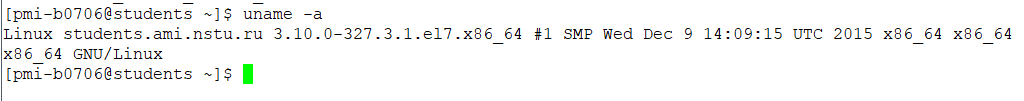
|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Теоретической и прикладной математики | | |
|  | | |
| Лабораторная работа № 5 | | |
| по дисциплине «Операционные системы, среды и оболочки» | | |
|  | | |
| **Анализ функционирования и диагностика IP-сетей** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-02 |
| Вариант: | 6 |
| Студент: | Сидоров Даниил, |
|  | Дюков Богдан |
| Преподаватель: | Кобылянский Валерий Григорьевич, |
|  | Филиппова Елена Владимировна  . |
|
|  |  |
|  | | |
| Новосибирск | | |
| 2022 | | |

1. **Цель работы**

Приобретение практических навыков работы с сетевыми командами операционных систем Windows и Linux, предназначенными для анализа и диагностики сетей TCP/IP, а также со средствами интерактивной диагностики сетей.

1. **Ход работы для 1-го этапа**
2. Подключились с помощью клиента Putty к серверу **fpm2.ami.nstu.ru** и с помощью команды **uname** получили полную информацию об установленной операционной системе и аппаратной платформе.



Linux – имя ядра

Students.ami.nstu.ru – имя машины в сети

3.10.0-327.3.1.е17.х86\_64 -номер выпуска ОС

#1 SMP Wed Dec 9 14.09.15 UTC 2015 – версия ядра

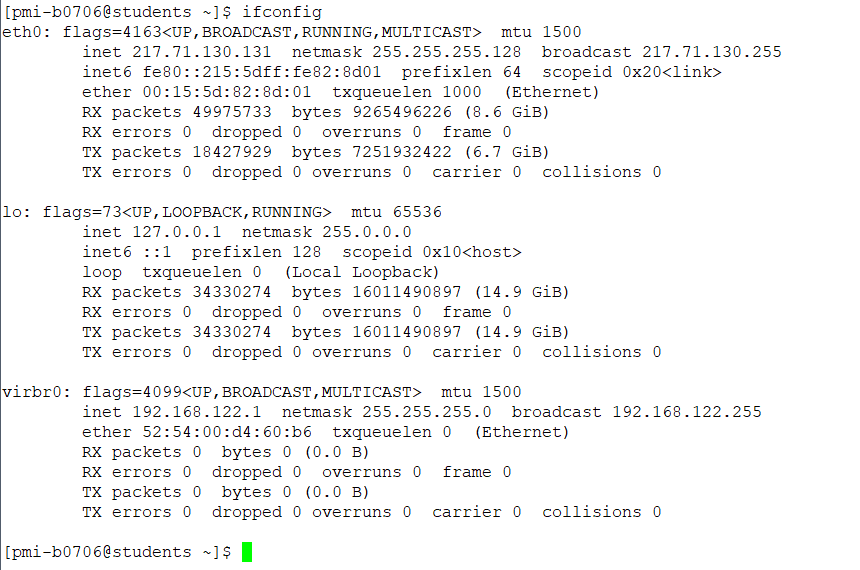
х86\_64 – тип оборудования машины

х86\_64 – тип процессора

х86\_64 – тип аппаратной платформы

GNU/Linux – имя ОС

1. Получили статистику по сетевым интерфейсам РК и сервера **fpm2.ami.nstu.ru**.



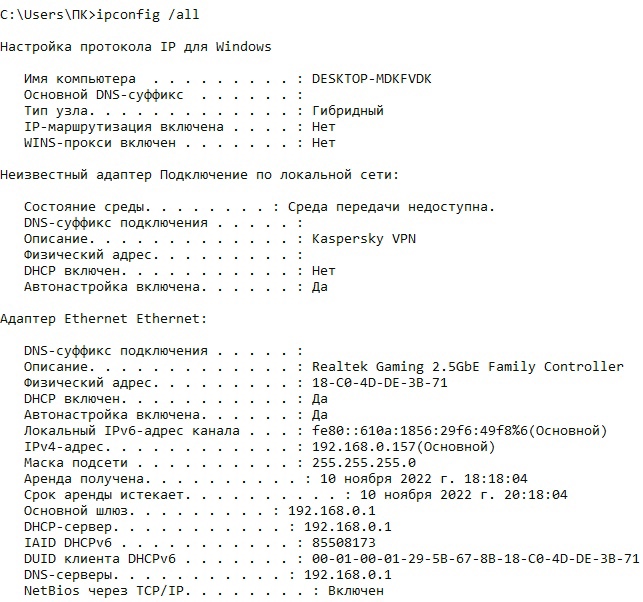
3 сетевых интерфейса:

1)Адрес виртуального интернет порта.

2) Петлевой адрес.

3) Виртуальным мост.

Для каждого интерфейса указываются физические и сетевые адреса (ether, inet, inet6), маска сети (netmask), максимальный размер кадра (MTU), число принятых и переданных пакетов (RX packets, TX packets).



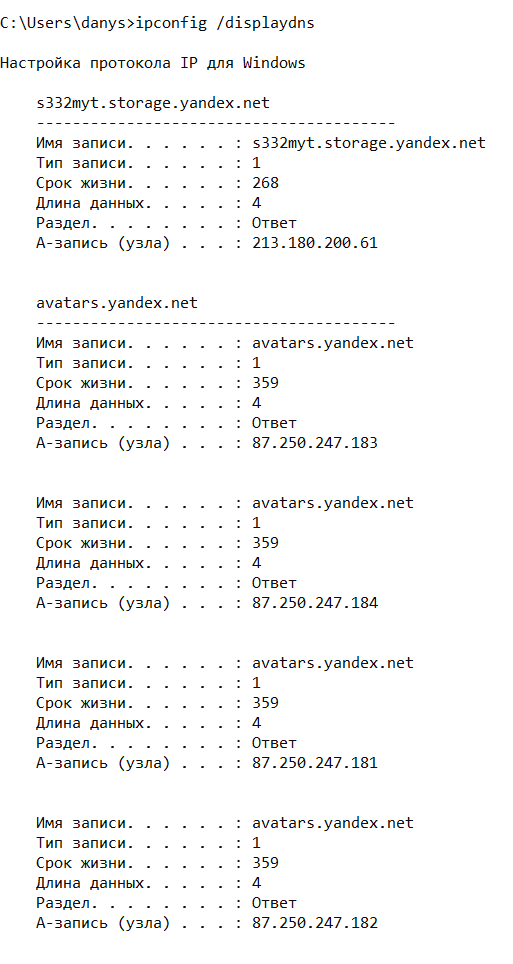
Локальный IPv4-адрес маршрутизатора 192.168.0.157

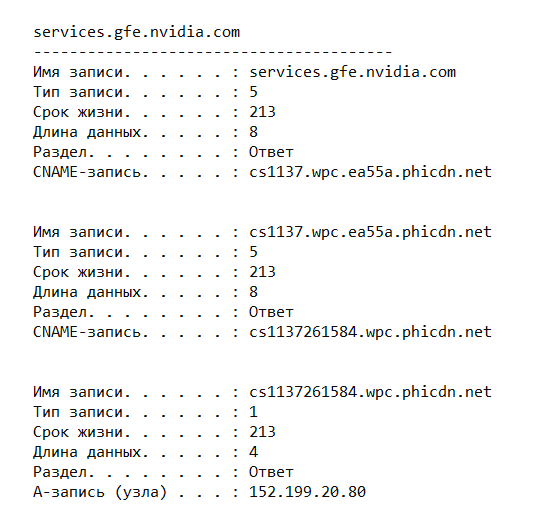
МАС-адрес маршрутизатора 18-С0-4D-DE-3B-71

Маска подсети: 255.255.255.0

Основной шлюз: 192.168.0.1 и прочее.

1. Просмотрели содержимое DNS-кэша, очистили кэш.





В состав DNS-записи входят следующие поля:

**Имя записи**- Определяет домен, к которому относится (привязана) данная ресурсная запись.

**Тип записи**- Указывает на тип (назначение) данной ресурсной записи.

Срок жизни- время жизни (хранения) DNS-записи в кэше DNS-сервера.

**Длина данных**- длина поля данных

**Class** (Класс). Здесь указывается тип рабочей сети. Теоретически, система может работать во всех ее типах. Но, TCP/IP сети — самые распространенные. Поэтому, поле редко используется.

**Наиболее важные типы записей:**

**А-запись**- Address record указывает на конкретный IP-адрес домена. Без нее сайт работать не будет. По этой записи система определяет к какому серверу обращаться за получением информации, когда пользователь вводит название сайта в адресную строку веб-браузера.

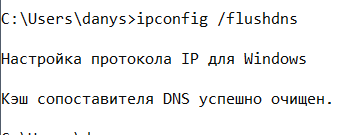
Например, запрос А-записи на имя refferals.icann.org вернет его IPv4 адрес 192.0.34.164

**AAAA запись DNS** — аналог предыдущей А-записи. В значении указывается внешний IP-адрес в формате IPv6.

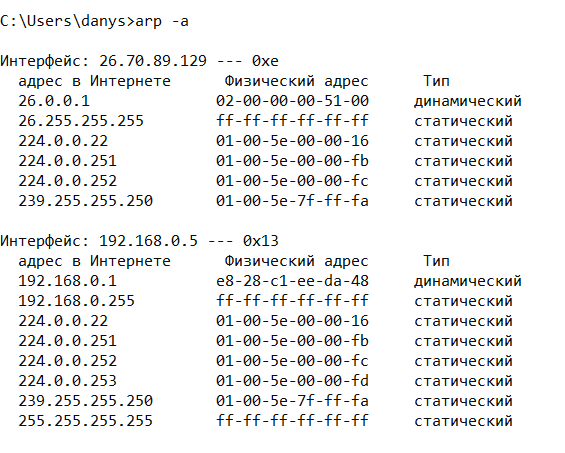
**MX-запись** задает почтовый сервер, который будет принимать и отправлять почту для данного домена. Запись может указывать на внутренний или внешний почтовый сервер.

**CNAME-запись**- CNAME («каноническое имя») указывает на расположение хостов на одном сервере. С ее помощью, можно прописать несколько доменов и поддоменов в рамках одного сервера. Используется для перенаправления на другое имя.

**Очистили кэш DNS:**



1. Просмотрели содержимое ARP-таблицы, пояснили характеристики записей, выполнили добавление и удаление статических записей.



В таблице 3 столбца:

1) Адрес в Интернете некоторого хоста

2) Физический адрес Хоста

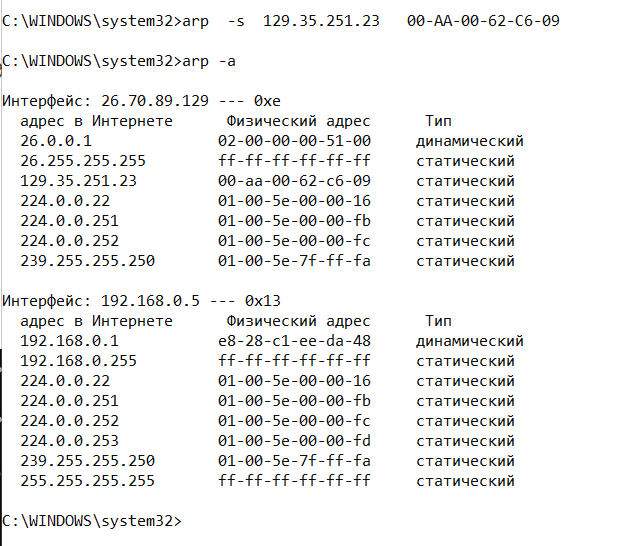
3) Тип записи

Маршрутизатор в локальной сети имеет IP-адрес 192.168.0.1 и MAC-адрес e8-28-c1-ee-da-48.

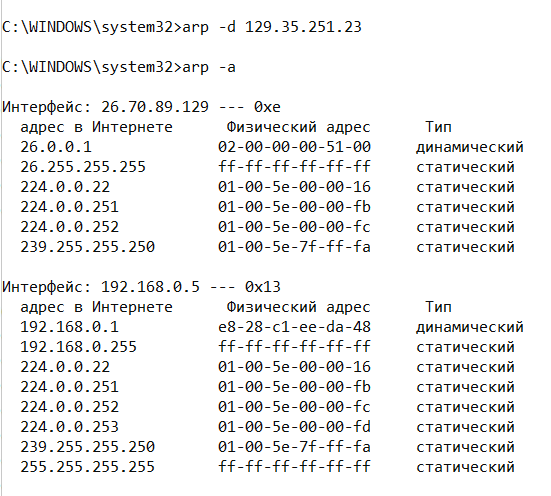
Статические записи создаются вручную и существуют до следующей перезагрузки устройства, динамические записи периодически обновляются.

Адрес 255.255.255.255 является широковещательным и интерпретируется как MAC-адрес FF-FF-FF-FF-FF-FF, который примут все машины согласно протоколу Ethernet.

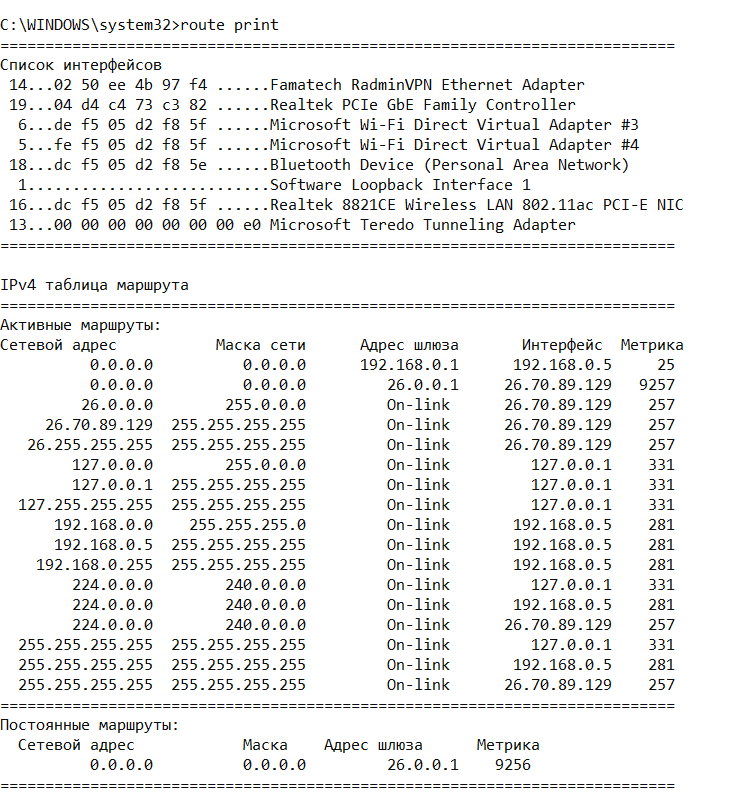
Добавили запись:

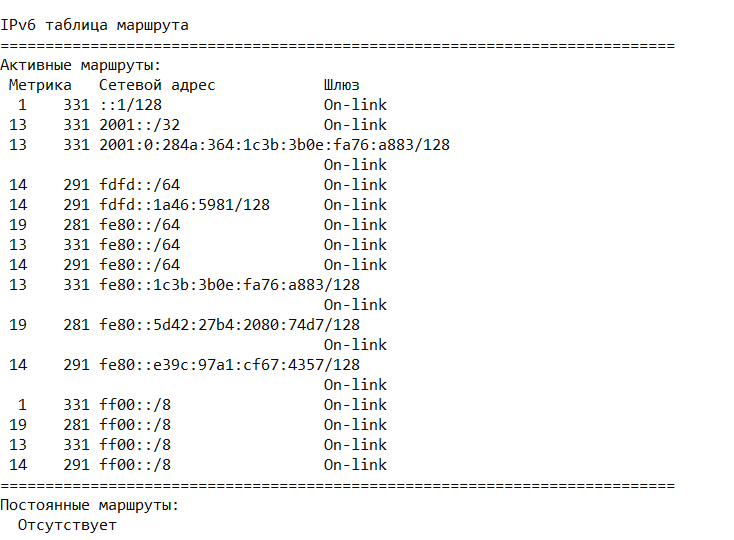


Удалили запись:



1. Просмотрели содержимое таблицы маршрутизации, пояснили характеристики записей





Выведен список IPv4 и IPv6 интерфейсов.

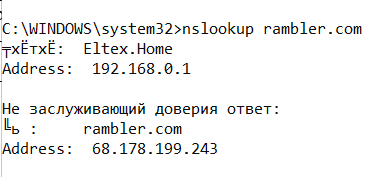
Поле «Адрес шлюза» задает адрес ближайшего маршрутизатора для продвижения IP-пакета к заданной сети, поле «Интерфейс» указывает IP-адрес выходного сетевого интерфейса текущего маршрутизатора, а поле «Метрика» хранит целое число, используемое в качестве критерия выбора маршрута передачи пакета, например, количество промежуточных маршрутизаторов, которые должен пройти пакет, чтобы достигнуть адресата.

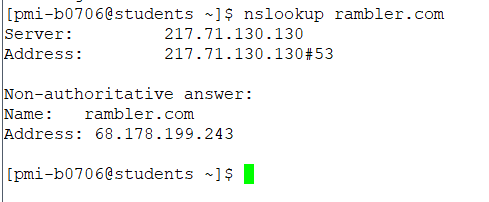
Рассмотрим IPv4 маршруты:

По ним видно, что пакеты отправляются на адрес маршрутизатора, как на адрес шлюза. А маршрутизатор в свою очередь, исходя из своих таблиц маршрутизации, определит куда переслать пакет.

6. В командном режиме на РК и на сервере определили IP-адреса поисковых систем в соответствии с вариантом из таблицы.

|  |  |
| --- | --- |
| Вариант | Название |
| 6 | rambler.com |

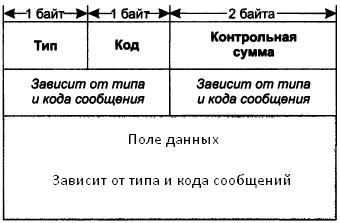




В начале нам выводится адрес маршрутизатора локальной сети. Для РК это 192.168.0.1, для сервера 217.71.130.130. После выводится IP адрес поисковой системы.

7. В командном режиме на РК и на сервере определили IP-адрес узлов сети в соответствии с номером варианта, указанного в таблице, выполнили его пингование и трассировку.

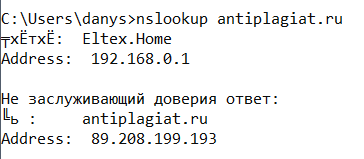
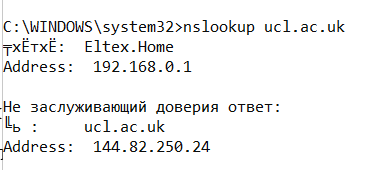
Пингование и трассировка происходит при помощи протокола ICMP, который формирует ICMP - сообщения.



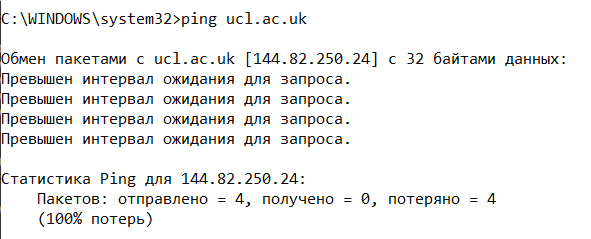


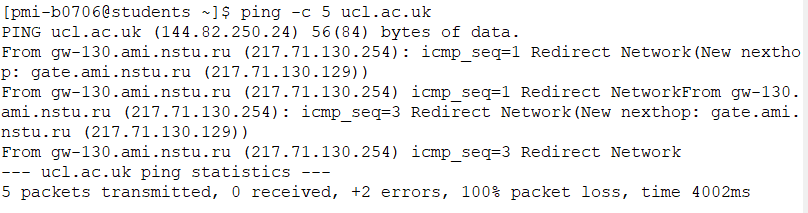
|  |  |
| --- | --- |
| Вариант | Доменное имя |
| 6 | ucl.ac.uk, antiplagiat.ru |

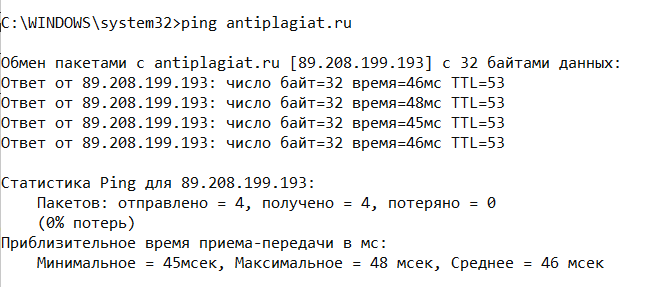
Определили IP-адрес:

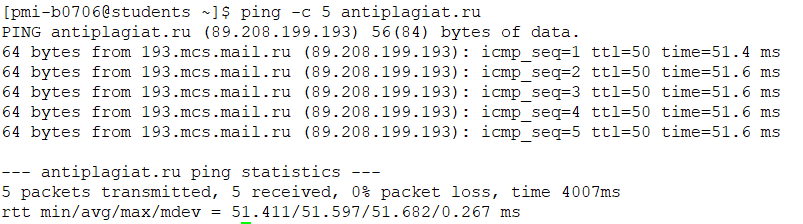


Выполнили пингование на РК и сервере:

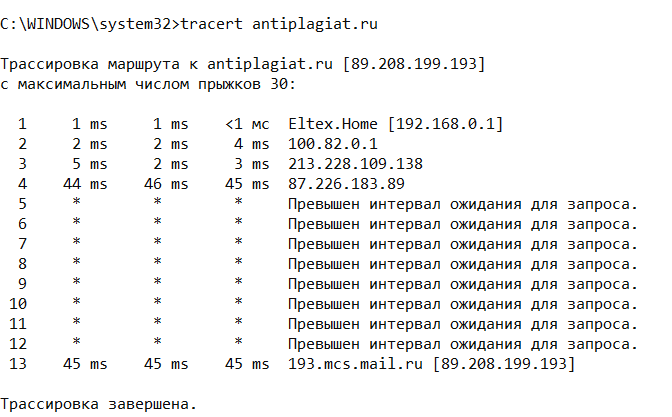


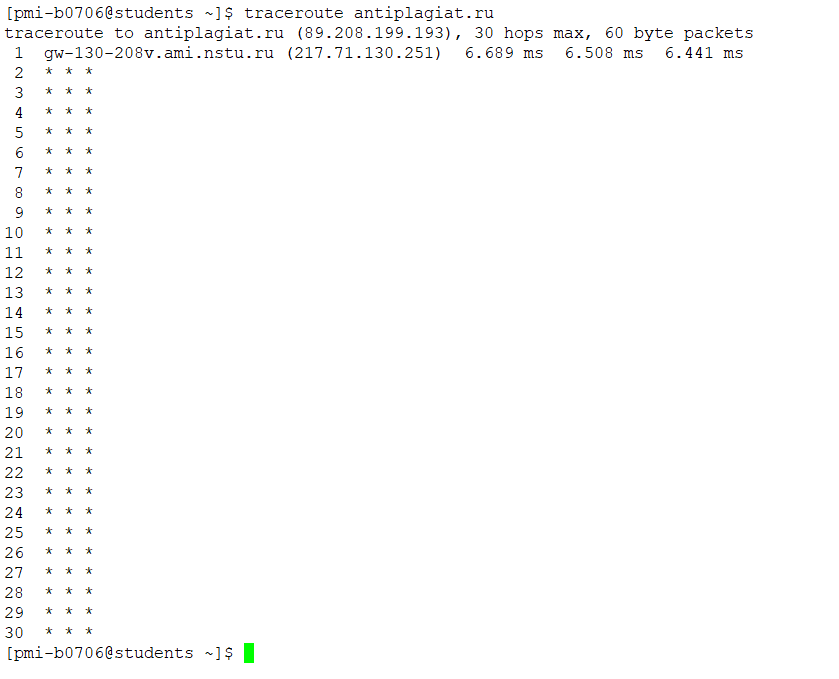


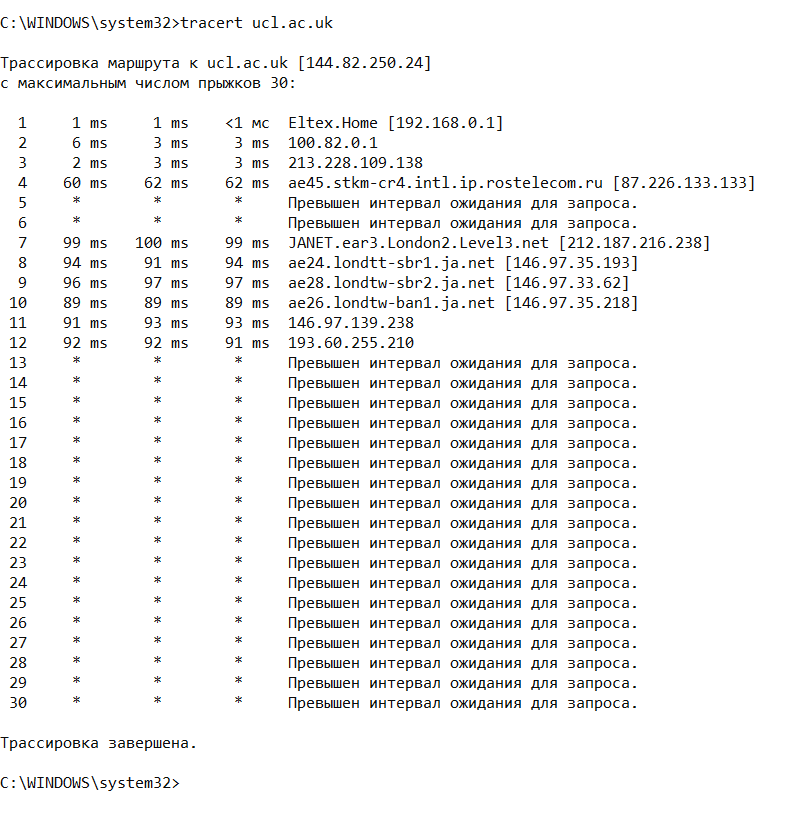


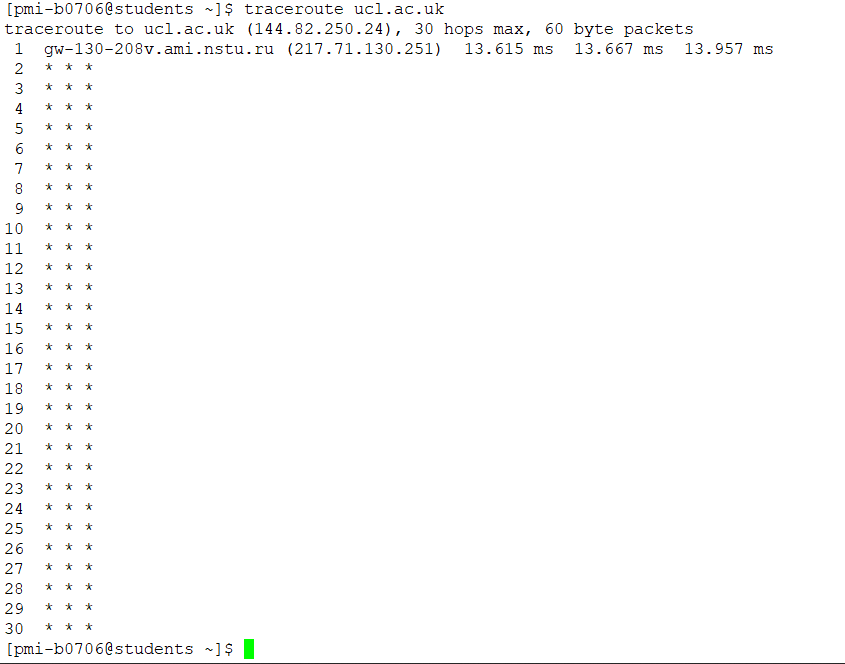


Выполнили трассировку на РК и сервере:



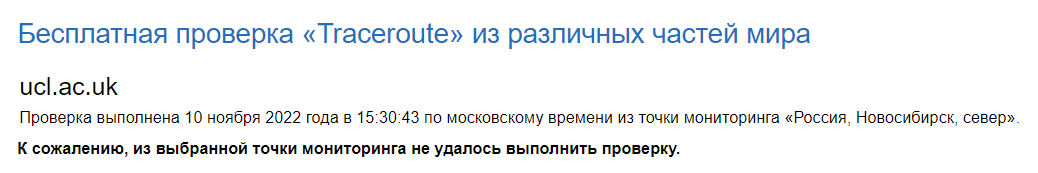


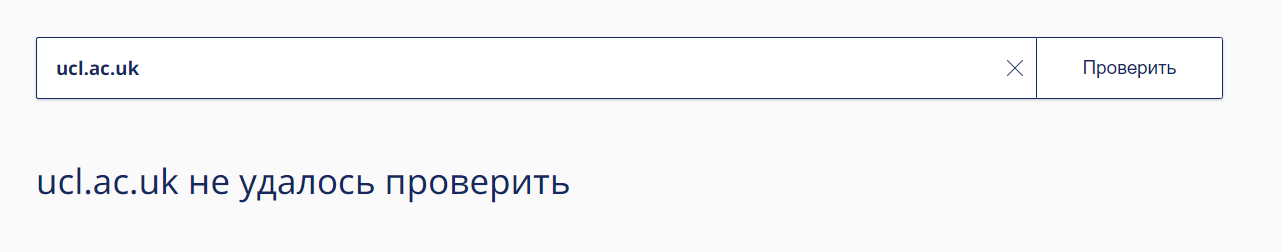


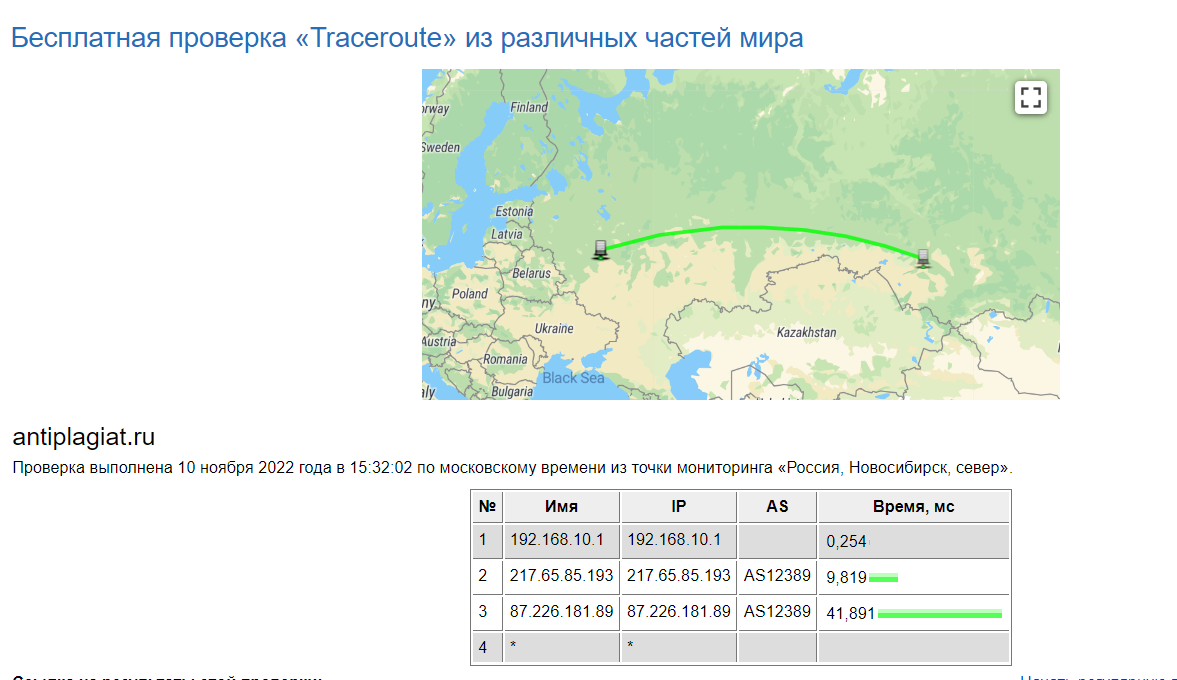


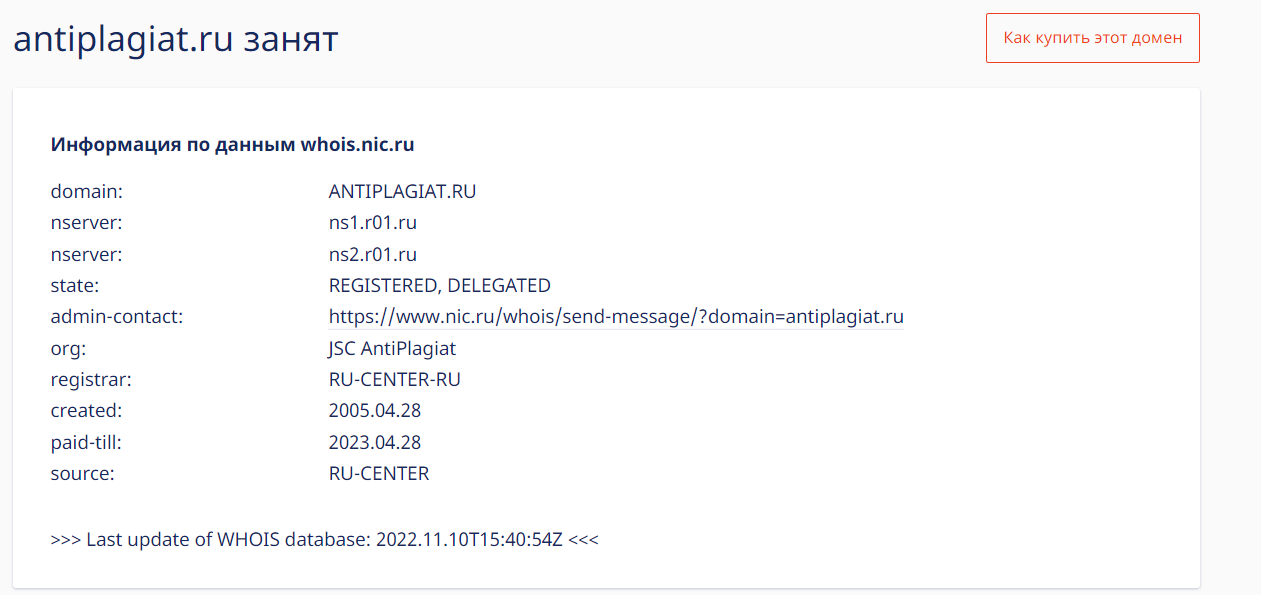
Исходя из результатов можно сделать вывод, что доступ к данным сайтам заблокирован на сервере, в то время как на РК доступ есть только для **antiplagiat.ru** и на трассировке видно каждый конечный пункт.

8. С помощью интерактивных сетевых сервисов (**ping-admin.ru, whois.ru**) выполнили трассировку, определили местонахождение и владельца узла сети в соответствии с номером варианта, указанного в таблице. Результат трассировки в виде скриншота географической карты представили в отчете и выполнили его анализ. Начальный пункт трассировки – г. Новосибирск.









Исходя из результатов трассировки видно, что для доступа к сайту выбирается определённый доступный шлюз, ближайший к нужному нам ресурсу, доступным маршрутизатору. Например, для компьютера шлюз – локальный маршрутизатор, для маршрутизатора – маршрутизатор провайдера, для маршрутизатора провайдера – маршрутизатор ещё более высшего уровня и так далее. Наш пакет проходит по роутерам для того, чтобы получить доступ к нужному нам ресурсу.

1. **Ход работы для 2-го этапа**

Вариант задания:

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Аналог | ОС | Функция |
| 6 | traceroute | Linux | трассировка маршрута передачи данных |

Наша программа выполняет трассировку маршрута до указанного узла. Трассировка обеспечивается путем отправки ICMP эхо-запроса с постепенным увеличением TTL. По его истечении промежуточный узел будет нам сообщать об этом. Процесс повторяется до тех пор, пока пакет не достигнет целевого узла и при получении ответа от этого узла трассировка считается завершённой.

**Текст программы**

**Файл traceroute.h**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <unistd.h>**

**#include <sys/time.h>**

**#include <sys/ioctl.h>**

**#include <arpa/inet.h>**

**#include <sys/socket.h>**

**#include <pthread.h>**

**#include <stdbool.h>**

**#include <sys/types.h>**

**#include <netinet/ip.h>**

**#include <netinet/ip\_icmp.h>**

**#include <netdb.h>**

**/\*Максимальный размер пакета\*/**

**#define MAX\_SPACK 4096**

**/\* Структура, описывающая заголовок IP пакета \*/**

**struct header\_ipv4**

**{**

**u\_char ver\_len; /\* 4 бита - версия + 4 бита - длина заголовка в октетах \*/**

**u\_char dscp\_ecn; /\* 6 бит - тип обслуживания + 2 бита - перегрузка \*/**

**short length;**

**short ident; /\* Идентификатор фрагмента \*/**

**short flag\_offset; /\* 3 бита флаги, остальное - смещение фрагмента \*/**

**u\_char ttl;**

**u\_char proto; /\* Тип инкапсулированного протокола \*/**

**short chsum;**

**int sip; /\* IP адрес источника \*/**

**int dip; /\* IP адрес назначения \*/**

**};**

**/\***

**\* Структура, описывающая формат ICMP пакета для отправки и приема echo reply и**

**\* echo request**

**\*/**

**struct icmp\_echo**

**{**

**u\_char type; /\* Тип сообщения \*/**

**u\_char code; /\* Код сообщения \*/**

**u\_short chsum;**

**u\_short id; /\* Идентификатор \*/**

**u\_short nseq; /\* Номер последовательности \*/**

**struct timeval time; /\* Временная метка \*/**

**};**

**/\***

**\* Структура, описывающая формат ICMP пакета, получаемого от узла при**

**\* истечении TTL**

**\*/**

**struct icmp\_timeexp**

**{**

**u\_char type; /\* Тип сообщения \*/**

**u\_char code; /\* Код сообщения \*/**

**u\_short chsum;**

**u\_short zero;**

**struct header\_ipv4 hip; /\* Заголовок пакета, TTL которого закончился \*/**

**};**

**/\* Функция принятия ICMP пакета\*/**

**int icmp\_recv(int sock, char \*ipto, char \*ipfrom);**

**/\* Функция подсчета контрольной суммы ICMP пакета \*/**

**u\_short icmp\_checksum(u\_char \*packet);**

**/\*Функция формирования ICMP пакета для эхо запроса/ответа \*/**

**void icmp\_build(struct icmp\_echo \*header, u\_char type, u\_char code);**

**Файл icmp.c**

**#include "traceroute.h"**

**/\* Функция подсчета контрольной суммы ICMP пакета \*/**

**u\_short icmp\_checksum(u\_char \*packet)**

**{**

**int length; /\* Длина ICMP пакета \*/**

**u\_short fbyte; /\* Первый байт 16 битного блока \*/**

**u\_short sbyte; /\* Второй байт 16 битного блока \*/**

**int chsum;**

**length = sizeof(struct icmp\_echo);**

**chsum = 0;**

**/\***

**\* Считаем контрольную сумму, разворачивая по 2 байта. Обязательно до**

**\* этого поле контрольной суммы в заголовке должно быть обнулено**

**\*/**

**for(int i = 0; i < length; i += 2)**

**{**

**fbyte = packet[i];**

**sbyte = packet[i + 1];**

**chsum += (fbyte<<8)|sbyte;**

**}**

**chsum = chsum + (chsum>>16);**

**return (u\_short)~chsum;**

**}**

**/\***

**\* Формирует ICMP пакет для эхо запроса/ответа**

**\* первый параметр - Указатель на пакет, который будет сформирован**

**\* второй параметр - Тип пакета**

**\* третий параметр - Код сообщения**

**\*/**

**void icmp\_build(struct icmp\_echo \*packet, u\_char type, u\_char code)**

**{**

**packet->type = type;**

**packet->code = code;**

**packet->id = htons(rand()%256);**

**packet->nseq = htons(rand()%256);**

**/\***

**\* Наличие временной метки является обязательным для корректной**

**\* обработки узлом назначения.**

**\*/**

**gettimeofday(&(packet->time), NULL);**

**packet->chsum = 0;**

**packet->chsum = htons(icmp\_checksum((u\_char \*)packet));**

**}**

**/\***

**\* Принимает ICMP пакет**

**\***

**\* Первый параметр - сокет, на котором принимаем**

**\***

**\* Возврат -1 Истекло время ожидания ответа**

**\* Возврат 0 Ответ пришел от конечного узла**

**\* Возврат 1 Ответ пришел от промежуточного матршрутизатора**

**\*/**

**int icmp\_recv(int sock, char \*ipto, char \*ipfrom)**

**{**

**u\_char packet[MAX\_SPACK]; /\* Перехваченный пакет \*/**

**int length; /\* Длина перехваченной части \*/**

**struct header\_ipv4 \*hdr\_ip; /\* Заголовок IP \*/**

**struct icmp\_echo \*hdr\_icmpe; /\* ICMP ответ от конечного узла \*/**

**struct icmp\_timeexp \*hdr\_icmpt; /\* ICMP, ответ от промежуточного узла \*/**

**struct in\_addr tmp; /\* Нужен для коневертации IP в строку \*/**

**/\* Принимаем пакеты, пока не попадется нужный или не истечет время ожидания ответа \*/**

**while (1)**

**{**

**length = recvfrom(sock, &packet, MAX\_SPACK, 0, NULL, NULL);**

**if (length == -1)**

**{**

**return -1;**

**}**

**hdr\_ip = (struct header\_ipv4 \*)&packet;**

**/\* Проверяем является ли это ответом конечного узла \*/**

**if (hdr\_ip->sip == inet\_addr(ipto))**

**return 0;**

**/\* Проверяем является ли это сообщением о истечении TTL \*/**

**if (packet[sizeof(struct header\_ipv4)] == 11)**

**{**

**hdr\_icmpt = (struct icmp\_timeexp \*)(packet + sizeof(struct header\_ipv4));**

**/\* Убеждаемся, что это пакет для нас \*/**

**if (hdr\_icmpt->hip.dip == inet\_addr(ipto))**

**{**

**tmp.s\_addr = hdr\_ip->sip;**

**strcpy(ipfrom, inet\_ntoa(tmp));**

**return 1;**

**}**

**}**

**}**

**return -1;**

**}**

**Файл main.c**

**#include "traceroute.h"**

**#define MAX\_HOPE 30**

**/\*Подсчет времени с момента отправки пакета до его получения\*/**

**static double GetTimeout(struct timeval start)**

**{**

**struct timeval end;**

**gettimeofday(&end, NULL);**

**double seconds = end.tv\_sec - start.tv\_sec;**

**double useconds = end.tv\_usec - start.tv\_usec;**

**return (seconds \* 1000.0 + useconds / 1000.0) + 0.5;**

**}**

**/\*Получение Ip адреса из доменного имени\*/**

**static bool GetIPFromHostname(const char\* hostname, char\* host\_ip)**

**{**

**struct hostent\* he;**

**struct in\_addr\*\* addr\_list;**

**if ((he = gethostbyname(hostname)) == NULL)**

**{**

**herror("gethostbyname");**

**return false;**

**}**

**addr\_list = (struct in\_addr\*\*)he->h\_addr\_list;**

**for(int i = 0; addr\_list[i] != NULL; ++i)**

**{**

**strcpy(host\_ip , inet\_ntoa(\*addr\_list[i]));**

**return true;**

**}**

**return false;**

**}**

**int main(int argc, void \*argv[])**

**{**

**int sock;**

**struct sockaddr\_in addr;**

**struct icmp\_echo packet;**

**u\_short ttl; /\*Текущий TTL\*/**

**struct timeval time; /\*Время ожидания ответа\*/**

**char ipfrom[16]; /\*IP промежуточного узла\*/**

**int reply; /\*Тип ответа на посланый запрос\*/**

**/\*Проверка параметров на валидность\*/**

**if (argc != 2)**

**{**

**printf("Неверно указаны параметры\n");**

**exit(-1);**

**}**

**/\*Получаем ip-адрес из доменного имени\*/**

**char host\_ip[16];**

**GetIPFromHostname(argv[1], host\_ip);**

**if(inet\_addr(host\_ip) == INADDR\_NONE)**

**{**

**printf("Некорректно задан адрес\n");**

**exit(-1);**

**}**

**sock = socket(AF\_INET, SOCK\_RAW, IPPROTO\_ICMP);**

**if (sock == -1) {**

**perror("socket");**

**exit(-1);**

**}**

**if (strcmp(host\_ip, "EMPTY") != -1)**

**{**

**printf("traceroute to %s (%s), 30 hops max:\n", argv[1], host\_ip);**

**}**

**/\*Задаем таймаут для получения\*/**

**time.tv\_sec = 2;**

**time.tv\_usec = 0;**

**/\*Заполняем ip-назначения\*/**

**addr.sin\_family = AF\_INET;**

**addr.sin\_addr.s\_addr = inet\_addr(host\_ip);**

**if (setsockopt(sock, SOL\_SOCKET, SO\_RCVTIMEO,&time,sizeof(time)) == -1)**

**{**

**perror("Не удалось настроить таймер. Возможна блокировка");**

**}**

**/\*Начинаем отправлять icmp-пакеты, постепенно увеличивая TTL\*/**

**for(int ttl = 1; ttl < MAX\_HOPE; ttl++)**

**{**

**/\*Запускаем таймер\*/**

**struct timeval start;**

**gettimeofday(&start, NULL);**

**/\*Установка параметров сокета\*/**

**setsockopt(sock, SOL\_IP, IP\_TTL, &ttl, sizeof(ttl));**

**/\*Формирование icmp-пакета**

**1й параметр - указатель на пакет**

**2й параметр - тип пакета**

**3й параметр - код сообщения**

**\*/**

**icmp\_build(&packet, 8, 0);**

**/\*Отправляем пакет\*/**

**sendto(sock, (void \*)&packet, sizeof(packet), 0,**

**(struct sockaddr \*)&addr, sizeof(addr));**

**/\*Получаем ответ\*/**

**reply = icmp\_recv(sock, (char \*)host\_ip, ipfrom);**

**switch(reply)**

**{**

**/\*В случае, если истекло время ожидания ответа\*/**

**case -1:**

**printf("%2d %s\n", ttl, "\* \* \*");**

**break;**

**/\*В случае, если ответ от конечного узла\*/**

**case 0:**

**/\*Фиксируем время получения первого пакета\*/**

**printf("%2d\t\t%s\t\t%06.3lf ms", ttl, (char \*)host\_ip, GetTimeout(start));**

**/\*Делаем то же самое еще с двумя пакетами\*/**

**for (uint8\_t packet\_num = 0; packet\_num != 2; ++packet\_num)**

**{**

**gettimeofday(&start, NULL);**

**sendto(sock, (void \*)&packet, sizeof(packet), 0,**

**(struct sockaddr \*)&addr, sizeof(addr));**

**reply = icmp\_recv(sock, (char \*)host\_ip, ipfrom);**

**printf("\t%06.3lf ms", GetTimeout(start));**

**}**

**/\* Завершаем выполнение програмы\*/**

**ttl = MAX\_HOPE;**

**printf("\n");**

**break;**

**/\*Если ответ пришел от промежуточного маршрутизатора\*/**

**case 1:**

**/\*Фиксируем время получения первого пакета\*/**

**printf("%2d\t\t%s\t\t%06.3lf ms", ttl, ipfrom, GetTimeout(start));**

**/\*Делаем то же самое еще с двумя пакетами\*/**

**for (uint8\_t packet\_num = 0; packet\_num != 2; ++packet\_num)**

**{**

**gettimeofday(&start, NULL);**

**sendto(sock, (void \*)&packet, sizeof(packet), 0,**

**(struct sockaddr \*)&addr, sizeof(addr));**

**reply = icmp\_recv(sock, (char \*)host\_ip, ipfrom);**

**printf("\t%06.3lf ms", GetTimeout(start));**

**}**

**printf("\n");**

**break;**

**}**

**}**

**close(sock);**

**}**

Компиляция программы и генерация объектного файла осуществлялась с помощью команды:

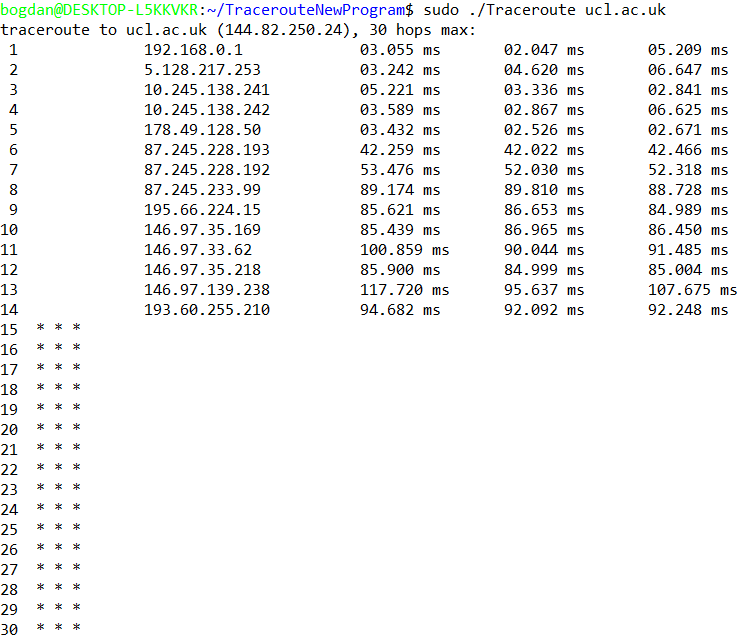
**gcc traceroute.h main.c imcp.c -o Traceroute**

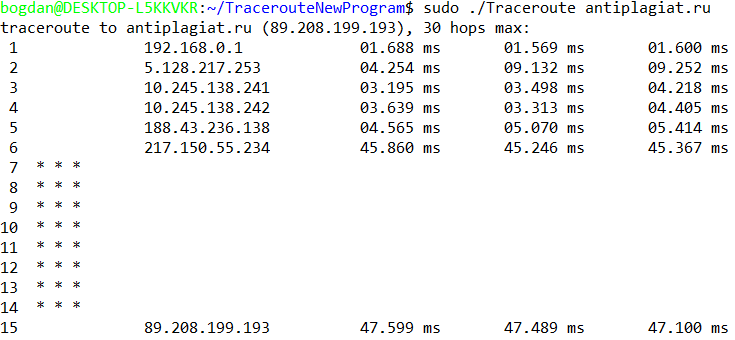
А запуск программы:

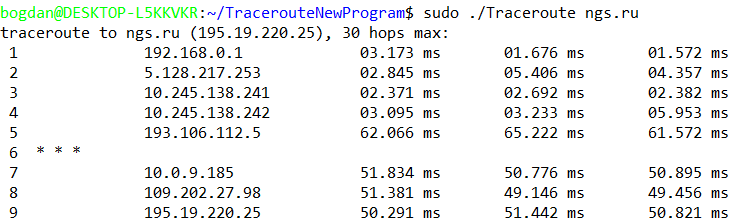
**sudo ./Traceroute ya.ru**

1. **Набор тестов**

Выполним трассировку до узлов, которые были использованы на 1-м этапе и для сайта ngs.ru







1. **Вывод**

Контрольные вопросы проработаны.