

**Цель работы:** ознакомление с протоколом Netflow

**Реализация:** данная работа реализована с помощью нескольких языков программирования: С++ и Python, так как первый язык для меня является более простым в понимании, а второй удобным для реализации построения графиков. C помощью скрипта запускается программa.

**Исходный код:**

main.cpp

|  |
| --- |
| #include  <iostream> |
| #include <fstream> |
| #include <string> |
| #include <vector> |
| #include <sstream> |
| #include <iterator> |
| using namespace std; |
| /\* |
| \* get\_tarif() - читает файл с описанием тарифа и возвращает структуру с  данными о нем. |
| \* Пример файла: |
| \* ip: 217.15.20.194 |
| \* k: 1 |
| \* firstfree: 1000 |
| \*/ |
| struct s\_tarif{ |
| string ip; |
| int k; |
| int firstfree; |
| }; |
| s\_tarif get\_tarif(ifstream\* stream){ |
| s\_tarif tarif; |
| string a; |
| { |
| getline(\*stream, a); |
| std::istringstream buf(a); |
| std::istream\_iterator<std::string> beg(buf), end; |
| std::vector<std::string> tokens(beg, end); |
| tarif.ip = tokens[1]; |
| } |

|  |
| --- |
| { |
| getline(\*stream, a); |
| std::istringstream buf(a); |
| std::istream\_iterator<std::string> beg(buf), end; |
| std::vector<std::string> tokens(beg, end); |
| tarif.k = stoi(tokens[1]); |
| } |
| { |
| getline(\*stream, a); |
| std::istringstream buf(a); |
| std::istream\_iterator<std::string> beg(buf), end; |
| std::vector<std::string> tokens(beg, end); |
| tarif.firstfree = stoi(tokens[1])\*131072; |
| } |
| return tarif; |
| } |
| /\* |
| \* process() читает строку из файла и возвращает 0 если ip не совпадает с  данным в тарифе, |
| \* количество байт в пакете если совпадает и -1 если файл закончился |
| \*/ |
| int process(ifstream\* stream, string\* target\_ip, ofstream\* forpython){ |
| string str; |
| if(getline(\*stream, str)) { |
| std::istringstream buf(str); |
| std::istream\_iterator<std::string> beg(buf), end; |
| std::vector<std::string> tokens(beg, end); |
| if (tokens[0]=="Summary:") |
| return -1; |
| string ip = tokens[7].substr(0, tokens[7].find\_first\_of(':')); |
| int bytes; |
| if(tokens[12]!="M"){ |
| bytes = stoi(tokens[11]); |
| }else{ |
| bytes = stod(tokens[11])\*1024\*1024; |
| } |
| if (str.find(\*target\_ip) != str.npos) { |
| \*forpython << tokens[0] << " " << tokens[1] << " "; |
| return bytes; |
| } else { |
| return 0; |
| } |
| }else{ |
| return -1; |

|  |
| --- |
| } |
| } |
| int main() { |
| string s; |
| ifstream inf("data"); |
| ofstream forpython("forplot"); |
| ifstream tarif\_file("tarif"); |
| s\_tarif tarif = get\_tarif(&tarif\_file); |
| getline(inf, s); |
| int fullsumm = 0; |
| while(true) { |
| int packet\_size = process(&inf, &tarif.ip, &forpython); |
| if (packet\_size == -1) { |
| break; |
| } |
| fullsumm += packet\_size; |
| if(packet\_size>0) |
| forpython << packet\_size<< endl; |
| } |
| int a = fullsumm - (tarif.firstfree); |
| /\* if (a <= tarif.firstfree) |
| fullsumm = 0; |
| else\*/ |
| fullsumm -= tarif.firstfree; |
| std::cout << (double) fullsumm /131072 \* tarif.k << endl; |
| system("python3 plot.py"); |
| inf.close(); |
| forpython.close(); |
| tarif\_file.close(); |
| return 0; |
| } |

plot.py

|  |
| --- |
| import datetime  import matplotlib.pyplot as plt |
| data = [] |
| x = [] |
| y = [] |
| with open('forplot') as file: |
| for i in file.readlines(): |
| splitted = i.split() |
| datestr = splitted[0]+' '+splitted[1] |
| date = datetime.datetime.strptime(datestr, '%Y-%m-%d %H:%M:%S.%f') #  2020-02-25 12:29:46.040 |

|  |
| --- |
| data.append((date,int(splitted[2]))) |
| data.sort(key=lambda x: x[0]) |
| x = [i[0] for i in data] |
| a = 0 |
| for i in data: |
| a += i[1] |
| y.append(a/128) |
| (fig, ax) = plt.subplots(1, 1) |
| ax.plot(x, y) |
| for n, label in enumerate(ax.xaxis.get\_ticklabels()): |
| if n % 2 != 0: |
| label.set\_visible(False) |
| ax.yaxis.set\_major\_formatter(plt.FormatStrFormatter('%d')) |
| plt.ylabel('Kbit') |
| plt.savefig('plot.png') |

nfdump\_to\_data.sh

|  |
| --- |
| echo  "Making input  files" |
| rm forplot plot.png |
| nfdump -r nfcapd.202002251200 > data |
| echo -e "ip: 217.15.20.194\nk: 1\nfirstfree: 1000" > tarif |
| echo -e "Installing dependencies for plotting" |
| echo "Compiling" |
| g++ -o lab2.out main.cpp |
| echo "Computing" |
| ./lab2.out |

Выводы: в ходе лабораторной работы был изучен формат NeTflow, была произведена тарификация абонентов.