Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

ФАКУЛЬТЕТ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Управление мобильными устройствами

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«Обработка и тарификация трафика NetFlow»

Вариант 10

Выполнил студент,

группы N3350 – Находкин Александр Михайлович

Подпись:

Проверил: доцент ФБИТ,

Университет ИТМО,

Федоров Иван Романович

Санкт-Петербург

2020

# Цель работы:

Обработка и тарификация трафика NetFlow

# Задачи:

1. Привести файл nfcapd в читабельный вид;

2. Сформировать собственную программу для тарификации;

3. Построить график зависимости объема трафика от времени;

4. Протарифицировать трафик в соответствии с вариантом задания.

# Реализация:

Программа была реализована на языке C++ из-за достаточных знаний этого языка для выполнения данной работы.

# Выполнение программы:

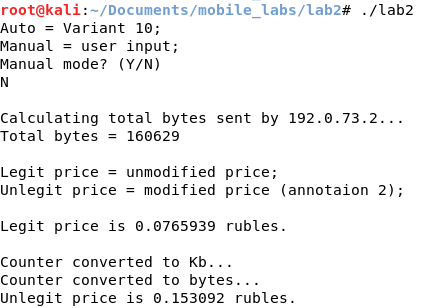
Используя утилиту nfdump, приводим файл в читаемый вид в формате csv:



Скомпилируем программу:



Запустим программу и проверим 10-й вариант:



Содержимое файла outputData.txt после завершения программы:

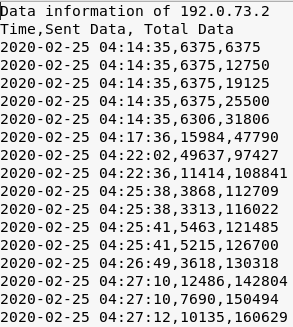
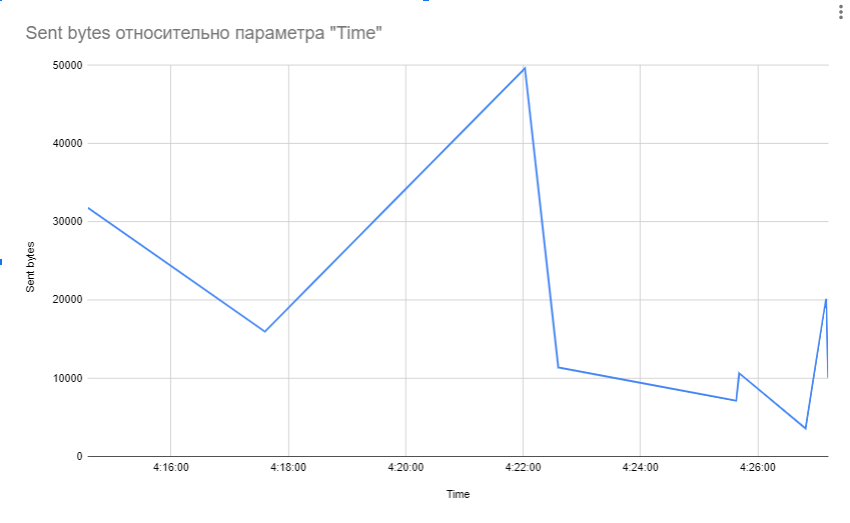


График:



# Выводы:

Во время выполнения данной работы были проанализированы различные ключи и форматы вывода информации утилиты nfdump, а также проанилизрован трафик nfcapd.

# Исходный код:

Исходный код можно найти на:

<https://github.com/larentoun/ITMO_Mobiles_N3350_NakhodkinAM/tree/master/lab2>

#include <iostream>

#include <string>

#include <sstream>

#include <fstream>

#include <vector>

#include <stdbool.h>

using namespace std;

void read\_data(float &totalBytes, string ipAddress)

{

//float totalBytes = 0; //Total price for subscriber

//float totalPrice = 0; //

//Open file

fstream fileToRead;

fileToRead.open("nfdumpFile.csv", ios::in);

if (!fileToRead)

{

cout << "Failed to open nfdumpFile.csv!" << endl;

exit(1);

}

//Create a vector which will be written info in

vector<string> row(48);

string line, word, temp;

//Read every row and compare it to the request

cout << "Calculating total bytes sent by " << ipAddress << "..." << endl;

ofstream outputFile;

outputFile.open("outputData.txt");

if (!outputFile.is\_open())

{

cout << "ERROR! read\_data can't read file! Exiting..." << endl;

exit(1);

}

outputFile << "Data information of " << ipAddress << endl;

outputFile << "Time,Sent Data, Total Data" << endl;

while (fileToRead >> temp)

{

if (temp[0] != '2')

{

continue;

}

row.clear();

getline(fileToRead, line);

stringstream s(line);

//Read every column data of a row and

//Store it in a string variable, 'word'

while (getline(s, word, ','))

{

row.push\_back(word);

}

//row[3] = source address

//row[4] = destination address

//row[12] = In byte

//row[14] = Out byte

//Change totalBytes and totalPrice if needed

if (row[3] == ipAddress)

{

totalBytes = totalBytes + stof(row[12]);

outputFile << row[1] << "," << stof(row[12]) << "," << totalBytes << endl;

}

}

cout << "Total bytes = " << totalBytes << endl;

fileToRead.close();

//return totalPrice;

}

void get\_price(float& totalBytes, float& totalPrice, float pricePerMbA, float pricePerMbB, float pricePerMbCounter, bool isLegit)

{

if (isLegit == true)

{

if (totalBytes > pricePerMbCounter \* 1024 \* 1024)

{

totalPrice = pricePerMbA \* (pricePerMbCounter) + pricePerMbB \* (totalBytes / 1024 / 1024 - pricePerMbCounter);

}

else

{

totalPrice = pricePerMbA \* (totalBytes / 1024 / 1024);

}

cout << "Legit price is " << totalPrice << " rubles." << endl;

}

else

{

if (totalBytes < pricePerMbCounter \* 1024 \* 1024) // Counter set to Mb

{

cout << "Counter converted to Kb..." << endl;

if (totalBytes < pricePerMbCounter \* 1024) // Counter set to Kb

{

cout << "Counter converted to bytes..." << endl;

if (totalBytes < pricePerMbCounter) // Counter set to b

{

totalPrice = pricePerMbA \* totalBytes / 1024 / 1024;

}

else

{

totalPrice = pricePerMbA \* (pricePerMbCounter) / 1024 / 1024 + pricePerMbB \* (totalBytes / 1024 / 1024 - pricePerMbCounter / 1024 / 1024);

}

}

else

{

totalPrice = pricePerMbA \* (pricePerMbCounter) / 1024 + pricePerMbB \* (totalBytes / 1024 / 1024 - pricePerMbCounter / 1024);

}

cout << "Unlegit price is " << totalPrice << " rubles." << endl;

}

else

{

cout << "Counter is unconverted..." << endl;

totalPrice = pricePerMbA \* (pricePerMbCounter)+pricePerMbB \* (totalBytes / 1024 / 1024 - pricePerMbCounter);

cout << "Unlegit price is legit and is " << totalPrice << " rubles." << endl;

}

}

}

int main()

{

float totalBytes = 0;

float totalPrice = 0;

string ipAddress;

float pricePerMbA;

float pricePerMbB;

float pricePerMbCounter;

string modeSelected;

cout << "Auto = Variant 10;" << endl;

cout << "Manual = user input;" << endl;

cout << "Manual mode? (Y/N)" << endl;

cin >> modeSelected;

cout << endl;

if (modeSelected == "Y")

{

cout << "Ip Address = ";

cin >> ipAddress;

cout << "Mb counter (0 if no counter) = ";

cin >> pricePerMbCounter;

if (pricePerMbCounter == 0)

{

cout << "Price per Mb = ";

cin >> pricePerMbA;

pricePerMbB = pricePerMbA;

}

else

{

cout << "Price per Mb before counter = ";

cin >> pricePerMbA;

cout << "Price per Mb after counter = ";

cin >> pricePerMbB;

}

cout << endl;

cout << "Legit price = unmodified price;" << endl;

cout << "Unlegit price = modified price (annotaion 2);" << endl;

cout << "Legit mode? (Y/N)" << endl;

cin >> modeSelected;

cout << endl;

if (modeSelected == "Y")

{

read\_data(totalBytes, ipAddress);

get\_price(totalBytes, totalPrice, pricePerMbA, pricePerMbB, pricePerMbCounter, true);

}

else

{

read\_data(totalBytes, ipAddress);

get\_price(totalBytes, totalPrice, pricePerMbA, pricePerMbB, pricePerMbCounter, false);

}

return 0;

}

else

{

//Variant 10

ipAddress = "192.0.73.2";

pricePerMbA = 0.5;

pricePerMbB = 1;

pricePerMbCounter = 200; // Change price after 200 Mb

read\_data(totalBytes, ipAddress);

cout << endl;

cout << "Legit price = unmodified price;" << endl;

cout << "Unlegit price = modified price (annotaion 2);" << endl;

cout << endl;

get\_price(totalBytes, totalPrice, pricePerMbA, pricePerMbB, pricePerMbCounter, true);

cout << endl;

get\_price(totalBytes, totalPrice, pricePerMbA, pricePerMbB, pricePerMbCounter, false);

return 0;

//End Variant 10

}

}