# 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 таблице 5.1. Подробное описание данных утилит содержится в системе интерактивной помощи Windows (вызывается нажатием кнопки *F1*). В таблице 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 |

Продолжение табл. 5.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| **ping** | Формат команды:  **«ping<*сетевой узел*> параметры»**  Параметры:  **-t** – Бесконечная (до нажатия клавиш <Ctrl>+<Break>) отправка пакетов на указанный узел  **-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 | Мощный инструмент диагностики (с помощью протокола ICMP). Команда ping позволяет проверить: работоспособность IP-соединения; правильность 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 до места 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 |

Окончание табл. 5.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| **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  **–RR** – Освобождает имена NetBIOS, зарегистрированные на WINS-сервере, а затем обновляет их регистрацию | Средство диагностики разрешения имен NetBIOS |
| **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.

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).

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 . . . . . . . . . . . . :RealtekRTL8139 FamilyPCIFastEthernetNIC

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

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);
* ping<NetBIOS-имя узла>, когда в зоне сервера DNS нет записи для сервера DC2 (поиск IP-адреса производится широковещательным запросом) (рис. 5.4);
* 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);
* ping<FQDN-имя узла>, когда в зоне сервера DNS нет записи для сервера DC2 (узел DC2 не будет найдет в сети) (рис. 5.6);
* ping<FQDN-имя узла>, когда в зоне сервера DNS есть запись для сервера DC2 (узел успешно найден) (рис. 5.7);
* ping -a <IP-адрес> (обратное разрешение IP-адреса в имя узла) (рис. 5.8).

|  |
| --- |
| Рис. 5.3. Использование команды *ping*с заданным IP-адресом  Рис. 5.4. Использование команды *ping* с заданным NetBIOS-именем узла (с широковещательным запросом)  C:\>pingdc2  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  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 = 0мсек, Среднее = 0 мсек  Рис. 5.5. Использование команды *ping* с заданным NetBIOS-именем узла (при условии существования записи на DNS-сервере)  C:\>pingdc2.world.ru  При проверке связи не удалось обнаружить узел dc2.world.ru. Проверьте имя узла иповторите попытку.  C:\>pingdc2  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. 5.6. Использование команды *ping* с заданным FQDN-именем узла (при условии отсутствия записи на DNS-сервере)  C:\>pingdc2.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)  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 –dwww.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 –nwww.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*

## 5.2. Разработка программы эхо-запроса для диагностики TCP/IPсоединения

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. В данном разделе мы будем использовать протокол ICMP для определения доступности и работоспособности сетевого устройства.

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 APIIphlpapi.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>.

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

|  |
| --- |
| HANDLE hIcmpFile;  ..............................................  hIcmpFile = IcmpCreateFile();  if (hIcmpFile == INVALID\_HANDLE\_VALUE)  {  printf("\tНевозможно открыть дискриптор.\n");  printf(" Ошибка IcmpCreatefile: %ld\n",  WSAGetLastError () );  return -1;  } |

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

Выполняем подготовительные действия. Задаем IP-адрес опрашиваемого узла в ADDR и преобразуем строку к сетевому формату при помощи функции inet\_addr. Определяем данные эхо-запроса в SendData. Определяем размер буфера эхо-ответа ReplySize как сумму размера буфера эхо-запроса и размера структуры ICMP\_ECHO\_REPL, возвращаемой в ответ на эхо-запрос Создаем буфер для эхо-ответа ReplyBuffer и выделяем для него память:

|  |
| --- |
| #define ADDR "81.19.70.1" //IP- адрес  ..............................................  unsigned long ipaddr = inet\_addr(ADDR);  //преобразование IP-адреса  char SendData[32] = "Data Buffer";  //данные эхо-запроса  LPVOID ReplyBuffer = NULL;  //данные эхо-ответа  DWORDReplySize = 0;  //размер данных эхо-ответа  ..............................................  ReplySize = sizeof(ICMP\_ECHO\_REPLY) + sizeof(SendData);  //Выделяем память  ReplyBuffer = (VOID\*) malloc(ReplySize);  if (ReplyBuffer == NULL)  {  printf("\tНевозможно выделить память\n");  return -1;  } |

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

|  |
| --- |
| DWORD IcmpSendEcho  (  \_\_in HANDLE IcmpHandle,  //дискрипторполученный  //функцией IcmpCreateFile  \_\_in IPAddr DestinationAddress,  //IP адресзапроса  \_\_in LPVOID RequestData,  //укзательнабуфферэхо-запроса  \_\_inWORDRequestSize,  //размер буффера эхо-запроса  \_\_in PIP\_OPTION\_INFORMATION RequestOptions,  //укзатель на структуру с опициями запроса  \_\_inoutLPVOIDReplyBuffer,  //укзательнабуфферэхо-ответа  \_\_inDWORD ReplySize,  //размер буффера эхо-ответа  \_\_in DWORD Timeout  //таймаут в миллисекундах  ); |

IcmpSendEcho отправляет ICMP эхо-запрос по указанному IP-адресу и возвращает любые ответы, полученные в пределах заданного тайм-аута (Timeout) и до исчерпания пространства буфера ReplyBuffer. Эта функция синхронна (то есть приостанавливает выполнение процесса до завершения своей работы), и во избежание блокировки процесс должен перед ее вызовом породить соответствующую нить.

|  |
| --- |
| dwRetVal =  IcmpSendEcho(hIcmpFile, ipaddr, SendData,  sizeof(SendData), NULL,  ReplyBuffer, ReplySize, 1000);  if (dwRetVal != 0)  {  PICMP\_ECHO\_REPLY pEchoReply =  (PICMP\_ECHO\_REPLY)ReplyBuffer;  struct in\_addr ReplyAddr;  ReplyAddr.S\_un.S\_addr = pEchoReply->Address;  printf("\tПосылка icmp сообщения на %s\n",  ADDR);  if (dwRetVal> 1) {  printf("\tПолучен %ldicmp ответ\n",  dwRetVal);  printf("\tИнформация:\n");  }  else {  printf("\tПолучен %ld icmp ответ\n",  dwRetVal);  printf("\tИнформация:\n");  }  printf("\tПолученоот %s\n",  inet\_ntoa( ReplyAddr ) );  printf("\t Статус = %ld\n",  pEchoReply->Status);  printf("\tВремяотклика = %ldмиллисекунд \n",  pEchoReply->RoundTripTime);  }  else {  printf("\tВызов IcmpSendEcho завершился с  ошибкой.\n");  printf("\tIcmpSendEcho ошибка: %ld\n",  WSAGetLastError () );  return -1;  } |

Первым параметром функции IcmpSendEcho служит манипулятор, полученный на втором шаге инициализацииhIcmpFile. Вторым – IP-адрес опрашиваемого узлаipaddr. Третьим и четвертым параметрами являются указатель на отправляемые в эхо-запросе данные SendData и размер этих данных sizeof(SendData)соответственно. Пятый параметр является указателем на структуру, содержащую дополнительные опции запроса (в том числе и TTL–Timetolive– время жизни пакета). Установим этот параметр в NULL. Шестой и седьмой параметры описывают соответственно указатель на буфер для эхо-ответовReplyBuffe и размер этого буфераReplySize. Тайм-аутуказывается восьмым параметром и будет задан 1000 миллисекунд.

Одно из возвращаемых функцией IcmpSendEcho значение –dwRetVal– представляет собой количество полученных эхо-ответов. По завершении работы функции IcmpSendEcho буфер, на который указывает ReplyBuffer, будет заполнен массивом структур типа ICMP\_ECHO\_REPLY, за которым последуют опции и данные ответов. Буфер должен быть достаточно большим, чтобы разместить хотя бы одну структуру ICMP\_ECHO\_REPLY плюсMax (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  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; |

Наибольший интерес для нашей цели представляет поле RoundTripTime, с помощью которого можно определить время, прошедшее с момента отправки эхо-запроса до получения эхо-ответа (RTT). Выводим на консоль IP-адрес, статус и время отклика или в случае ошибки – код ошибки.

Для корректного завершения работы с ICMP-функциями с помощью вызоваIcmpCloseHandle (hIcmpFile) освобождается ICMP-манипулятор:

|  |
| --- |
| BOOLbRetVal;  ..............................................  //освобождаем дескриптор  bRetVal = IcmpCloseHandle(hIcmpFile);  if (bRetVal)  printf("\tHandle was closed\n");  else  printf("IcmpCloseHandle failed with error: %ld\n",  WSAGetLastError () );  } |

Результат работы программы представлен на рис. 5.11:

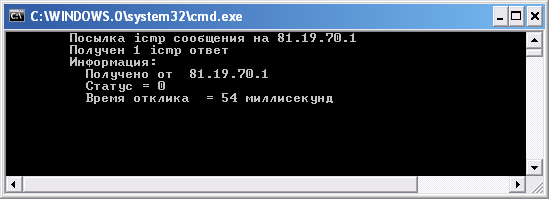


Рис. 5.11. Результат работы программы