | (df['Dst IP Addr:Port'].str.contains('192.168.250.1'))].reset_index(drop=True).copy()
        total_traffic
```

Out[7]:

	Date first seen	Event	XEvent	Proto	Src IP Addr:Port	Dst IP Addr:Port	X-Src IP Addr:Port	X-Dst IP Addr:Port	In Byte	Out Byte
0	2020-02-25 11:30:01.860	INVALID	Ignore	UDP	192.168.250.62:58474	192.168.250.1:123	0.0.0.0:0	0.0.0.0:0	152	0.0
1	2020-02-25 11:30:01.860	INVALID	Ignore	UDP	192.168.250.1:123	192.168.250.62:58474	0.0.0.0:0	0.0.0.0:0	152	0.0
2	2020-02-25 11:30:02.530	INVALID	Ignore	UDP	192.168.250.50:62595	192.168.250.1:53	0.0.0.0:0	0.0.0.0:0	132	0.0
3	2020-02-25 11:30:02.540	INVALID	Ignore	UDP	192.168.250.1:53	192.168.250.50:62595	0.0.0.0:0	0.0.0.0:0	450	0.0
4	2020-02-25 11:30:02.700	INVALID	Ignore	UDP	192.168.250.50:60512	192.168.250.1:53	0.0.0.0:0	0.0.0.0:0	126	0.0

Таких записей оказалось 7064 штук.

На следующем шаге значения из столбика «Date first seen» были приведены к типу numpy.datetime, а также значения из столбика «In Byte», которые были записаны в виде «x.x M», были переведены в байты.

В завершение этого шага все значения из столбика «In Byte» были просуммированы.

```
In [8]: total_traffic['Date first seen'] = total_traffic['Date first seen'].apply(pd.to_datetime).copy()
```

```
In [9]: total_traffic['In Byte'] = total_traffic['In Byte'].apply(
            lambda x: float(x) if x[-1] != 'M' else float(x[:-1]) * 1024 * 1024).copy()
        result = sum(total_traffic['In Byte'].values)
```

Далее результат переводится в Кб.

```
In [10]: result = (result / 1024)
```

```
In [11]: result
```

```
Out[11]: 6190.453515625
```

Для отображения графика зависимости объёма данных от времени нам необходимо импортировать модуль matplotlib.

```
In [12]: import matplotlib.pyplot as plt
```

Для построения графика нам необходимо сгруппировать записи по времени (значения в столбике «In Byte» суммируются по группам).

```
In [13]: traffic_after_group = (total_traffic.groupby('Date first seen').aggregate(sum)).copy()
```

Далее строится график (Данные были сгруппированы по миллисекундам).

```
In [15]: plt.figure(figsize=(15, 10))
         plt.title('Миллисекунды')
         plt.plot(traffic_after_group.index, traffic_after_group['In Byte'].values)

         plt.show()
```

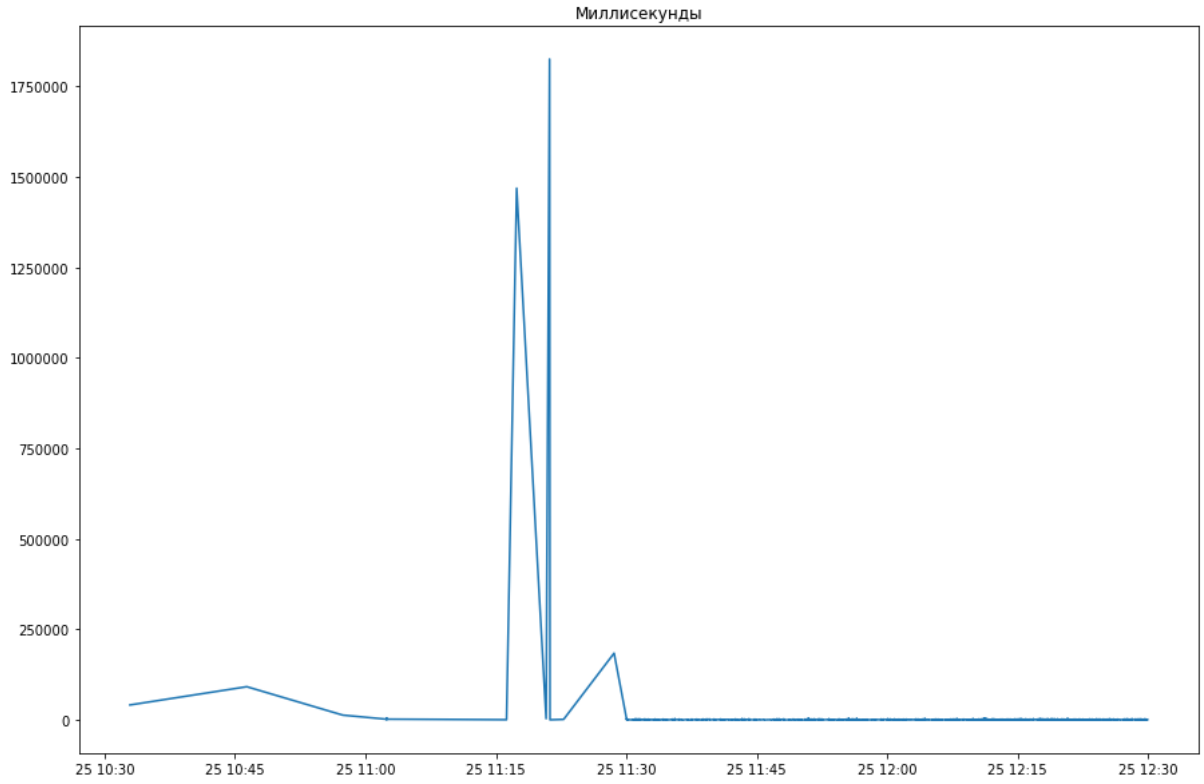

Рисунок 1 – График зависимости объёма трафика от времени (Миллисекунды)

Однако, на данном графике не сильно просматривается данная зависимость.

Следующим шагом было принято решение, сгруппировать данные по секундам и по минутам.

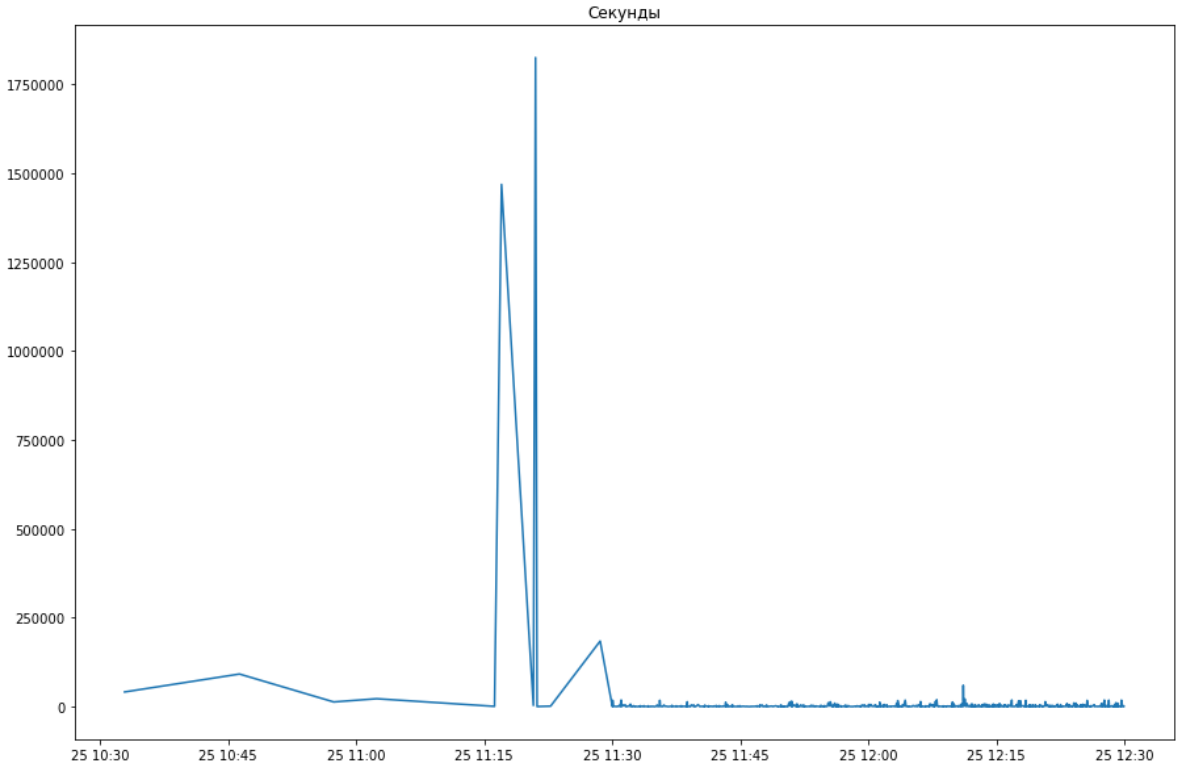

Рисунок 2 – График зависимости объёма трафика от времени (Секунды)

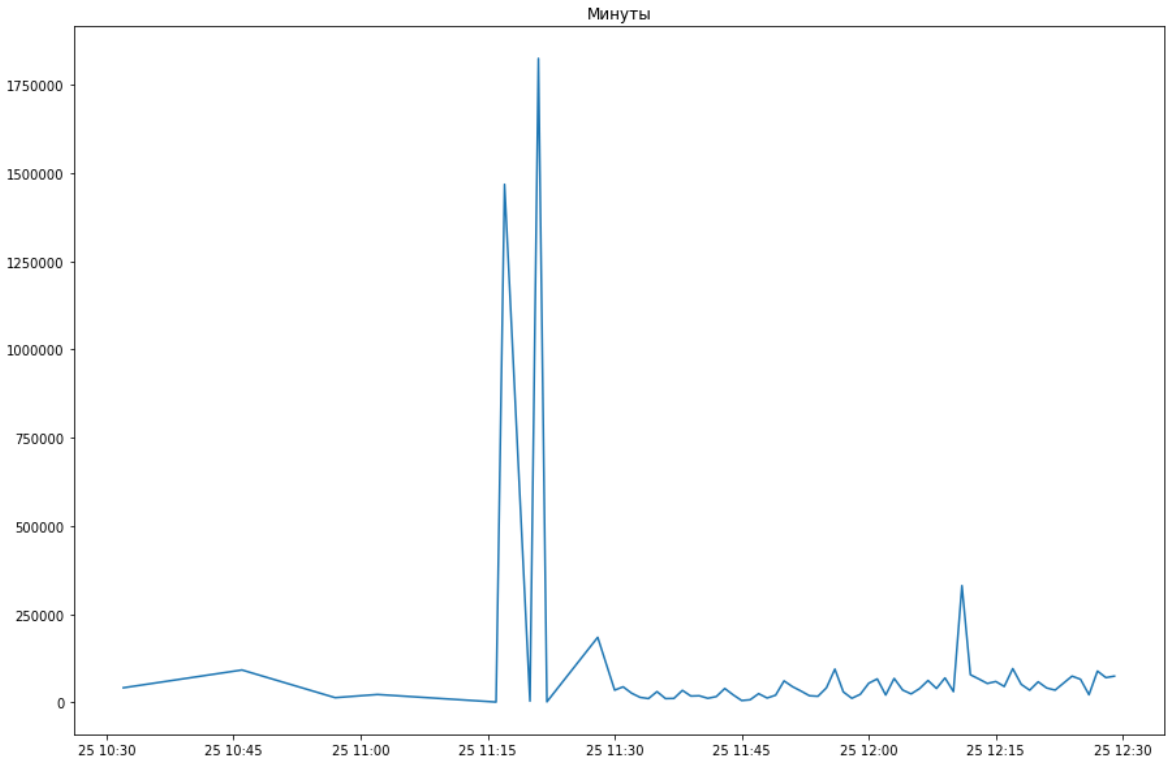

Рисунок 3 – График зависимости объёма трафика от времени (Минуты)

Наиболее четко зависимость просматривается на последнем графике, поэтому было принято решение использовать его в качестве основного.

На последнем шаге необходимо было произвести тарификацию согласно варианту.

In [26]:
```
total_price = 0
price_per_500 = 0.5

while result > 0:
    total_price += 500 * price_per_500 if result >= 500.0 else result * price_per_500

    price_per_500 += 0.5
    result -= 500

total_price = round(total_price, 2)
```

Итоговый результат

In [29]:
```
print('Итоговый результат: {}'.format(total_price))
```

```
Итоговый результат: 20737.95
```

Видим результат, округленный до сотых: **20737.95**

Вывод: в ходе выполнения данной лабораторной работы мною обследован предоставленный файл «nfcapd.202002251200», который был преобразован в файл «nfcapd.txt», который в свою очередь был преобразован в более удобный для анализа вид, который записан в итоговый файл «new_nfcapd.txt», после чего из него были извлечены необходимые данные. После чего было реализовано простейшее правило тарификации для услуг типа «Интернет» по общему объёму трафика.