**5. ДИАГНОСТИКА TCP/IP И DNS**

**5.1. Утилиты диагностики TCP/IP и DNS**

Любая операционная система имеет набор диагностических утилит для тестирования сетевых настроек и функционирования коммуникаций. Большой набор диагностических средств есть и в системах семейства Windows (как графических, так и в режиме командной строки).

Утилиты командной строки, являющиеся инструментами первой необходимости для проверки настроек протокола TCP/IP и работы сетей и коммуникаций представлены в табл. 5.1. Подробное описание данных утилит содержится в системе интерактивной помощи Windows (вызывается нажатием кнопки *F1*). В табл. 5.1 указаны основные и наиболее часто используемые параметры этих команд, а также дано их краткое описание.

Таблица 5.1

**Утилиты диагностики TCP/IP и DNS**

|  |  |  |
| --- | --- | --- |
| Название  утилиты | Параметры | Комментарии |
| 1 | 2 | 3 |
| **ipconfig** | **/?** ­– Отобразить справку по команде  **/all** – Отобразить полную информацию о настройке параметров всех адаптеров  **/release** – Освободить динамическую IP-конфигурацию  **/renew** – Обновить динамическую IP-конфигурацию с DHCP-сервера  **/flushdns** – Очистить кэш разрешений DNS  **/registerdns** – Обновить регистрацию на DNS-сервере  **/displaydns** – Отобразить содержимое кэша разрешений DNS | Служит для отображения всех текущих параметров сети TCP/IP и обновления параметров DHCP и DNS. При вызове команды ipconfig без параметров выводятся IP-адрес, маска подсети и основной шлюз для каждого сетевого адаптера |
| **arp** | **-a** — Отображает текущие ARP-записи | Отображение и изменение ARP-таблиц |
| **ping** | Формат команды:  **«ping <*сетевой узел*> параметры»**  Параметры:  **-t** – Бесконечная (до нажатия клавиш <Ctrl>+<Break>) отправка пакетов на указанный узел | Мощный инструмент диагностики (с помощью протокола ICMP). Команда ping позволяет проверить: работоспособность IP-соединения; правильность |

Продолжение табл. 5.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
|  | **-a** – Определение имени узла по IP-адресу  **-n <число>** – Число отправляемых запросов  **-l <размер>** – Размер буфера отправки  **-w <таймаут>** – Таймаут ожидания каждого ответа в миллисекундах | настройки протокола TCP/IP на узле; работоспособность маршрутизаторов; работоспособность системы разрешения имен FQDN или NetBIOS; доступность и работоспособность какого-либо сетевого ресурса |
| **tracert** | **-d** – Без разрешения IP-адресов в имена узлов  **-h <максЧисло>** – Максимальное число прыжков при поиске узла  **-w <таймаут>** – Таймаут каждого ответа в миллисекундах | Служебная программа для трассировки маршрутов, используемая для определения пути, по которому IP-дейтаграмма доставляется по месту назначения |
| **pathping** | **-n** – Без разрешения IP-адресов в имена узлов  **-h максЧисло** – Максимальное число прыжков при поиске узла  **-q <число\_запросов>** – Число запросов при каждом прыжке  **-w <таймаут>** – Таймаут каждого ответа в миллисекундах | Средство трассировки маршрута, сочетающее функции программ *ping* и *tracert* и обладающее дополнительными возможностями. Эта команда показывает степень потери пакетов на любом маршрутизаторе или канале, с ее помощью легко определить, какие маршрутизаторы или каналы вызывают неполадки в работе сети |
| **netstat** | **-a** – Отображение *всех* подключений и ожидающих (слушающих) портов  **-n** – Отображение адресов и номеров портов в числовом формате  **-o** – Отображение кода (ID) процесса каждого подключения  **-r** – Отображение содержимого локальной таблицы маршрутов | Используется для отображения статистики протокола и текущих TCP/IP-соединений |
| **nbtstat** | **–n** – Выводит имена пространства имен NetBIOS, зарегистрированные локальными процессами  **–c** – Отображает кэш имен NetBIOS (разрешение NetBIOS-имен в IP-адреса)  **–R** – Очищает кэш имен и перезагружает его из файла Lmhosts | Средство диагностики разрешения имен NetBIOS |

Окончание табл. 5.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
|  | **–RR** – Освобождает имена NetBIOS, зарегистрированные на WINS-сервере, а затем обновляет их регистрацию |  |
| **hostname** | Никаких ключей для данной утилиты не предусмотрено. | Это самая простая утилита – она выводит на экран имя компьютера |

Рассмотрим примеры использования утилит командной строки для диагностики протокола TCP/IP и символьной адресации (DNS).

1. Использование команды *ipconfig* (без параметров и с параметром /all) представлено на рис. 5.1.

Рис. 5.1. Использование команды *ipconfig*

C:\>ipconfig

Настройка протокола IP для Windows

Подключение по локальной сети - Ethernet адаптер:

DNS-суффикс этого подключения . . :

IP-адрес . . . . . . . . . . . . : 192.168.0.1

Маска подсети . . . . . . . . . . : 255.255.255.0

Основной шлюз . . . . . . . . . . :

С:\> ipconfig /all

Настройка протокола IP для Windows

Имя компьютера . . . . . . . . . : dc1

Основной DNS-суффикс . . . . . . : world.ru

Тип узла. . . . . . . . . . . . . : неизвестный

IP-маршрутизация включена . . . . : нет

WINS-прокси включен . . . . . . . : нет

Порядок просмотра суффиксов DNS . : world.ru

Подключение по локальной сети - Ethernet адаптер:

DNS-суффикс этого подключения . . :

Описание . . . . . . . . . . . . : Realtek RTL8139 Family PCI Fast Ethernet NIC

Физический адрес. . . . . . . . . : 00-11-D8-E7-14-F4

DHCP включен. . . . . . . . . . . : нет

IP-адрес . . . . . . . . . . . . : 192.168.0.1

Маска подсети . . . . . . . . . . : 255.255.255.0

Основной шлюз . . . . . . . . . . :

DNS-серверы . . . . . . . . . . . : 192.168.0.1

2. Использование команды *arp*.

Пусть в сети только два узла (сервер DC1 и сервер DC2). Тогда в кэше сервера DC1 будет только одна запись – отображение IP-адреса сервера DC2 на MAC-адрес сетевого адаптера (рис. 5.2).

C:\>arp -a

Интерфейс: 192.168.0.1 --- 0x10003

IP-адрес Физический адрес Тип

192.168.0.2 00-03-ff-e7-14-f4 динамический

Рис. 5.2. Использование команды *arp*

3. Использование команды *ping*.

Существуют различные варианты использования данной утилиты (в сети существуют два компьютера с именами DC1 и DC2, настроена DNS-адресация):

* ping <IP-адрес> (рис. 5.3);

C:\>ping 192.168.0.2

Обмен пакетами с 192.168.0.2 по с 32 байт данных:

Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128

Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128

Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128

Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128

Статистика Ping для 192.168.0.2:

Пакетов: отправлено = 4, получено = 4, потеряно = 0

(0% потерь)

Приблизительное время приема-передачи в мс:

Минимальное = 0мсек, Максимальное = 0 мсек, Среднее = 0 мсек

|  |
| --- |
| Рис. 5.3. Использование команды *ping* с заданным IP-адресом   * ping <NetBIOS-имя узла>, когда в зоне сервера DNS нет записи для сервера DC2 (поиск IP-адреса производится широковещательным запросом) (рис. 5.4);   C:\>ping dc2  Обмен пакетами с dc2 [192.168.0.2] с 32 байт данных:  Ответ от 192.168.0.2: число байт=32 время=16мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Статистика Ping для 192.168.0.2:  Пакетов: отправлено = 4, получено = 4, потеряно = 0  (0% потерь)  Приблизительное время приема-передачи в мс:  Минимальное = 0мсек, Максимальное = 16 мсек, Среднее = 4 мсек  Рис. 5.4. Использование команды *ping* с заданным NetBIOS-именем узла (с широковещательным запросом)   * ping <NetBIOS-имя узла>, когда в зоне сервера DNS есть запись для сервера DC2 (надо обратить внимание на подстановку клиентом DNS суффикса домена в запросе на имя узла, т. е в команде используется краткое NetBIOS-имя сервера, а в статистике команды выводится полное имя) (рис. 5.5);   C:\>ping dc2  Обмен пакетами с dc2.world.ru [192.168.0.2] с 32 байт данных:  Ответ от 192.168.0.2: число байт=32 время=16мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Статистика Ping для 192.168.0.2:  Пакетов: отправлено = 4, получено = 4, потеряно = 0  (0% потерь)  Приблизительное время приема-передачи в мс:  Минимальное = 0мсек, Максимальное = 16 мсек, Среднее = 4 мсек  Рис. 5.5. Использование команды *ping* с заданным NetBIOS-именем узла (при условии существования записи на DNS-сервере)   * ping <FQDN-имя узла>, когда в зоне сервера DNS нет записи для сервера DC2 (узел DC2 не будет найдет в сети) (рис. 5.6);   C:\>ping dc2.world.ru  При проверке связи не удалось обнаружить узел dc2.world.ru. Проверьте имя узла и повторите попытку.  Рис. 5.6. Использование команды *ping* с заданным FQDN-именем узла (при условии отсутствия записи на DNS-сервере)   * ping <FQDN-имя узла>, когда в зоне сервера DNS есть запись для сервера DC2 (узел успешно найден) (рис. 5.7);   C:\>ping dc2.world.ru  Обмен пакетами с dc2.world.ru [192.168.0.2] с 32 байт данных:  Ответ от 192.168.0.2: число байт=32 время=16мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Статистика Ping для 192.168.0.2:  Пакетов: отправлено = 4, получено = 4, потеряно = 0  (0% потерь)  Приблизительное время приема-передачи в мс:  Минимальное = 0мсек, Максимальное = 16 мсек, Среднее = 4 мсек  Рис. 5.7. Использование команды *ping* с заданным FQDN-именем узла (при условии существования записи на DNS-сервере)   * ping –a <IP-адрес> (обратное разрешение IP-адреса в имя узла) (рис. 5.8).   C:\>ping -a 192.168.0.2  Обмен пакетами с dc2.world.ru [192.168.0.2] с 32 байт данных:  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Ответ от 192.168.0.2: число байт=32 время<1мс TTL=128  Статистика Ping для 192.168.0.2:  Пакетов: отправлено = 4, получено = 4, потеряно = 0  (0% потерь)  Приблизительное время приема-передачи в мс:  Минимальное = 0мсек, Максимальное = 0 мсек, Среднее = 0 мсек  Рис. 5.8. Использование команды *ping* с обратным разрешением  IP-адреса в имя узла |

4. Использование команд *tracert* и *pathping*.

На рис. 5.9 и 5.10 приведены примеры трассировки маршрута до узла www.ru (если в вашем распоряжении только одна IP-сеть, то изучить работу данных команд будет невозможно).

C:\>tracert –d www.ru

Трассировка маршрута к www.ru [194.87.0.50]

с максимальным числом прыжков 30:

1 17 ms <1 мс <1 мс 192.168.0.1

2 1 ms <1 мс 1 ms 217.1.1.33

3 3 ms 3 ms 3 ms 217.1.10.1

4 \* \* \* Превышен интервал ожидания для запроса.

5 \* \* \* Превышен интервал ожидания для запроса.

6 10 ms 11 ms 10 ms 217.150.36.190

7 \* \* \* Превышен интервал ожидания для запроса.

8 13 ms 13 ms 15 ms 194.87.0.83

9 17 ms 12 ms 12 ms 194.87.0.50

Трассировка завершена.

Трассировка завершена.

Рис. 5.9. Использование команды *tracert*

C:\> pathping –n www.ru

Трассировка маршрута к www.ru [194.87.0.50]

с максимальным числом прыжков 30:

0 192.168.0.1

1 217.1.1.33

2 217.1.10.1

3 \* \* \*

Подсчет статистики за: 100 сек. ...

Исходный узел Маршрутный узел

Прыжок RTT Утер./Отпр. % Утер./Отпр. % Адрес

0 0мс 0/ 100 = 0% 0/ 100 = 0% 192.168.0.1

0/ 100 = 0% |

1 2мс 0/ 100 = 0% 0/ 100 = 0% 217.1.1.33

0/ 100 = 0% |

2 5мс 0/ 100 = 0% 0/ 100 = 0% 217.1.10.1

100/ 100 =100% |

3 --- 100/ 100 =100% 0/ 100 = 0% 0.0.0.0

Трассировка завершена.

Рис. 5.10. Использование команды *pathping*

Отметим, что в представленных примерах приведены лишь некоторые варианты использования команд. Отработку навыков их использования с иными ключами необходимо выполнить самостоятельно (при необходимости пользоваться внутренней справкой *команда*/?).

**5.2. Разработка программы эхо-запроса для диагностики** **TCP/IP соединения**

В сетевых протоколах существуют штатные средства, позволяющие выполнять диагностику функционирования системы. При использовании протокола TCP/IP таковым средством является протокол *ICMP* (*Internet Control Message Protocol*). Он описан в RFC792 и является частью уровня IP. Сообщения ICMP инкапсулируются в IP-дейтаграммы, так что они могут распространяться по TCP/IP-сетям.

Протокол ICMP используется для построения и поддержания в актуальном состоянии таблиц маршрутизации, определения параметра PMTU (Path Maximum Transmission Unit), диагностики сетевых проблем, осуществления контроля загруженности маршрутизаторов, исследования маршрутов передачи пакетов.

Существует возможность определения доступности и работоспособности сетевого устройства через sockets (необходимы права администратора). Однако в данном разделе мы будем использовать ICMP API. Для этого необходимы три функции. Первая из них (IcmpCreateFile) создает соединение, с которым мы собираемся работать. Вторая закрывает его (IcmpCloseHandle), а третья посылает через установленное соединение соответствующие данные (IcmpSendEcho).

Перед обращением к функциям ICMP следует указать следующие директивы для подключения библиотек, содержащих структуры и функции, предоставляемые Windows Sockets API (WSA) ws2\_32.lib и IP helper API Iphlpapi.lib (подробнее о их назначении будет рассказано в следующих разделах):

|  |
| --- |
| ..............................................  #include <winsock2.h>  #include <iphlpapi.h>  #include <icmpapi.h>  #pragma comment(lib, "iphlpapi.lib")  #pragma comment(lib, "ws2\_32.lib") |

Обратите внимание, что директива #include <iphlpapi.h> должна предшествовать #include <icmpapi.h>.

Определите статусы ответов (полный перечень приведен в приложении)

#define IP\_STATUS\_BASE 11000

#define IP\_SUCCESS 0

#define IP\_DEST\_NET\_UNREACHABLE 11002

#define IP\_DEST\_HOST\_UNREACHABLE 11003

#define IP\_DEST\_PROT\_UNREACHABLE 11004

#define IP\_DEST\_PORT\_UNREACHABLE 11005

#define IP\_REQ\_TIMED\_OUT 11010

#define IP\_BAD\_REQ 11011

#define IP\_BAD\_ROUTE 11012

#define IP\_TTL\_EXPIRED\_TRANSIT 11013

Напишите функцию Ping. Первый параметр – адрес хоста, который будет пинговаться, второй – время ожидания ответа (в мс) после запроса, третий – количество посылаемых запросов:

void Ping ( const char\* cHost,

unsigned int Timeout,

unsigned int RequestCount) {

//

}

Первым шагом должно стать получение манипулятора для выполнения ICMP-запросов. Возвращается манипулятор hIP функцией IcmpCreateFile без параметров:

// Создать файл сервиса

HANDLE hIP = IcmpCreateFile();

if (hIP == INVALID\_HANDLE\_VALUE) {

WSACleanup();

return;

}

Код ошибки может быть извлечен вызовом функции WSAGetLastError (основные коды ошибок перечислены в прил.).

Выполняем подготовительные действия. Определяем данные эхо-запроса в переменной SendData. Задаем буфер для эхо-ответа pIpe и выделяем для него память. Определяем переменные для статистики LostPacketsCount, MaxMS, MinMS. Определяем размер буфера эхо-ответа как сумму размера буфера эхо-запроса и размера структуры ICMP\_ECHO\_REPL, возвращаемой в ответ на эхо-запрос и выделяем из глобальной кучи память данного размера. Сохраняем идентификатор выделенного блока глобальной памяти (pIpe). В случае ошибки вызываем функцию IcmpCloseHandle(hIP) – закрывает ICMP и функцию WSACleanup() - завершает использование Winsock 2 DLL (Ws2\_32.dll).

IP-адрес опрашиваемого узла cHost преобразуем к типу unsigned long при помощи функции inet\_addr.

char SendData[32] = "Data for ping";//буфер для передачи

int LostPacketsCount = 0; // кол-во потеряных пакетов

unsigned int MaxMS = 0;// максимальное время ответа (мс)

int MinMS = -1; // минимальное время ответа (мс)

// Выделяем память под пакет – буфер ответа

PICMP\_ECHO\_REPLY pIpe =

(PICMP\_ECHO\_REPLY)GlobalAlloc(GHND,

sizeof(ICMP\_ECHO\_REPLY)

+ sizeof(SendData));

if (pIpe == 0) {

IcmpCloseHandle(hIP);

WSACleanup();

return;

}

pIpe->Data = SendData;

pIpe->DataSize = sizeof(SendData);

unsigned long ipaddr = inet\_addr(cHost);

Также необходимо настроить опции для включения в заголовок IP пакета. Например:

IP\_OPTION\_INFORMATION option = { 255, 0, 0, 0, 0 };

Здесь структура IP\_OPTION\_INFORMATION (для 64-разрядных систем используется IP\_OPTION\_INFORMATION32) имеет следующие параметры:

typedef struct {

unsigned char Ttl; //время доставки

unsigned char Tos; //тип сервиса

unsigned char Flags; //флаги IP заголовка

unsigned char OptionsSize;

//размер опций в батах

unsigned char \*OptionsData;

//указатель на опции

} IP\_OPTION\_INFORMATION, \*PMP\_OPTION\_INFORMATION;

Третий шаг – приступаем к отправке эхо-запросов. Для этих целей служит функция IcmpSendEcho со следующим прототипом:

DWORD IcmpSendEcho

(

\_\_in HANDLE IcmpHandle,

//дискриптор полученный

//функцией IcmpCreateFile

\_\_in IPAddr DestinationAddress,

//IP адрес запроса

\_\_in LPVOID RequestData,

//укзатель на буффер эхо-запроса

\_\_in WORD RequestSize,

//размер буффера эхо-запроса

\_\_in PIP\_OPTION\_INFORMATION RequestOptions,

//укзатель на структуру с опициями запроса

\_\_inout LPVOID ReplyBuffer,

//укзатель на буффер эхо-ответа

\_\_in DWORD ReplySize,

//размер буффера эхо-ответа

\_\_in DWORD Timeout

//таймаут в миллисекундах

);

Она отправляет ICMP эхо-запрос по указанному IP-адресу и возвращает любые ответы, полученные в пределах заданного тайм-аута (Timeout) и до исчерпания пространства буфера. Эта функция синхронна (то есть приостанавливает выполнение процесса до завершения своей работы), и во избежание блокировки процесс должен перед ее вызовом породить соответствующую нить. Функция IcmpSendEcho использует для передачи IPv4 ICMP. Существуют еще версии функции: IcmpSendEcho2Ex и IcmpSendEcho2. Для IPv6, используйте функции Icmp6CreateFile и Icmp6SendEcho2.

for (unsigned int c = 0; c < RequestCount; c++) {

int dwStatus = IcmpSendEcho( hIP,

ipaddr,

SendData,

sizeof(SendData),

&option,

pIpe,

sizeof(ICMP\_ECHO\_REPLY) +

sizeof(SendData),

Timeout);

if (dwStatus > 0) {

for (int i = 0;i< dwStatus;i++) {

unsigned char\* pIpPtr =

(unsigned char\*)&pIpe->Address;

cout << "Ответ от " << (int)\*(pIpPtr)

<< "." << (int)\*(pIpPtr + 1)

<< "." << (int)\*(pIpPtr + 2)

<< "." <<(int) \*(pIpPtr + 3)

<< ": число байт = " << pIpe->DataSize

<< " время = " << pIpe->RoundTripTime

<< "мс TTL = " << (int)pIpe->Options.Ttl;

MaxMS = (MaxMS > pIpe->RoundTripTime) ?

MaxMS : pIpe->RoundTripTime;

MinMS = (MinMS < (int)pIpe->RoundTripTime

&& MinMS >= 0) ?

MinMS : pIpe->RoundTripTime;

cout << endl;

}

}

Первым параметром функции IcmpSendEcho служит манипулятор, полученный на первом шаге инициализации hIP. Вторым — IP-адрес опрашиваемого узла ipaddr. Третьим и четвертым параметрами являются указатель на отправляемые в эхо-запросе данные SendData и размер этих данных соответственно sizeof(SendData). Пятый параметр является указателем option на структуру, содержащую дополнительные опции запроса (в том числе и TTL – Time to live – время жизни пакета). Его можно установить в NULL. Шестой и седьмой параметры описывают соответственно указатель на буфер для эхо-ответов pIpe и размер этого буфера. Тайм-аут указывается восьмым параметром (Timeout) и является входным параметром функции Ping.

Возвращаемое функцией IcmpSendEcho значение dwStatus представляет собой количество полученных эхо-ответов. По завершении работы функции IcmpSendEcho параметр pIpe, буфер ответа, будет заполнен массивом структур типа ICMP\_ECHO\_REPLY, за которым последуют опции и данные ответов. Он должен быть достаточно большим, чтобы разместить хотя бы одну структуру ICMP\_ECHO\_REPLY (для 64-разрядных – ICMP\_ECHO\_REPLY32) плюс Max (sizeof(SendData), 8) байт, так как сообщение об ошибке ICMP занимает 8 байт. Структура ICMP\_ECHO\_REPLY описывается следующим образом:

typedef struct icmp\_echo\_reply

{

IPAddr Address;

// адрес ответившего узла

ULONG Status;

// статус ответа

ULONG RoundTripTime;

// время прохождения запроса в мс

USHORT DataSize;

// размер данных ответа

USHORT Reserved;

// зарезервировано

PVOID Data;

// указатель на данные ответа

Struct IP\_OPTION\_INFORMATION Options;

// опции ответа

} ICMP\_ECHO\_REPLY, \*PICMP\_ECHO\_REPLY;

Обращаясь к полям структуры выводим на консоль IP-адрес, время, прошедшее с момента отправки эхо-запроса до получения эхо-ответа (RTT), число переданных байт и т.п. Фиксируем максимальное и минимальное время.

В случае ошибки считаем число потерянных пакетов и выводим код ошибки:

else {

if (pIpe->Status) {

LostPacketsCount++;

switch (pIpe->Status) {

case IP\_DEST\_NET\_UNREACHABLE:

case IP\_DEST\_HOST\_UNREACHABLE:

case IP\_DEST\_PROT\_UNREACHABLE:

case IP\_DEST\_PORT\_UNREACHABLE:

cout<<"Remote host may be down."<<endl;

break;

case IP\_REQ\_TIMED\_OUT:

cout<< "Request timed out."<<endl;

break;

case IP\_TTL\_EXPIRED\_TRANSIT:

cout<< "TTL expired in transit."<<endl;

break;

default:

cout<< "Error code: %ld"

<< pIpe->Status<<endl;

break;

}

}

Для корректного завершения работы с ICMP-функциями с помощью вызова IcmpCloseHandle() освобождается ICMP-манипулятор:

IcmpCloseHandle(hIP);

WSACleanup();

В заключении можно вывести статистику:

if (MinMS<0) MinMS = 0;

unsigned char\* pByte = (unsigned char\*)&pIpe->Address;

cout<<"Статистика Ping "

<< (int)\*(pByte)

<< "." << (int)\*(pByte + 1)

<< "." << (int)\*(pByte + 2)

<< "." << (int)\*(pByte + 3)<<endl;

cout << "\tПакетов: отправлено = "<< RequestCount

<< ", поучено = "

<< RequestCount - LostPacketsCount

<< ", потеряно = " << LostPacketsCount

<< "<" <<(int)(100/(float)RequestCount)\*

LostPacketsCount

<< " % потерь>, "<<endl;

cout<<"Приблизительное время приема-передачи:"

<<endl<< "Минимальное = " << MinMS

<< "мс, Максимальное = " << MaxMS

<< "мс, Среднее = " << (MaxMS + MinMS) / 2

<< "мс" << endl;

Вызов функции выглядит следующим образом:

int main(int argc, char \*\*argv)

{

setlocale(LC\_ALL, "RUS");

Ping("81.19.70.1", 60, 10);

getch();

return 0;

}

Результат работы программы представлен на рис. 5.11:

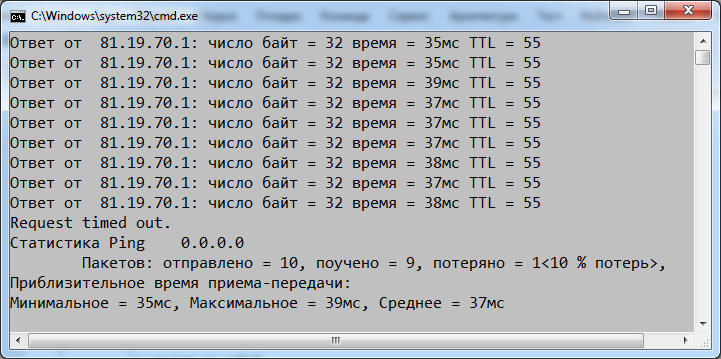


Рис. 5.11. Результат работы программы

**5.3. Лабораторная работа №7**

**Цель:** Изучение методов диагностики TCP/IP сетей.

**Задание:** Лабораторная работа состоит из двух частей:

* + - 1. Изучение использования команд *ipconfig*, *ping*, *tracert*, *pathping, arp, nbtstat* с использованием различных ключей в клиент-серверной сети с настроенной динамической адресацией и DNS-сервером. Приобретение навыков определения символьных и физических адресов удаленных компьютеров по известному IP-адресу;
      2. Разработка программы, реализующей функции эхозапроса; проверка ее работы в клиент-серверной сети, настроенной в предыдущих лабораторных работах.
      3. Поменяйте параметры  *IP\_OPTION\_INFORMATION option.*
      4. Переделайте программу таким образом, чтобы параметры функции Ping считывались через аргументы командной строки. Продемонстрируйте выполнение программы.