

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»

Выполнил:

Студент группы N3351

Чебунин Константин Олегович 

Проверил:

Федоров Иван Романович _____

Санкт-Петербург

2020 г.

Цель работы: изучить протокол NetFlow, реализовать программный модуль для обработки трафика NetFlow v5 из файла и протарифицировать трафик в соответствии с вариантом задания.

Практическая часть:

В качестве языка программирования для реализации задачи был выбран язык C++, в связи с его удобством в обработке файлов и простыми функциями для обработки строковых переменных. В качестве плагина для оформления графиков был выбран бесплатно распространяемый плагин freeglut.

Задание звучит так:

Протарифицировать абонента с IP-адресом 192.168.250.1 с коэффициентом k: 0,5руб/Мб первые 500Мб, после каждого последующих 500Мб k увеличивается на 0,5руб. Так как в исходном файле количество трафика было слишком мало для демонстрации работы данного правила тарификации, единицы измерения Мб были заменены на Кб.

Работа программы выглядит так:

1. Построчное считывание данных файла с пропуском первой строки, в которой хранятся названия столбцов.
2. Проверка совпадения ip-адреса абонента с исходным ip-адресом посимвольным сравнением.
3. Если ip-адреса совпадают добавление количества трафика к переменной счётчика.
4. Тарификация абонента в соответствии с тарифным правилом, для этого количество Кб трафика делится на 500, и в зависимости от количества пакетов по 500 Кб рассчитывается итоговая сумма.
5. Построение графика зависимости количества трафика от времени

Работа по построению графика выглядит так:

1. Так как временные метки в файле NetFlow не отсортированы по возрастанию, то из всех полученных строк с совпадающим ip-адресом значения времени и трафика записывается в переменную типа vector.

2. Сортировка временных меток по возрастанию с соответствующей перестановкой значений трафика.
3. Привод временных меток к более удобному формату вычитанием самой первой временной метки для получения нулевого значения.
4. Приведение количества к трафику к сумме трафика на текущую временную метку.
5. Приведение значений к процентному соотношению от максимальных значений для большей наглядности графика без разницы от полученных значений.
6. Поточечное построение графика

Пример работы программы можно увидеть на изображениях ниже:

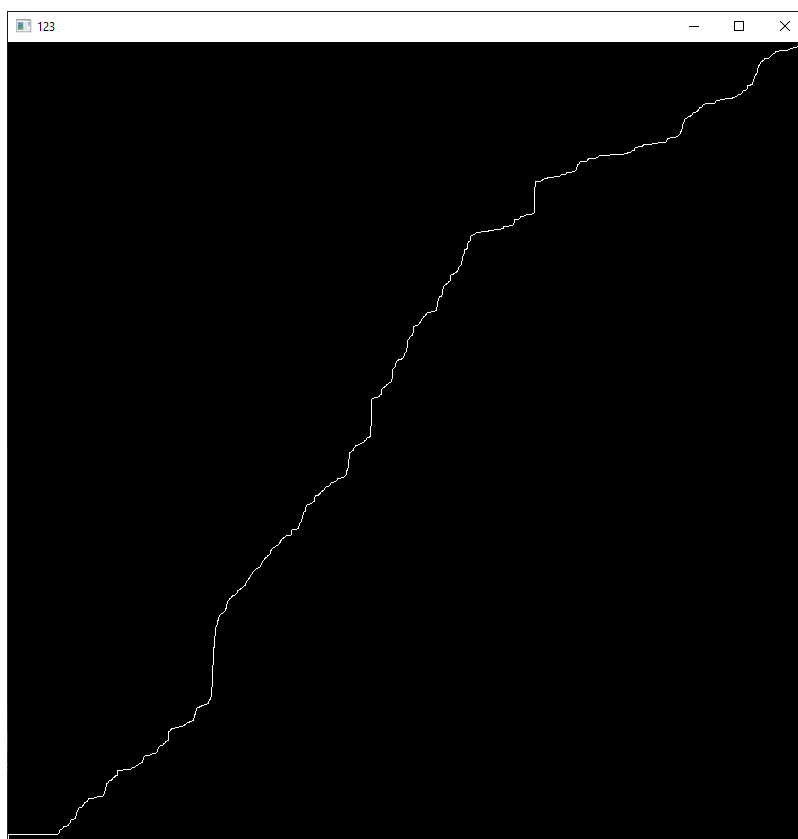
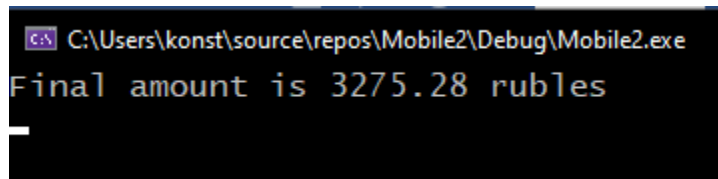


Рис.1 Полученный график

A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\konst\source\repos\Mobile2\Debug\Mobile2.exe". The command prompt displays the text "Final amount is 3275.28 rubles" in a monospaced font. The cursor is positioned at the end of the line.

```
C:\Users\konst\source\repos\Mobile2\Debug\Mobile2.exe
Final amount is 3275.28 rubles
```

Рис.2 Результат работы программы.

Выводы:

В данной работе мы изучили протокол NetFlow, а также реализовали пот помощи языка программирования программный модуль для обработки трафика NetFlow v5 из файла и реализации правила тарификации в соответствии с вариантом задания.

Приложение 1.

Исходный код:

```
#include <string>
#include <locale.h>
#include <iostream>
#include <fstream>
#include <windows.h>
#include "glut.h"
#include "stdlib.h"
#include "freeglut_ext.h"
#include "freeglut_std.h"
#include "freeglut.h"
#include <vector>

using namespace std;
int flag0;
vector <float> a;
vector <float> b;

void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0, 100, 0, 100, 1.0, -1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glPushMatrix();
    glColor3f(1.0, 1.0, 1.0);
    srand(100);
    glBegin(GL_LINE_STRIP);
    for (int i = 0; i < flag0; i++)
    {
        glVertex2d((a[i]-a[0])/(a[flag0-1]-a[0])*100, b[i]/b[flag0-1]*100);
    }
    glEnd();
    glPopMatrix();

    glutSwapBuffers();
}

int main(int argc, char** argv)
{
    ifstream text("input.txt");
    string line;
    string date="";
    string time="";
    string event="";
    string xevent="";
    string proto="";
    string src="";
    string dst="";
    string x_src="";
    string x_dst="";
    string in="";
```

```

string out="";
string arr1="";
string arr2="";
float ins = 0;
float sum = 0;
string source = "192.168.250.1";
int flag1;
flag0 = 0;
flag1 = 0;
getline(text, line);
while (!text.eof())
{
    while (date == "") getline(text, date, ' ');
    if (date == "Summary:") break;
    while (time == "") getline(text, time, ' ');
    while (event == "") getline(text, event, ' ');
    while (xevent == "") getline(text, xevent, ' ');
    while (proto == "") getline(text, proto, ' ');
    while (src == "") getline(text, src, ' ');
    while (arr1 == "") getline(text, arr1, ' ');
    while (dst == "") getline(text, dst, ' ');
    while (x_src == "") getline(text, x_src, ' ');
    while (arr2 == "") getline(text, arr2, ' ');
    while (x_dst == "") getline(text, x_dst, ' ');
    while (in == "") getline(text, in, ' ');
    while (out == "") getline(text, out);
    if (out == "M")
    {
        ins = stof(in) * 1000000;
        getline(text, out);
    }
    else
    {
        ins = stof(in);
    }
    if (src.length() > source.length())
    {
        for (int i = 0; i < source.length(); i++)
            if (source[i] == src[i])
                flag1 = flag1 + 1;
    }
    if (flag1 == source.length())
    {
        sum = sum + ins;
        flag0 = flag0 + 1;
        a.push_back((((float)time[0] * 10 + (float)time[1]) * 60 * 60 * 100 +
        ((float)time[3] * 10 + (float)time[4]) * 60 * 1000 + ((float)time[6] * 10 +
        (float)time[7]) * 1000 + ((float)time[9] * 100 + (float)time[10]) * 10 + (float)time[11]
        ) / 50000);
        b.push_back(ins / 1024);
    }
    flag1 = 0;
    date = "";
    time = "";
    event = "";
    xevent = "";
    proto = "";
    src = "";
    dst = "";
    x_src = "";
    x_dst = "";
    in = "";
    out = "";
    arr1 = "";

```

```

    arr2 = "";
}
float price;
price = 0;
int price1;
int flag2;
flag2 = 0;
sum = sum / 1024;
price1 = (int)(sum / 500);
for (int i = 1; i <= price1; i++)
{
    price = price + 500 * 0.5 * i;
    flag2 = i;
}
float tmp;
float tmp2;
for (int i = 0; i < flag0; ++i)
{
    for (int j = 0; j < flag0 - 1; ++j)
    {
        if (a[j + 1] < a[j])
        {
            tmp = a[j + 1];
            tmp2 = b[j + 1];
            a[j + 1] = a[j];
            a[j] = tmp;
            b[j + 1] = b[j];
            b[j] = tmp2;
        }
    }
}
float sss;
sss = b[0];
for (int i = 1; i < flag0; i++)
{
    b[i] = b[i] + b[i-1];
    sss = sss + b[i];
}
price = price + (sum - 500.0 * flag2) * (flag2 + 1.0) * 0.5;
cout << "Final amount is " << price << " rubles" << '\n';
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
glutInitWindowSize(800, 800);
glutInitWindowPosition(100, 100);
glutCreateWindow("123");
glClearColor(0.0, 0.0, 0.0, 0.0);
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}

```