

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
Владимирский государственный университет имени А.Г. и Н.Г. Столетовых

Кафедра информационных систем
и программной инженерии

Х.М. Салех
**ИНФОКОММУНИКАЦИОННЫЕ
СИСТЕМЫ И СЕТИ**

Методические указания к лабораторным работам

Владимир 2015 г

Содержание

ЛАБОРАТОРНАЯ РАБОТА №1.....	3
ЛАБОРАТОРНАЯ РАБОТА №2.....	53
ЛАБОРАТОРНАЯ РАБОТА №3.....	75
ЛАБОРАТОРНАЯ РАБОТА №4.....	102
ЛАБОРАТОРНАЯ РАБОТА №5.....	118
ЛАБОРАТОРНАЯ РАБОТА №6.....	139

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
Владимирский государственный университет имени А.Г. и Н.Г. Столетовых

Кафедра информационных систем
и программной инженерии

ЛАБОРАТОРНАЯ РАБОТА №1 БАЗОВАЯ НАСТРОЙКА СЕТЕВЫХ СРЕДСТВ ОПЕРАЦИОННОЙ СИСТЕМЫ

Владимир 2015 г

1. Цель работы

Получить навыки базовой настройки сетевых средств ОС Windows.

2. Работа с командной строкой Windows

Консоль командной строки присутствует во всех версиях операционных систем Windows. Ранние версии ОС поддерживали режим MS-DOS напрямую, что позволяло выполнять простые команды прямо из консоли. Представители же семейства NT, такие как Windows 2000 или Windows Server 2003, работают уже совсем по другим принципам, однако MS-DOS в них тоже поддерживается, но через виртуальную машину (NT Virtual DOS Machine, NTVDM), что позволяет контролировать и администрировать системные ресурсы прямо из консоли командного режима. В качестве интерпретатора командного режима выступает программа `cmd.exe`, запуск которой осуществляется через меню «Пуск > Выполнить». Кроме того, для запуска консоли можно воспользоваться элементом меню «Пуск -> Программы -> Стандартные -> Командная строка».

Запустив консоль командного режима, пользователь может управлять ресурсами как локальной системы, так и ресурсами удаленной машины. Существуют команды, выполняющие мониторинг системы и выявляющие критические места в настройках сервера. Отличием работы из командной строки является полное отсутствие больших и громоздких графических утилит. Программы командной строки позволяют более тонкую настройку в виде параметров-ключей, указанных справа от самой команды.

С помощью специальных файлов-скриптов (наборов команд, выполняющихся последовательно или в запрограммированном порядке) администратор может свести к минимуму выполнение рутинных ежедневных операций. Существующие современные утилиты могут запускать такие скрипты с заданной периодичностью без присутствия администратора системы.

Сам администратор может выполнять как одиночные команды, так и список команд, используя специальные управляющие символы (&, |). Например:

Команда 1 & Команда 2 — сначала будет выполнена Команда 1 и только затем Команда 2;

Команда 1 && Команда 2 — только после успешного выполнения Команды 1 будет запущена Команда 2.

Существует возможность перенаправить выводимый программой поток напрямую в текстовый файл для дальнейшей обработки. Для этого необходимо использовать управляющий символ «>» и имя текстового файла. Пример вывода содержания текущего каталога в текстовый файл Report.txt при помощи команды dir приведен ниже:

```
dir> Report.txt
```

В современных операционных системах существует множество команд и утилит. Запомнить такое количество различных программ, а тем более их параметров очень сложно, поэтому одним из самых важных параметров для каждой программы является сочетание символов `/?`. Выполнив команду с таким параметром, пользователь получит исчерпывающее сообщение о применении утилиты и синтаксисе ее параметров.

Для того чтобы закрыть консоль командной строки, необходимо выполнить команду `exit`.

2.1. Утилиты командной строки

Большинство рассматриваемых сетевых утилит для полноценной работы требуют наличия административных привилегий. Для операционных систем семейства Windows 2000/XP достаточно того, чтобы пользователь работал под учетной записью члена группы администраторов. Интерпретатор командной строки `cmd.exe` можно запустить с использованием **меню Пуск - Выполнить - cmd.exe**. В среде операционных систем Windows Vista/Windows 7 интерпретатор `cmd.exe` должен быть запущен для выполнения с использованием пункта контекстного меню "Запустить от имени администратора". Командные файлы, в которых используются сетевые утилиты, также должны выполняться в контексте учетной записи с привилегиями администратора.

В списке представлены сетевые утилиты командной строки для получения информации о сетевых настройках, выполнения операций по конфигурированию и диагностике сети.

В описании команд используется

< текст > - текст в угловых скобках. Обязательный параметр.

[текст] - текст в квадратных скобках. Необязательный параметр.

(текст) - текст в круглых скобках. Необходимо выбрать один из параметров.

Вертикальная черта | - разделитель для взаимоисключающих параметров. Нужно выбрать один из них.

Многоточие ... - возможно повторение параметров.

2.2. Основные утилиты:

1. Hostname

Одна из основных утилит TCP/IP. Она выводит имя системы, на которой запущена команда: C:\>hostname

2. Ping

Команда Ping лежит в основе диагностики сетей TCP/IP. Если до системы не удастся «достучаться» с помощью этой команды, вероятнее всего, с такой системой связаться не удастся.

Синтаксис:

ping [-t] [-a] [-n число] [-l размер] [-f] [-i TTL] [-v TOS] [-r число] [-s число] [[-j списокУзлов] | [-k списокУзлов]] [-w таймаут] конечноеИмя

Параметры:

-t	Отправка пакетов на указанный узел до команды прерывания. Для вывода статистики и продолжения нажмите <Ctrl>+<Break>, для прекращения - <Ctrl>+<C>
-a	Определение адресов по именам узлов
-n число	Число отправляемых запросов

-i TTL	Задание срока жизни пакета (поле "Time To Live")
-г число	Запись маршрута для указанного числа переходов
-j списокУзлов	Свободный выбор маршрута по списку узлов
-k списокУзлов	Жесткий выбор маршрута по списку узлов
-w таймаут	Таймаут каждого ответа в миллисекундах

Примеры:

Чтобы опросить станцию с IP-адресом 192.168.100.1, следует набрать:
C:\>ping 192.168.100.1

3. Tracert

Эта команда используется для верификации пути через маршрутизатор между данной станцией и удаленной. Tracert фиксирует число переходов или «прыжков» (hop), которые потребовалось совершить на пути к станции назначения.

Синтаксис:

tracert [-d] [-h максЧисло] [-j списокУзлов] [-w интервал] имя

Параметры:

-d	Без разрешения в имена узлов.
-h максЧисло	Максимальное число прыжков при поиске узла
-j списокУзлов	Свободный выбор маршрута по списку узлов
-w интервал	Интервал ожидания каждого ответа в миллисекундах

Примеры:

Например, чтобы вывести трассу маршрута к <http://www.vtsnet.ru>, нужно набрать: C:\>tracert www.vtsnet.ru

Трассировка маршрута к www.vtsnet.ru [81.27.51.133] с максимальным числом прыжков 30:

```

1  <1 мс  <1 мс  <1 мс  192.168.128.100
2  14 ms  15 ms  13 ms  192.168.149.217
3  17 ms  18 ms  18 ms  192.168.194.253
4  28 ms  22 ms  20 ms  www3.vtsnet.ru [81.27.51.133]
```

Трассировка завершена.

4. Pathping

Одна из самых полезных новых команд диагностики TCP/IP. Она объединяет функциональность Ping и Tracert. Команда Pathping опрашивает каждый маршрутизатор на пути между источником и приемником сигнала, после чего фиксирует задержки при каждой ретрансляции сигнала и потери пакетов.

Синтаксис:

pathping [-g Список] [-h Число_прыжков] [-i Адрес] [-n] [-p Пауза]
[-q Число_запросов] [-w Таймаут] [-P] [-R] [-T] [-4] [-6] узел

Параметры:

-h Число_прыжков	Максимальное число прыжков при поиске узла
-p Пауза	Пауза между отправками (мсек)
-q Число_запросов	Число запросов при каждом прыжке
-w Таймаут	Время ожидания каждого ответа (мсек)
-4	Принудительно использовать IPv4
-6	Принудительно использовать IPv6

Примеры:

Для использования Pathping при тестировании <http://www.vtsnet.ru> необходимо набрать:

C:\Documents and Settings\avk.SMART>pathping www.vtsnet.ru

Трассировка маршрута к www.vtsnet.ru [81.27.51.133] с максимальным числом прыжков 30:

```
0  jose.smart.inreco.ru [192.168.128.50]
1  192.168.128.100
2  192.168.149.217
3  192.168.194.253
4  www3.vtsnet.ru [81.27.51.133]
```


Подсчет статистики за: 100 сек. ...

Исходный узел	Маршрутный узел	Прыжок	RTT	Утер./Отпр. %
Утер./Отпр. %	Адрес			

0	jose.smart.inreco.ru
---	----------------------

[192.168.128.50]

0/ 100 = 0%		1	0мс	0/ 100 = 0%	0/ 100 = 0%	192.168.128.100
-------------	--	---	-----	-------------	-------------	-----------------

0/ 100 = 0%		2	14мс	1/ 100 = 1%	1/ 100 = 1%	192.168.149.217
-------------	--	---	------	-------------	-------------	-----------------

0/ 100 = 0%		3	19мс	0/ 100 = 0%	0/ 100 = 0%	192.168.194.253
-------------	--	---	------	-------------	-------------	-----------------

1/ 100 = 1%		4	22мс	1/ 100 = 1%	0/ 100 = 0%	www3.vtsnet.ru
-------------	--	---	------	-------------	-------------	----------------

[81.27.51.133] Трассировка завершена.

5. Ipconfig

Эта команда отображает текущие настройки TCP/IP. Кроме того, Ipconfig может вывести отчет об адресах серверов DNS: C:\>ipconfig /all

Синтаксис:

ipconfig [/? | /all | /release [адаптер] | /renew [адаптер] | /flushdns | /displaydns /registerdns /showclassid адаптер /setclassid адаптер [устанавливаемый_код_класса_dhcp]] Где адаптер Полное имя или имя, содержащие подстановочные знаки "*" и "?" (* - любое количество знаков, ? - один любой знак). См. примеры

Параметры:

/all	Отобразить полную информацию о настройке параметров
/release	Освободить IP-адрес для указанного адаптера
/renew	Обновить IP-адрес для указанного адаптера
/registerdns	Обновить все DHCP-аренды и перерегистрировать DNS-имена

По умолчанию отображается только IP-адрес, маска подсети и стандартный шлюз для каждого подключенного адаптера, для которого выполнена привязка с TCP/IP.

Для ключей /Release и /Renew, если не указано имя адаптера, то будет освобожден или обновлен IP-адрес, выданный для всех адаптеров, для которых существуют привязки с TCP/IP.

Примеры:

> ipconfig	Отображает краткую информацию
> ipconfig /all	Отображает полную информацию
> ipconfig /renew	Обновляет сведения для всех адаптеров
> ipconfig /renew EL*	Обновляет сведения для адаптеров, начинающихся с EL....
> ipconfig /release *ELINK?21*	Освобождает IP-адреса для всех адаптеров, имена которых удовлетворяют запросу: ELINK-21 или myELELINKi21adapter и т.п.

6. Netstat

Утилита netstat присутствует во всех версиях Windows, однако, существуют некоторые отличия используемых параметров командной строки и результатов ее выполнения, в зависимости от операционной системы. Используется для отображения TCP и UDP -соединений, слушаемых портов, таблицы маршрутизации, статистических данных для различных протоколов.

Команда Netstat показывает текущий статус и статистику подключений по TCP/IP или UDP. При этом выводятся данные как о локальных, так и об удаленных именах и портах активных сетевых соединений. Ключ ? показывает все доступные ключи при работе с Netstat.

Синтаксис:

netstat [-a] [-b] [-e] [-n] [-o] [-p протокол] [-r] [-s] [-v] [интервал]

Параметры:

-a	Отображение всех подключений и ожидающих портов
-b	Отображение исполняемого файла, участвующего в создании каждого подключения, или ожидающего порта. Иногда известные исполняемые файлы содержат множественные независимые компоненты. Тогда отображается последовательность компонентов, участвующих в создании подключения, либо ожидающий порт. В

	этом случае имя исполняемого файла находится снизу в скобках [], сверху - компонент, который им вызывается, и так до тех пор, пока не достигается ТСП/IP. Заметьте, что такой подход может занять много времени и требует достаточных разрешений.
-e	Отображение статистики Ethernet. Он может применяться вместе с параметром -s
-n	Отображение адресов и номеров портов в числовом формате
-o	Отображение соединений, включая идентификатор процесса (PID) для каждого соединения
-p	протокол Отображение соединений для заданного протокола. Протокол может принимать значения tcp, udp, tcpv6, udpv6 . При использовании совместно с параметром -s в качестве протокола можно задавать tcp, udp, icmp, ip, tcpv6, udpv6, icmpv6, ipv6
-r	Отображение содержимого таблицы маршрутов
-s	Отображение статистических данных по протоколам. По умолчанию данные отображаются для IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP и UDPv6. Параметр -p позволяет указать подмножество выводимых данных
-v	Отображать подробную информацию. При использовании с параметром -b, отображает последовательность компонентов, участвующих в создании подключения, или ожидающий порт для всех исполняемых файлов. интервал Интервал обновления отображаемой информации в секундах. Повторный вывод статистических данных через указанный промежуток времени в секундах. Для прекращения вывода данных нажмите клавиши CTRL+C. Если параметр не задан, сведения о текущей конфигурации выводятся один раз
/?	Отобразить справку по использованию netstat

Команды:

При использовании утилиты netstat.exe удобно пользоваться командами постраничного вывода (more), перенаправления стандартного вывода в файл (>) и поиска текста в результатах (find). netstat -a | more - отобразить все соединения в постраничном режиме вывода на экран. netstat -a >

C:\netstatall.txt - отобразить все соединения с записью результатов в файл C:\netstatall.txt. netstat -a | find /I "LISTENING" - отобразить все соединения со статусом LISTENING. Ключ /I в команде find указывает, что при поиске текста не нужно учитывать регистр символов. netstat -a | find /I "listening" > C:\listening.txt - отобразить все соединения со статусом LISTENING с записью результатов в файл C:\listening.txt.

Пример отображаемой информации:

Активные подключения

Имя	Локальный адрес	Внешний адрес	Состояние
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING
[httpd.exe]			
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
Не удастся получить сведения о владельце			
TCP	0.0.0.0:5800	0.0.0.0:0	LISTENING
[WinVNC.exe]			
TCP	127.0.0.1:50197	127.0.0.1:50198	ESTABLISHED
[firefox.exe]			
UDP	192.168.0.107:1900	*:*	
SSDPSRV [svchost.exe] . . .			

Имя - название протокола.

Локальный адрес - локальный IP-адрес участвующий в соединении или связанный со службой, ожидающей входящие соединения (слушающей порт). Если в качестве адреса отображается 0.0.0.0 , то это означает - "любой адрес", т.е в соединении могут использоваться все IP-адреса существующие на данном компьютере. Адрес 127.0.0.1 - это петлевой интерфейс, используемый в качестве средства IP протокола для взаимодействия между процессами без реальной передачи данных.

Внешний адрес - Внешний IP-адрес, участвующий в создании соединения.

Состояние - состояние соединения. Состояние Listening говорит о том, что строка состояния отображает информацию о сетевой службе, которая ожидает входящие соединения по соответствующему протоколу на адрес и порт, отображаемые в колонке "Локальный адрес ". Состояние **ESTABLISHED** указывает на активное соединение. В колонке "Состояние" для соединений по протоколу TCP может отображаться текущий этап TCP-сессии определяемый по обработке значений флагов в заголовке TCP - пакета (Syn, Ask, Fin ...). Возможные состояния:

CLOSE_WAIT - ожидание закрытия соединения.
CLOSED - соединение закрыто.
ESTABLISHED - соединение установлено.
LISTENING - ожидается соединение (слушается порт)
TIME_WAIT - превышение времени ответа.

Имя программного модуля, связанного с данным соединением отображается, если задан параметр -b в командной строке при запуске netstat.exe.

Примеры:

- Чтобы вывести все активные подключения, отсортированные по возрастанию номера порта, необходимо набрать: C:\>netstat -n
- Получить список слушаемых портов и связанных с ними программ:
 - netstat -a -b
 - netstat -ab - параметры командной строки можно объединять. Параметр -ab эквивалентен -a -b netstat -a -n -b - отобразить список всех соединений с числовыми номерами портов
 - netstat -anb - аналогично предыдущей команде.
 - netstat -anbv - при использовании параметра -v отображается последовательность компонентов, участвующих в создании соединения или слушаемого порта.
- Получить статистические данные:
 - netstat -e - получить статистические данные для Ethernet. Отображаются суммарные значения принятых и полученных байт для всех сетевых адаптеров.
 - netstat -e -v - кроме статистических данных, отображается информация о каждом сетевом адаптере.
 - netstat -e -s - дополнительно к статистике Ethernet, отображается статистика для протоколов IP , ICMP , TCP , UDP
 - netstat -s -p tcp udp - получить статистику только по протоколам TCP и UDP

7. Route

Эта команда нужна для редактирования или просмотра таблицы маршрутов IP из командной строки. Windows 2000 использует таблицу маршрутов в том случае, когда нужно отыскать путь к удаленному компьютеру по TCP/IP. Ключ ? выводит все доступные ключи при работе с Route.

Синтаксис:

route [-f] [-p] [команда [узел][MASK маска] [шлюз]
[METRIC метрика] [IF-интерфейс]

Параметры:

-f	Очистка таблиц маршрутов от записей для всех шлюзов. При указании одной из команд, таблицы очищаются до выполнения команды
-p	При использовании с командой ADD задает сохранение маршрута при перезагрузке системы. По умолчанию маршруты не сохраняются при перезагрузке. Игнорируется для остальных команд, изменяющих соответствующие постоянные маршруты. Этот параметр не поддерживается в Windows 95

команда *Одна из четырех команд:*

- PRINT -Печать маршрута
- ADD -Добавление маршрута
- DELETE-Удаление маршрута
- CHANGE-Изменение существующего маршрута _узел_ Адресуемый узел.

METRIC - Определение параметра метрика/цена для адресуемого узла.

Примеры:

Для просмотра таблицы маршрутов системы используется Route Print:
C:\>route print

8. Arp

Команда Arp используется для просмотра, добавления или удаления записей в таблицах трансляции адресов IP в физические адреса. Эти записи используются при работе протокола Address Resolution Protocol (ARP).

Синтаксис:

Отображение и изменение таблиц преобразования IP-адресов в физические, используемые протоколом разрешения адресов (ARP).

ARP -s inet_addr eth_addr [if_addr]

ARP -d inet_addr [if_addr]

ARP -a [inet_addr] [-N if_addr]

Параметры:

-a	Отображает текущие ARP-записи, опрашивая текущие данные протокола. Если задан inet_addr, то будут отображены IP и физический адреса только для заданного компьютера. Если более одного сетевого интерфейса используют ARP, то будут отображаться записи для каждой таблицы
-g	То же, что и ключ -a
inet_addr	Определяет IP-адрес
-N if_addr	Отображает ARP-записи для заданного в if_addr сетевого интерфейса
-d	Удаляет узел, задаваемый inet_addr. inet_addr может содержать символ шаблона * для удаления всех узлов
-s	Добавляет узел и связывает internet адрес inet_addr с физическим адресом eth_addr. Физический адрес задается 6 байтами (в шестнадцатеричном виде), разделенных дефисом. Эта связь является постоянной
eth_addr	Определяет физический адрес
if_addr	Если параметр задан, - он определяет интернет адрес интерфейса, чья таблица преобразования адресов должна измениться. Если не задан, - будет использован первый доступный интерфейс

Примеры:

arp -s 157.55.85.212 00-aa-00-62-c6-09

... Добавляет статическую запись.

arp -a

.. Выводит ARP-таблицу.

Чтобы просмотреть содержимое занесенных в кэш адресов IP и MAC-адресов конкретной системы, нужно набрать:

C:\>arp —a

9. Getmac

Утилита командной строки GETMAC присутствует в версиях Windows XP и старше. Используется для получения аппаратных адресов сетевых адаптеров (MAC-адресов) как на локальном, так и на удаленном компьютере.

Синтаксис:

```
getmac [/S <система> [/U <пользователь> [/P <пароль>]]]  
      [/FO <формат>] [/NH] [/V]
```

Параметры:

/S <система>	имя или IP-адрес удаленного компьютера
/U [<домен>\<>пользователь>	Имя пользователя. Если не задано, то используется текущая учетная запись
/P [<пароль>]	Пароль. Если задан параметр /U и не задан пароль, то он будет запрошен
/FO <формат>	Формат, в котором следует отображать результаты запроса. Допустимые форматы: "TABLE" (таблица), "LIST" (список), "CSV" (разделяемые запятыми поля). Если параметр не задан, то используется вывод в виде таблицы (TABLE)
/NH	Указывает, что строка заголовков столбцов не должна отображаться в результирующем файле форматов TABLE и CSV
/V	Отображение подробной информации. В отображаемой информации присутствует

	имя сетевого подключения и название сетевого адаптера
/?	Вывод справки по использованию команды

Примеры:

- GETMAC /? - отобразить краткую справку об использовании
- GETMAC. GETMAC /FO csv - выдать информации о MAC-адресах всех существующих на локальном компьютере сетевых адаптеров в формате CSV (полей с разделителями в виде запятой).
- GETMAC /S COMPUTER /NH /V - получить MAC адреса сетевых адаптеров для удаленного компьютера COMPUTER, не отображать заголовки столбцов в таблице и использовать отображение подробной информации. Для подключения к удаленному компьютеру используется текущая учетная запись пользователя.
- GETMAC /S 192.168.1.1 /NH /V - то же самое, но вместо имени компьютера задан его IP-адрес.
- GETMAC /S COMPUTER /U user /P password - получить MAC - адрес адаптеров удаленного компьютера COMPUTER. Для подключения к нему используется имя пользователя "user" и пароль "password"
- GETMAC /S COMPUTER /U mydomain\user - для подключения к удаленному компьютеру используется учетная запись пользователя "user" в домене "mydomain". Пароль пользователя вводится по запросу.
- GETMAC /S COMPUTER /U mydomain\user /P password - то же самое, что и в предыдущем случае, но пароль задан в командной строке.

10. Nbtstat

Команда NBTSTAT позволяет получить статистику протокола NetBIOS over TCP/IP (NetBT), таблицу имен локальных и удаленных компьютеров и содержимое кэш NetBIOS имен. Применение NBTSTAT позволяет принудительно обновить кэш NetBIOS-имен компьютеров и имена, зарегистрированные с помощью серверов Windows Internet Name Service (WINS).

Синтаксис:

nbtstat[-aRemoteName] [-AIPAddress] [-c] [-n] [-r] [-R] [-RR] [-s] [-S]
[Interval]

Параметры:

-a RemoteName	отображает таблицу имен удаленного компьютера
-NetBIOS	имена соответствуют перечню NetBIOS-приложений, выполняющихся на удаленном компьютере
-A IPAddress	то же самое, что и в предыдущем случае, но вместо имени удаленного компьютера используется его IP-адрес
-c	отображает кэш имен NetBIOS и соответствующих им IP-адресов
-n	отображает таблицу NetBIOS-имен на локальном компьютере. Состояние "Зарегистрирован" означает, что имя зарегистрировано с использованием широковещательного запроса или с помощью сервера WINS
-r	отображает статистику разрешения NetBIOS-имен. На компьютерах под управлением Windows XP и старше, выдается отдельная статистика о разрешении имен с помощью широковещательной рассылки и с помощью сервера имен WINS
-R	очистка кэш NetBIOS-имен и загрузка данных из секции #PRE файла LMHOSTS
-RR	очистка кэш NetBIOS - имен на локальном компьютере и их повторная регистрация с использованием сервера WINS
-s	отображает статистику NetBIOS - сессий между клиентом и сервером и NetBIOS-имена удаленных узлов
-S	отображает статистику сессий между клиентом и сервером и IP-адреса удаленных узлов
Interval	интервал обновления отображаемых данных в секундах. Для прекращения автоматического обновления используется комбинация клавиш CTRL+C
/?	отобразить справку по использованию NBTSTAT

Примеры:

`nbtstat -n` - вывести список зарегистрированных NetBIOS-имен на локальном компьютере.

`nbtstat -a SERVER` - вывести список зарегистрированных NetBIOS-имен на компьютере SERVER.

`nbtstat -A 192.168.1.1` - вывести список зарегистрированных NetBIOS-имен на удаленном компьютере с IP-адресом 192.168.1.1 .

`nbtstat -RR` - выполнить очистку и перерегистрацию NetBIOS-имен на локальном компьютере.

11. Netsh.exe

Утилита сетевой оболочки NETSH (NETwork SHell) - наиболее полное и функциональное стандартное средство управления сетью с использованием командной строки в среде Windows XP и старше. Набор внутренних команд сетевой оболочки пополняется с появлением новых версий операционной системы, что необходимо учитывать при работе в локальной сети с различными ОС. Так, например, команда уровня wlan (`netsh wlan` - управление беспроводной сетью) может использоваться на компьютерах под управлением Windows Vista и старше и отсутствует в Widows XP. Синтаксис используемых команд и параметров также может различаться в разных операционных системах семейства Windows.

При запуске NETSH.EXE без параметров на экран выводится приглашение к вводу внутренних команд оболочки. Набор команд представляет собой многоуровневую структуру, позволяющую выполнять необходимые действия в выбранном контексте. При вводе знака вопроса ? можно получить краткую справку по доступному перечню команд на данном уровне. Ввод команды данного уровня со знаком вопроса вызовет отображение справки по ее использованию. Аналогичную справку можно получить, введя определенную команду и, после перехода на уровень ее выполнения, ввести знак вопроса. При необходимости, можно выполнить нужное действие без использования интерактивного режима, указав в качестве параметров командной строки последовательный набор внутренних команд NETSH и необходимых параметров.

Команды NETSH можно выполнить и на удаленном компьютере с использованием подключения по локальной сети. Netsh также предоставляет возможность выполнения сценариев, представляющих собой группу команд в текстовом файле, выполняемых в режиме очереди на определенном компьютере. В целом, возможности NETSH настолько обширны, что трудно

найти сетевую задачу, которую невозможно было бы решить с использованием данной утилиты.

Синтаксис:

NETSH.EXE [-a AliasFile] [-c Context] [-r RemoteMachine] [-u
[DomainName\]UserName] [-p Password | *] [Command | -f ScriptFile]

Параметры:

-a AliasFile	не завершать работу а перейти к приглашению ввода команд после выполнения AliasFile. AliasFile - имя текстового файла, в котором содержатся одна или несколько команд netsh
-c Context	изменить контекст (уровень) команд netsh
-r RemoteMachine	выполнять команды netsh на удаленном компьютере. В качестве RemoteMachine может использоваться имя или IP-адрес
[-u DomainName\]UserName	имя пользователя для подключения к удаленному компьютеру. Если не задано, то используется текущее имя пользователя
-p Password	пароль для подключения к удаленному компьютеру
Command	команда оболочки netsh, которую необходимо выполнить
-f ScriptFile	аналогично ключу -a, но после выполнения команд файла сценария Scriptfile, работа netsh завершается

Команды утилиты:

?	Отображение списка команд
add	Добавление элемента конфигурации в список элементов

advfirewall	Изменения в контексте 'netsh advfirewall'
branchcache	Изменения в контексте 'netsh branchcache'
bridge	Изменения в контексте 'netsh bridge'
delete	Удаление элемента конфигурации из списка элементов
dhcpclient	Изменения в контексте 'netsh dhcpclient'
dnsclient	Изменения в контексте 'netsh dnsclient'
dump	Отображение сценария конфигурации
exec	Запуск файла сценария
firewall	Изменения в контексте 'netsh firewall'
help	Отображение списка команд
http	Изменения в контексте 'netsh http'
interface	Изменения в контексте 'netsh interface'
ipsec	Изменения в контексте 'netsh ipsec'
lan	Изменения в контексте 'netsh lan'
mbn	Изменения в контексте 'netsh mbn'
namespace	Изменения в контексте 'netsh namespace'
nap	Изменения в контексте 'netsh nap'
netio	Изменения в контексте 'netsh netio'
p2p	Изменения в контексте 'netsh p2p'
ras	Изменения в контексте 'netsh ras'
rpc	Изменения в контексте 'netsh rpc'
set	Обновление параметров конфигурации
show	Отображение информации
trace	Изменения в контексте 'netsh trace'
wcn	Изменения в контексте 'netsh wcn'

wfp	Изменения в контексте 'netsh wfp'
winhttp	Изменения в контексте 'netsh winhttp'
winsock	Изменения в контексте 'netsh winsock'
wlan	Изменения в контексте 'netsh wlan'

Чтобы получить справку по команде, введите эту команду, затем пробел и "?".

Также доступны следующие дочерние контексты: advfirewall branchcache bridge dhcpclient dnsclient firewall http interface ipsec lan mbn namespace nap netio p2p ras rpc trace wcn wfp winhttp winsock wlan

Примеры:

- netsh advfirewall show global последовательно выполняется команда первого уровня advfirewall, в ее контексте, команда следующего уровня show с параметром global.
- Как получить справку в виде текстового файла для выбранного контекста NETSH

Для примера, нужно получить справку в контексте работы с конфигурацией беспроводной сети **wlan** .

Последовательно выполняем команды:

```
netsh
wlan
set file open C:\wlanhelp.txt
?
set file close
```

В данном примере, команда **set file open C:\wlanhelp.txt** устанавливает режим вывода консольных сообщений в файл с именем C:\wlanhelp.txt. После установки данного режима, все, что вводится с клавиатуры и отображается на экране, будет записано в указанный текстовый файл. Таким образом, можно создавать файлы журналов отдельных сессий использования **netsh** . Вместо параметра **open** можно использовать **append** и имя уже

существующего файла журнала. В таком режиме данные будут записываться в конец существующего текстового файла.

- Как сохранить и восстановить сетевую конфигурацию

Команда **dump** создает сценарий, который содержит текущую конфигурацию. Если данные сценария сохранить в текстовый файл, то при необходимости, его можно будет использовать для восстановления измененных параметров с помощью команды загрузки и выполнения скриптов **exes**.

Для сохранения используется команда:

dump Имя файла сценария

Для восстановления настроек из файла сценария используется команда:

exes Имя файла сценария

В некоторых версиях netsh команда dump с указанием имени файла почему-то не работает. Однако, для сохранения конфигурации можно воспользоваться способом, описанным выше - использовать запись в файл командой **set file open C:\mynet.sav**.

```
netsh
set file open C:\mynet.sav
dump
quit
```

Остается только слегка исправить полученный файл сценария C:\mynet.sav - удалить 1ю строчку с командой **dump** и последние - с приглашением netsh и (или) командой **quit**.

Второй способ - использовать netsh с перенаправлением вывода команды dump в файл:

netsh dump > C:\mynet.sav

Для сохранения отдельного контекста конфигурации можно воспользоваться командой **dump** на соответствующем уровне:

netsh interface dump > C:\myinterf.cnf - сохранить настройки сетевых интерфейсов в виде сценария netsh в файле C:\myinterf.cnf.

Для восстановления сетевой конфигурации можно воспользоваться

netsh exec C:\mynet.sav

Обычно, после восстановления сетевых настроек из файла сценария , требуется перезапуск некоторых сетевых служб, а желательнее - выполнить перезагрузку Windows.

- Как выполнить переключение между контекстами netsh

Иногда требуется выполнить некоторые команды на одном уровне, перейти на другой, и снова вернуться на предыдущий. Для выполнения таких переходов используются команды **pushd** и **popd** . Принцип переключения между контекстами основан на обработке очереди в соответствии с правилом "первым вошел - последним вышел" или first-in-last-out (FILO) stack. Команда **pushd** запоминает текущий уровень (контекст) в стеке, а команда **popd** извлекает его из стека.

Например:

netsh> - приглашение первого уровня команды nesh
pushd - введена команда запоминания контекста в стек
netsh> - в приглашении netsh не меняется, контекст прежний.
interface ipv4 - переход на уровень interface и уровень ipv4
netsh interface ipv4> - соответственно, изменилась строка приглашения, отображая текущий контекст выполнения команды netsh
set address local static 192.168.1.9 255.255.255.0 192.168.1.1 1 - команда, меняющая настройки IP протокола.
netsh interface ip> - контекст выполнения команды, отображаемый в приглашении не изменяется.
popd - команда извлечения из стека запомненного контекста.
netsh > - строка приглашения изменилась, отображая текущий контекст выполнения команды netsh. .

Без использования команд `pushd` и `popd` практически невозможно полноценное использование сценариев `netsh`.

- Как найти примеры выполнения сетевых настроек с помощью `netsh`

Кроме сохранения и восстановления настроек использование команды `dump` позволяет получить примеры в виде сценария, соответствующего текущей конфигурации. Например, дамп секции `interface` дает пример выполнения команд `netsh` в контексте настроек сетевых интерфейсов.

Пример сценария:

```
#=====
# Конфигурация интерфейса
#=====
pushd interface
reset all
popd
# Конец конфигурации интерфейса
...
# -----
# Настройка IP-интерфейсов
# -----
pushd interface ip
# Интерфейс настройки IP для "Подключение по локальной сети"
```

```
set address name=" Подключение по локальной сети " source=static
addr=192.168.0.1 mask=255.255.255.0 set dns name="Подключение по
локальной сети" source=static addr=192.168.0.2 mask=255.255.255.0 set wins
name=" Подключение по локальной сети " source=static addr=192.168.0.9
```

Строки сценария, начинающиеся с символа `#`, являются комментариями. Команды `pushd` и `popd` позволяют определить контекст исполнения других команд `netsh`. Команды настроек конфигурации плюс справочная информация самой `netsh` позволяют довольно легко получить командную строку для выполнения отдельных сетевых настроек:

Пример: Сменить IP-адрес в командной строке:

netsh interface ip set address name="Подключение по локальной сети" source=static addr=192.168.0.58 mask=255.255.255.0

name - имя сетевого подключения

source - static - статический IP-адрес. Возможно значение DHCP, если адрес назначается автоматически сервером DHCP. addr - значение IP-адреса

mask - значение маски сети.

Для получения сведений о дополнительных возможностях конфигурирования сетевых интерфейсов можно перейти на соответствующий контекст выполнения netsh, и выполнить интересующую команду с параметром ? . Например:

netsh - старт NETSH

interface - перейти в контекст настройки сетевых интерфейсов interface

ip - перейти в контекст настройки протокола IP

set file open C:\setaddr.txt - записывать сессию в файл. Эта команда используется, если нужна справочная информация в виде текстового файла .

set address ? выдать справку по использованию

set address set file close - закрыть файл справки.

quit - завершить работу с netsh

Для Windows Vista / Windows 7 синтаксис будет немного отличаться, уровню ip будет соответствовать уровень **ipv4** :

netsh - старт NETSH

interface - перейти в контекст настройки сетевых интерфейсов interface

ipv4 - перейти в контекст настройки протокола IP

set file open C:\setaddr.txt - записывать сессию в файл. Эта команда используется, если нужна справочная информация в виде текстового файла .

set address ? выдать справку по использованию set address

set file close

quit - завершить работу с netsh

Пример синтаксиса для смены адреса DNS-сервера в настройках сетевого подключения "Подключение по локальной сети 2" на адрес публичного DNS-сервера Google в среде Windows: 7:

netsh interface ipv4 set dnsservers name="Подключение по локальной сети 2" static 8.8.8.8 primary

Из информации файла справки следует, что возможно использование параметров командной строки netsh без указания ключевых слов:

```
netsh interface ip set address name="Подключение по локальной  
сети" source=static addr=192.168.0.58 mask=255.255.255.0  
gateway=192.168.0.1 gwmetric=1
```

Аналогично, без указания ключевых слов:

```
netsh interface ip set address name="Подключение по локальной  
сети" static 192.168.0.58 255.255.255.0 192.168.0.1 1
```

При изменении одного из параметров настроек необходимо указывать и остальные. Например, только для изменения адреса шлюза по умолчанию недостаточно выполнить команду:

```
netsh interface ip set address name="Подключение по локальной  
сети" gateway=192.168.0.1 gwmetric=1
```

При ее выполнении отсутствующие параметры (IP-адрес и маска) будут сброшены. Для правильной смены шлюза по умолчанию команда должна быть следующей:

```
netsh interface ip set address name="Подключение по локальной  
сети" source=static addr=192.168.0.58 mask=255.255.255.0  
gateway=192.168.0.1 gwmetric=1
```

12. Утилита NET.EXE

Утилита NET.EXE существует во всех версиях Windows и является одной из самых используемых в практической работе с сетевыми ресурсами. Позволяет подключать и отключать сетевые диски, запускать и останавливать системные службы, добавлять и удалять пользователей, управлять совместно используемыми ресурсами, устанавливать системное время, отображать статистические и справочные данные об использовании ресурсов и многое другое. Выполнение команды net без параметров вызывает краткую справку со списком возможных уровней использования,

запуск с параметром help позволяет получить более подробную информацию об использовании net.exe:

Синтаксис:

NET HELP имя_команды

-или-

NET имя_команды /HELP

Параметры:

Справочная система NET.EXE, пожалуй, является одной из лучших в семействе операционных систем Windows. Подробную справку по использованию нужной команды, например use , можно получить несколькими способами:

net use ? - справка о синтаксисе команды

net use /help - подробная справка по использованию команды с описанием используемых ключей.

net help use - аналогично предыдущей форме вызова справки.

net help use | more - отобразить справку в постраничном режиме выдачи на экран. Удобно пользоваться в тех случаях, когда текст не помещается на экране. Нажатие Enter перемещает текст на одну строку, нажатие пробела - на один экран.

net help use > C:\helpuse.txt - создать текстовый файл справки C:\helpuse.txt

Команды:

NET ACCOUNTS NET HELP NET SHARE

NET COMPUTER NET HELPMMSG NET START

NET CONFIG NET LOCALGROUP NET STATISTICS

NET CONFIG SERVER NET NAME NET STOP

NET CONFIG WORKSTATION NET PAUSE NET TIME

NET CONTINUE NET PRINT NET USE

NET FILE NET SEND NET USER

NET GROUP NET SESSION NET VIEW

NET HELP SERVICES - эта команда выводит список служб, которые можно запустить.

NET HELP SYNTAX - эта команда выводит объяснения синтаксических правил, используемых при описании команд в Справке.

NET HELP имя_команды | MORE - просмотр справки по одному экрану за раз.

При описании команды NET используются следующие синтаксические соглашения:

- - Заглавными буквами набраны слова, которые должны быть введены без изменений, строчными буквами набраны имена и параметры, которые могут изменяться, например, имена файлов.
- - Необязательные параметры заключены в квадратные скобки - [].
- - Списки допустимых параметров заключены в фигурные скобки - {}. Необходимо использовать один из элементов такого списка.
- - Символ | (вертикальная черта) используется в качестве разделителя элементов списка. Возможно использование только одного из элементов списка. Например, в соответствии с изложенными соглашениями, необходимо ввести NET COMMAND и один из переключателей - SWITCH1 или SWITCH2. Указанное в квадратных скобках имя [name] является необязательным параметром: NET COMMAND [name] {SWITCH1 | SWITCH2} - Запись [...] означает, что указанный элемент может повторяться. Повторяющиеся элементы должны быть разделены пробелом.
- - Запись [...] означает, что указанный элемент может повторяться, но повторяющиеся элементы должны быть разделены запятой или точкой с запятой, но не пробелом.
- - При вводе в командной строке можно использовать русские названия служб, при этом они должны быть заключены в кавычки и не допускается изменение прописных букв на строчные и наоборот. Например, команда NET START "Обозреватель сети" запускает службу обозревателя сети.

Примеры:

- Работа с системными службами

Данный режим использования NET.EXE, в некоторой степени, является не характерным для основного предназначения утилиты, и начиная с Windows XP, для управления системными службами используется специальная утилита командной строки SC.EXE. Тем не менее, NET.EXE в среде любой версии операционных систем Windows может быть использована для запуска и остановки системных служб (сервисов). Согласно справочной информации, список служб, которыми можно управлять с помощью net.exe можно получить, используя следующую команду:

net help services

Но это не совсем верно, и на самом деле, с помощью net.exe можно запустить или остановить практически любую системную службу, и в том числе, не представленную в списке , отображаемом при выполнении данной команды . Для остановки используется параметр stop, а для запуска - параметр start:

net stop dnscache - остановить службу **dnscache**

net start dnscache - запустить службу **dnscache**

Возможно использование как короткого, так и полного имени ("Dnscache" - короткое, "DNSклиент" - полное имя службы). Имя службы, содержащее символы русского алфавита и пробелы заключается в двойные кавычки.

net stop "DNS-клиент" - остановить службу **DNS-клиент** .

Полное имя службы можно скопировать из "Панель управления" - "Администрирование" - "Службы" - Имя службы - "Свойства" - "Выводимое имя". Для приостановки некоторых системных служб или продолжения работы ранее приостановленной службы используются команды **NET PAUSE** и **NET CONTINUE** :

net pause "Планировщик заданий" - приостановить службу "Планировщик заданий" **net continue schedule** - продолжить работу службы "Планировщик заданий". Имя службы задано в коротком формате.

- Работа с сетевыми дисками

net use - отобразить список сетевых дисков, подключенных на данном компьютере.

Состояние	Локальный	Удаленный	Сеть
Отсоединен	X:	\\SERVER\movies	Microsoft Windows Network OK
OK	Y:	\\SERVER\shares	Microsoft Windows Network

В колонке "Локальный" отображается буква сетевого диска, а в колонке "Удаленный" - имя удаленного сетевого ресурса в формате UNC. UNC - это Общее соглашение об именах (Uniform Naming Convention) или

универсальное соглашение об именовании (universal naming convention), соглашение об именовании файлов и других ресурсов, дающее определение местоположения ресурса. Имя, соответствующее UNC - полное имя ресурса в сети, включающее имя сервера и имя совместно используемого (разделяемого, сетевого) ресурса (принтера, каталога или файла). Синтаксис UNC-пути к каталогу или файлу следующий: **\\Сервер\СетевойКаталог[\ОтносительныйПуть]** Сервер - сетевое имя компьютера, **СетевойКаталог** - это сетевое имя общего каталога на этом компьютере, а необязательный **ОтносительныйПуть** - путь к каталогу или файлу из общего каталога. **СетевойКаталог** не обязательно называется так же, как ассоциированный с ним каталог на сервере, имя даётся в ходе открытия общего доступа к каталогу в файловой системе компьютера. В операционных системах семейства Windows, если в конце имени разделяемого ресурса используется знак \$ то такой ресурс является скрытым и не отображается в проводнике при просмотре сетевого окружения. Это правило относится не только к автоматически создаваемым ресурсам для системного администрирования (C\$, D\$, ADMIN\$ и т.п.), но и для любого пользовательского разделяемого ресурса. Если, например, для сетевого доступа выделена папка под именем "movies", то она будет видна в сетевом окружении, а если - под именем "movies\$" - то нет. Для того чтобы скрыть в сетевом окружении отдельный компьютер используется команда:

NET config server /hidden:yes

Чтобы вернуть отображение компьютера в сетевом окружении.

NET config server /hidden:no

UNC-пути можно использовать и для локальной машины, только в этом случае вместо имени "Сервер" нужно подставлять знак "?" или ".", а путь к файлу указывать вместе с буквой диска. Например так: "\\?\C:\Windows\System32\file.exe".

Для отключения сетевого диска или устройства используется команда net use с ключом **/DELETE**

net use X: /delete - отключить сетевой диск X:

Регистр букв в данном ключе не имеет значения и можно использовать сокращения:

```
net use Y: /del
```

Примеры выполнения команды NET USE для подключения сетевых дисков:

```
net use X: \\server\shares - подключить сетевой диск X: которому соответствует разделяемый сетевой каталог с именем shares на компьютере с именем server
```

```
net use Y:\C$ /USER:Администратор admpass - подключить сетевой диск Y: которому соответствует скрытый ресурс C$ (корневой каталог диска C:) .  
При подключении к удаленному компьютеру используется имя пользователя Администратор и пароль admpass
```

То же самое, но с использованием учетной записи в домене mydomain

```
net use Y:\C$ /USER:mydomain\Администратор admpass  
net use Y:\C$ /USER:Администратор@mydomain admpass
```

Если в командной строке пароль не задан, то он будет запрошен при подключении к сетевому ресурсу. Если ключ /USER не задан, то для авторизации на удаленном компьютере используется текущая учетная запись. `net use Y:\C$ /SAVECRED` - выполнить подключение с запоминанием полномочий (credentials) пользователя. При первом подключении, будет выдан запрос на ввод имени пользователя и пароля , которые будут запомнены и не будут запрашиваться при последующих подключениях. Параметр /savecred не работает в версиях Домашняя и Начальная Windows 7 / Windpws XP.

Для изменения режима запоминания подключенных сетевых дисков используется ключ /PERSISTENT

```
net use /PERSISTENT:NO - не запоминать сетевые подключения.  
net use /PERSISTENT:YES - запоминать сетевые подключения.
```

Необходимо учитывать, что режим, определяемый значением ключа /PERSISTENT, относится к вновь создаваемым подключениям. Если,

например, сетевой диск X: был создан при установленном режиме запоминания (PERSISTENT:YES), а затем вы выполнили смену режима командой net use /PERSISTENT:NO и подключили сетевой диск Y: , то после перезагрузки системы, не будет восстановлено подключение диска Y: , но будет восстановлено подключение диска X.

- Работа с файлами и каталогами

NET SHARE - эта команда позволяет выделить ресурсы системы для сетевого доступа . При запуске без других параметров, выводит информацию обо всех ресурсах данного компьютера, которые могут быть совместно использованы . Для каждого ресурса выводится имя устройства или путь и соответствующий комментарий.

net share - получить список разделяемых в локальной сети ресурсов данного компьютера.

Пример списка:

Общее имя	Ресурс	Заметки

G\$	G:\	Стандартный общий ресурс
E\$	E:\	Стандартный общий ресурс
IPC\$		Удаленный IPC
ADMIN\$	C:\WINDOWS	Удаленный Admin INSTALL
C:\INSTALL		Дистрибутивы и обновления

net share INSTALL - получить информацию о разделяемом ресурсе с именем INSTALL .

Имя общего ресурса	INSTALL
Путь	C:\INSTALL
Заметки	Дистрибутивы и обновления
Макс. число пользователей	Не ограничен
Пользователи	Administrator
Кэширование	Вручную

Для добавления нового разделяемого по сети ресурса используется параметр /ADD

net share TEMP="C:\Documents And Settings\LocalSettings\games" -
добавить новый разделяемый каталог под именем TEMP
net share TEMP="C:\Documents And Settings\LocalSettings\games" /users:5
- добавить новый разделяемый каталог под именем TEMP с максимальным
числом одновременно подключающихся пользователей равным 5 .

Кроме этого, при создании разделяемого ресурса можно указать краткое его
описание (заметку) с помощью параметра /REMARK и режим кэширования
файлов с помощью параметра /CACHE .

**NET SHARE имя_ресурса=диск:путь [/USERS:число | /UNLIMITED]
[/REMARK:"текст"] [/CACHE:Manual | Automatic | No]
[/CACHE:Manual | Documents| Programs | None]**

Для удаления существующего разделяемого ресурса используется параметр
/DELETE:

net share TEMP /DELETE - удалить разделяемый ресурс под именем TEMP

Удаление выполняется только для имени разделяемого ресурса и не
затрагивает каталог локального диска, связанный с данным именем.

Для работы с файлами, открытыми по сети на данном компьютере,
используется команда **NET FILE** . По каждому открытому ресурсу
выводится идентификационный номер, путь файла,
имя пользователя, которым используется файл, и количество блокировок при
совместном использовании. Кроме того, команда NET FILE позволяет
закрыть совместно используемый файл и снять блокировки .

net file - получить список открытых по сети файлов .

net file 4050 /close - принудительно закрыть файл, идентификатор которого
равен 4050

Для получения списка компьютеров рабочей группы или домена с
разделяемыми ресурсами используется команда

net view - отобразить список компьютеров в сетевом окружении.

net view | more - отобразить список компьютеров в постраничном режиме вывода на экран.

net view > C:\computers.txt - отобразить список компьютеров с записью результатов в текстовый файл.

Синтаксис данной команды:

**NET VIEW [\\имя_компьютера [/CACHE] | /DOMAIN[:имя_домена]]
NET VIEW /NETWORK:NW [\\имя_компьютера]**

net view \\server - отобразить список сетевых ресурсов компьютера server

net view /DOMAIN:mydomain - отобразить список компьютеров с разделяемыми ресурсами в домене mydomain. Если имя домена не указано, то выводится список всех доступных компьютеров локальной сети.

net view /NETWORK:NW - отобразить список серверов Novell Netware, доступных в данной локальной сети.

net view /NETWORK:NW \\NWServer - отобразить список сетевых ресурсов сервера Netware с именем NWServer .

- Работа с пользователями и компьютерами

Утилита NET.EXE позволяет отобразить данные об учетных записях пользователей и групп, добавлять новые записи, удалять существующие, отображать параметры безопасности, связанные с авторизацией пользователей и некоторые другие операции по администрированию на локальном компьютере или контроллере домена.

NET ACCOUNTS - эта команда используется для обновления базы данных регистрационных записей и изменения параметров входа в сеть (LOGON) . При использовании этой команды без указания параметров, выводятся текущие значения параметров, определяющих требования к паролям и входу в сеть, - время принудительного завершения сессии, минимальную длину пароля, максимальное и минимальное время действия пароля и его уникальность.

Синтаксис данной команды:

NET ACCOUNTS [/FORCELOGOFF:{минуты | NO}]
[/MINPWLEN:длина] [/MAXPWAGE:{дни | UNLIMITED}]
[/MINPWAGE:дни] [/UNIQUEPW:число] [/DOMAIN]

Пример отображаемой информации по команде NET ACCOUNTS:

Принудительный выход по истечении времени через:	Никогда
Минимальный срок действия пароля (дней):	0
Максимальный срок действия пароля (дней):	42
Минимальная длина пароля:	0
Хранение неповторяющихся паролей:	Нет
Блокировка после ошибок ввода пароля:	Никогда
Длительность блокировки (минут):	30
Сброс счетчика блокировок через (минут):	30
Роль компьютера:	РАБОЧАЯ СТАНЦИЯ

При использовании в локальной сети, каждый компьютер может выполнять как роль сервера (server), предоставляющего свои ресурсы для совместного использования, так и рабочей станции (workstation), использующей разделяемые сетевые ресурсы. Основные настройки сетевых служб сервера и рабочих станций можно отобразить с помощью команд:

net config server - настройки сетевых служб для роли сервера.

net config workstation - настройки сетевых служб для роли рабочей станции.

Настройки служб сервера можно изменить с использованием параметров:

/AUTODISCONNECT:минуты - максимальное время, в течение которого сеанс пользователя может быть не активен, прежде чем соединение будет отключено. Можно использовать значение -1, которое означает, что отключение вообще не производится. Допустимый диапазон значений: от -1 до 65535; по умолчанию используется 15.

/SRVCOMMENT:"текст"

Добавляет текст комментария для сервера, который отображается на экране Windows и при выполнении команды NET VIEW. Максимальная

длина этого текста составляет 48 знаков. Текст должен быть заключен в кавычки.

/HIDDEN:{ YES | NO }

Указывает, должно ли выводиться имя данного сервера в списке серверов. Учтите, что "скрытие" сервера не изменяет параметров доступа к этому серверу. По умолчанию используется значение NO.

net config server /SRVCOMMENT:"Игровой сервер"
/AUTODISCONNECT:5 - автоотключение при неактивности пользователя - 5 минут..

net config server /HIDDEN:YES>/AUTODISCONNECT:-1 - автоотключение при неактивности пользователя не выполняется, сервер не отображается в сетевом окружении.

При выполнении на контроллере домена, утилита net.exe позволяет добавлять новые компьютеры в базу данных Active Directory (AD) или удалять существующие компьютеры из нее.

net computer \\notebook /add - добавить в домен компьютер notebook .

net computer \\notebook /del - удалить из домена компьютер notebook .

Для просмотра списка групп пользователей и изменения их состава, а также добавления новых или удаления существующих групп используются команды NET GROUP и NET LOCALGROUP. Первая из них используется только на контроллерах домена и предназначена для работы с группами пользователей в домене.

net group - отобразить список групп пользователей в текущем домене.

net localgroup - отобразить список групп пользователей данного компьютера. Синтаксис и назначение параметров этих команд практически не отличаются.

NET LOCALGROUP [имя_группы [/COMMENT:"текст"]] [/DOMAIN]
имя_группы {/ADD /COMMENT:"текст"} | /DELETE} [/DOMAIN]
имя_группы имя [...] {/ADD | /DELETE} [/DOMAIN]

имя_группы - имя локальной группы, которую необходимо добавить, изменить или удалить. Если указать только имя группы, то будет выведен

список пользователей или глобальных групп, являющихся членами этой локальной группы.

/COMMENT:"текст" - комментарий для новой или существующей группы. Текст должен быть заключен в кавычки.

/DOMAIN - Команда выполняется на основном контроллере домена в текущем домене. В противном случае операция выполняется на локальном компьютере. **имя [...]** - Список из одного или нескольких имен пользователей, которые необходимо добавить или удалить из локальной группы. Имена разделяются пробелом. Эти имена могут быть именами пользователей или глобальных групп, но не именами других локальных групп. Если пользователь зарегистрирован в другом домене, его имени должно предшествовать имя домена (например, SALES\RALPHR).

/ADD - Добавляет имя группы или имя пользователя в локальную группу. Регистрационная запись для добавляемых пользователей или глобальных групп должна быть создана заранее.

/DELETE - Удаляет имя группы или пользователя из локальной группы.

net localgroup Администраторы - отобразить список пользователей локальной группы Администраторы данного компьютера.

net localgroup Администраторы testuser /add - добавление в группу Администраторы нового пользователя с именем testuser

net localgroup Администраторы testuser /delete - удалить пользователя testuser из группы Администраторы

Для работы с учетными записями пользователей используется команда **net user**

**NET USER [имя_пользователя [пароль | *] [параметры]] [/DOMAIN]
имя_пользователя {пароль | *} /ADD [параметры] [/DOMAIN]
имя_пользователя [/DELETE] [/DOMAIN]**

имя_пользователя - имя пользователя, которое необходимо добавить, удалить, изменить или вывести на экран. Длина имени пользователя не должна превосходить 20 знаков.

пароль - пароль для учетной записи пользователя. Пароль должен отвечать установленным требованиям на длину - быть не короче, чем значение, установленное параметром /MINPWLEN в команде NET ACCOUNTS, и в то же время не длиннее 14 знаков. * - Вызывает открытие специальной строки

ввода пароля. Пароль не выводится на экран во время его ввода в этой строке.

/DOMAIN команда будет выполняться на контроллере домена в текущем домене.

/ADD - добавление нового пользователя.

/DELETE - удаление пользователя.

Параметры - Допустимые параметры:

/ACTIVE:{YES | NO} - Активизирует учетную запись или делает ее не активной. Если учетная запись не активна, пользователь не может получить доступ к серверу. По умолчанию используется значение YES (т.е. учетная запись активна).

/COMMENT:"текст" - Добавляет описательный комментарий об учетной записи (длиной не более 48 знаков). Текст должен быть заключен в кавычки.

/COUNTRYCODE:nnn - Использует кодовую страницу нужного языка для вывода справки и сообщений об ошибках. Значение 0 означает выбор кодовой страницы по умолчанию.

/EXPIRES:{дата | NEVER} - Устанавливает дату истечения срока действия учетной записи. Если используется значение NEVER, то время действия учетной записи не ограничено. Дата истечения срока действия задается в формате дд/мм/гг или мм/дд/гг, в зависимости от того, какая кодовая страница используется. Месяц может быть указан цифрами, названием месяца или трехбуквенным его сокращением. В качестве разделителя полей должен использоваться знак косой черты (/).

/FULLNAME:"имя" - Указывает настоящее имя пользователя (а не кодовое имя, заданное параметром имя_пользователя). Настоящее имя следует заключить в кавычки.

/HOMEDIR:путь Указывает путь к домашнему каталогу пользователя. Этот каталог должен существовать.

/PASSWORDCHG:{YES | NO} Определяет, может ли пользователь изменять свой пароль. По умолчанию используется значение YES (т.е. изменение пароля разрешено).

/PASSWORDREQ:{YES | NO} Определяет, является ли указание пароля обязательным. По умолчанию используется значение YES (т.е. пароль обязателен).

/PROFILEPATH[:путь] Устанавливает путь к профилю пользователя.

/SCRIPTPATH:путь Устанавливает расположение пользовательского сценария для входа в систему.

/TIMES:{промежуток | ALL} - Устанавливает промежуток времени, во время которого пользователю разрешен вход в систему. Этот параметр

задается в следующем формате: день[-день][,день[-день]], время[-время][,время[-время]] Время указывается с точностью до одного часа. Дни являются днями недели и могут указываться как в полном, так и в сокращенном виде. Время можно указывать в 12- и 24-часовом формате. Если используется 12-часовой формат, то можно использовать am, pm, a.m. или p.m. Значение ALL указывает, что пользователь может войти в систему в любое время, а пустое значение указывает, что пользователь не может войти в систему никогда. Разделителем полей указания дней недели и времени является запятая, разделителем при использовании нескольких частей является точка с запятой.

/USERCOMMENT:"текст" - Позволяет администратору добавлять или изменять текст комментария к учетной записи.

/WORKSTATIONS:{имя_компьютера[,...] | *} - Перечисляет до восьми различных компьютеров, с которых пользователь может войти в сеть. Если данный параметр имеет пустой список или указано значение *, пользователь может войти в сеть с любого компьютера.

Примеры использования:

net user - отобразить список пользователей

net user /DOMAIN - отобразить список пользователей текущего домена

net user VASYA /USERCOMMENT:"Тестовый пользователь " /add - добавить пользователя с именем VASYA

net user VASYA /delete - удалить созданного пользователя.

net user VASYA password /USERCOMMENT:"Тестовый пользователь " /add - создать учетную запись нового пользователя VASYA с паролем password . **net user VASYA * /USERCOMMENT:"Тестовый пользователь " /add** - то же, что и в предыдущей команде, но пароль будет запрошен при создании новой учетной записи.

net user VASYA * - изменить пароль существующего пользователя VASYA. Новый пароль будет запрошен при выполнении команды.

net user VASYA Boss - изменить пароль пользователя VASYA на новое значение Boss

Пример последовательности команд для создания нового пользователя с правами локального администратора:BR>

net user VASYA Boss /ADD - создание учетной записи.

net localgroup Администраторы VASYA /ADD - добавление пользователя в группу "Администраторы"

- Отправка сообщений по локальной сети

Для отправки сообщений в локальной сети используется команда **NET SEND**

NET SEND {имя | * | /DOMAIN[:имя] | /USERS} сообщение имя - имя пользователя, компьютера или имя для получения сообщений, на которое отправляется данное сообщение. Если это имя содержит пробелы, то оно должно быть заключено в кавычки (" "). * - отправка сообщения по всем именам, которые доступны в данный момент.

/DOMAIN[:имя домена] - сообщение будет отправлено по всем именам домена данной рабочей станции. Если указано имя домена, то сообщение отправляется по всем именам указанного домена или рабочей группы.

/USERS - сообщение будет отправлено всем пользователям, подключенным в настоящий момент к серверу.

Сообщение - текст отправляемого сообщения.

Для того чтобы получить сообщение, должна быть запущена "Служба сообщений" (MESSENGER). Имена пользователей, компьютеров и текст сообщений на русском языке должны быть в DOS-кодировке. Перечень доступных активных имен на данном компьютере и состояние службы сообщений можно получить с использованием команды `net name` без параметров. По всему списку имен, отображаемому в результате выполнения данной команды возможна отправка сообщений. Примеры использования:

net send VASYA привет! - отправка сообщения на имя VASYA .

net send * привет! - отправка сообщения всем пользователям локальной сети, имена которых можно определить.

net send /DOMAIN:mydomain Привет - отправка сообщения всем пользователям в домене mydomain

net send /USERS Привет! - отправка сообщений всем пользователям, зарегистрированным службой сервера данного компьютера.

- Статистика и синхронизация часов

Утилита NET.EXE позволяет получить статистические данные по использованию служб сервера и рабочей станции. Статистика содержит информацию о сеансах, доступе к сетевым устройствам, объемах принятых и

переданных данных, отказах в доступе и ошибках, обнаруженных в процессе сетевого обмена.

net statistics server - отобразить статистические данные для службы сервера

net statistics workstation - отобразить статистические данные для службы рабочей станции

Для изменения системного времени компьютера используется команда NET TIME:

NET TIME [\\компьютер | /DOMAIN[:домен]] /RTSDOMAIN[:домен] [/SET] [\\компьютер] /QUERYSNTP [\\компьютер] /SETSNTP[:список серверов NTP]

NET TIME синхронизирует показания часов компьютера с другим компьютером или доменом. Если используется без параметров в домене Windows Server, выводит текущую дату и время дня, установленные на компьютере, который назначен сервером времени для данного домена. Эта команда позволяет задать сервер времени NTP для компьютера. \\компьютер - имя компьютера, который нужно проверить или с которым нужно синхронизировать показания часов.

/DOMAIN[:домен] Задаёт домен, с которым нужно синхронизировать показания часов.

/RTSDOMAIN[:домен] - выполняет синхронизацию времени с сервером времени (Reliable Time Server) из указанного домена.

/SET - Синхронизирует показания часов компьютера со временем указанного компьютера или домена.

/QUERYSNTP - Отображает назначенный этому компьютеру сервер NTP
/SETSNTP[:ntp server list] - задать список серверов времени NTP для этого компьютера. Это может быть список IP-адресов или DNS-имен, разделённых пробелами. Если задано несколько серверов, список должен быть заключён в кавычки.

net time \\COMPUTER - отобразить время на компьютере COMPUTER. Вместо имени компьютера можно использовать его IP-адрес.

net time \\COMPUTER /SET - установить часы текущего компьютера по значению часов компьютера COMPUTER

net time \\COMPUTER /SET /YES - установить часы текущего компьютера по значению часов компьютера COMPUTER без запроса подтверждения.

Обычно ключ /YES используется в командных файлах, выполняющихся без участия пользователя.

net time /QUERYSNTP - отобразить сервер времени, определенный для данного компьютера.

net time \\\COMPUTER /QUERYSNTP - отобразить сервер времени, определенный для указанного компьютера.

net time /SETSNTTP:"1.ru.pool.ntp.org time.windows.com" - задать в качестве NTP-серверов узлы **1.ru.pool.ntp.org** и **time.windows.com**

13. Утилита Nslookup.exe

Утилита NSLOOKUP присутствует во всех версиях операционных систем Windows и является классическим средством диагностики сетевых проблем, связанных с разрешением доменных имен в IP-адреса. NSLOOKUP предоставляет пользователю возможность просмотра базы данных DNS-сервера и построения определенных запросов, для поиска нужных ресурсов DNS. Практически, утилита выполняет функции службы DNS-клиент в командной строке

Windows. После запуска, утилита переходит в режим ожидания ввода. Ввод символа ? или команды help позволяет получить подсказку по использованию утилиты.

Примеры:

nslookup	запуск утилиты
yandex.ru	отобразить IP-адрес(а) узла с именем yandex.ru . Точка в конце имени желательна для минимизации числа запросов на разрешение имени к серверу DNS. Если завершающей точки нет, то NSLOOKUP сначала попытается разрешить указанное имя как часть доменного имя компьютера, на котором она запущена
server 8.8.4.4	установить в качестве сервера имен DNS-сервер Google с IP-адресом 8.8.4.4
yandex.ru	повторить запрос с использованием разрешения имени DNS-сервером Yandex
set type=MX	установить тип записи MX
yandex.ru	отобразить MX-запись для домена yandex.ru - В примере узел

	обмена почтой для домена - mx.yandex.ru
mx.yandex.ru	отобразить информацию по mx.yandex.ru
set type=A	установить тип записи в А
mx.yandex.ru	получить IP-адреса для mx.yandex.ru
exit	завершить работу с nslookup

Возможно использование утилиты NSLOOKUP не в интерактивном режиме:

nslookup odnoklassniki.ru - определить IP-адрес узла odnokassniki.ru с использованием сервера DNS, заданного настройками сетевого подключения.

nslookup odnoklassniki.ru 8.8.8.8 - определить IP-адрес узла odnokassniki.ru с использованием DNS-сервера 8.8.8.8 (публичный DNS-сервер Google)

3.Примеры практического использования сетевых утилит командной строки

3.1. Управление профилями беспроводных сетей

Обычно, если вы хотите изменить или удалить профиль беспроводного подключения, это можно сделать, щелкнув правой кнопкой мыши сеть в списке и выбрав команду в появившемся меню. Но для выполнения некоторых задач необходимо использовать командную строку. В следующей таблице показано, как выполнять стандартные задачи.

Задача Инструкции Показать все профили беспроводной связи на компьютере В командной строке введите: netsh wlan show profiles Показать ключ безопасности доступного профиля Нажмите и удерживайте или щелкните правой кнопкой мыши сеть в списке, а затем коснитесь или щелкните Просмотреть свойства подключения. Показать ключ безопасности профиля вне доступа В командной строке введите: netsh wlan show profile name="ProfileName" key=clear Удалить доступный профиль Нажмите и удерживайте или щелкните правой кнопкой мыши сеть в списке, а затем коснитесь или щелкните Забыть эту сеть. Удалить профиль вне доступа В командной строке введите: netsh wlan delete profile name="ProfileName"

Переместить сеть вверх в списке приоритета Чтобы поместить сеть в верхнюю строку списка, достаточно подключиться к ней и установить автоподключение. Отключить автоматическое подключение к доступной сети Коснитесь или щелкните сеть в списке, затем нажмите Отключение. Отключить автоматическое подключение к сети вне доступа В командной строке введите: `netsh wlan set profileparameter name="ProfileName" connectionmode=manual`

3.2. Определение подмены адреса узла в файле hosts

Одним из последствий вирусного заражения довольно часто является блокировка доступа к сайтам антивирусных компаний, поисковым системам, популярным социальным сетям (Vkontakte, Odnoklassniki, Facebook, Twitter и т.п.). Подобный же прием используется для кражи учетных данных пользователей путем перенаправления на вредоносный сайт, адрес которого берется из зараженного файла hosts.

Порядок преобразования доменных имен в IP-адреса следующий:

- проверяется наличие данных об имени в кэш службы разрешения имен (процедура определения IP по имени уже выполнялась, и в памяти есть актуальные результаты). Если запись есть, то будут использованы ее данные. - проверяется наличие записи об имени и адресе в файле hosts. Если запись есть, то будут использованы ее данные.
- для разрешения доменного имени в IP-адрес выполняется запрос к серверу DNS, заданному в настройках сетевого подключения.

Файл hosts при настройках по умолчанию, находится в каталоге `\Windows\system32\drivers\etc\` и обычно содержит строки, начинающиеся с символа # , являющиеся комментариями, и одну запись для определения имени узла петлевого интерфейса:

127.0.0.1 localhost

127.0.0.1 - IP-адрес, localhost - имя. Если добавить запись 127.0.0.1 odnoklassniki.ru, то для имени odnoklassniki.ru будет использоваться адрес 127.0.0.1, который не предназначен для выполнения реальной передачи данных, и сервер с указанным именем станет недоступен. Если же вместо адреса 127.0.0.1 использовать адрес поддельного сервера, созданного злоумышленниками, то вместо реального сайта, соответствующего доменному имени, посетитель перейдет на поддельную страницу.

Структура записей файла `hosts` предполагает, что между адресом и соответствующим ему именем должен быть хотя бы один символ табуляции (пробел). Каждой записи отводится одна строка в файле `hosts`. Иногда, вредоносная программа выполняет смещение записей относительно отображаемой на экране части файла, заполняя видимую часть пробелами, а в непомещающейся в области просмотра части, могут присутствовать записи, например `31.214.145.172 odnoklassniki.ru 31.214.145.172 www.facebook.com 31.214.145.172 www.vk.com 31.214.145.172 www.vkontakte.ru`

Данный адрес взят из реально зараженного файла `hosts` и принадлежит сети одного из провайдеров Германии. Сейчас он безопасен, и не занят обслуживанием вредоносного сервера. На зараженном компьютере, в файл `hosts` было добавлено множество пустых строк, и поддельные записи располагались с разным смещением относительно начала строки, что могло затруднить ручной поиск. Кроме того, вредоносные программы могут использовать и некоторые другие способы подмены содержимого `hosts` - изменение местоположения самого файла, использование атрибута "скрытый" и имени с подменой символа на похожий по написанию символ национального алфавита - "о" и т.п. Другими словами, достоверно определить сам факт подмены адреса с помощью файла `hosts`, путем прямого анализа содержимого реестра, системных каталогов и самого файла занимает довольно длительное время и не всегда позволяет исключить ошибку поиска вредоносных записей. А, тем временем, задача легко решается с использованием всего лишь 2-х команд из рассмотренных выше - `ping` и `nslookup`.

`ping odnoklassniki.ru` - в ответе на пинг будет отображаться адрес, соответствующий имени `odnoklassniki.ru` при определении IP-адреса на данном компьютере

`nslookup odnoklassniki.ru` - получить IP-адрес, соответствующий имени `odnoklassniki.ru` от сервера DNS.

Если адрес по результатам пинга отличается от адреса, полученного от DNS-сервера, то присутствует факт подмены содержимого файла `hosts`. Для некоторых крупных доменов утилита `nslookup` может выдавать список из нескольких IP. Тогда IP-адрес, полученный в результатах пинга, должен присутствовать в списке адресов от `nslookup`.

Иногда, в качестве способа блокировки определенных сайтов, используется добавление несуществующих статических маршрутов для соответствующих IP-адресов или подсетей, что легко отследить с помощью утилиты `tracert`.

3.3. Как открыть порт в брандмауэре Windows

Разрешить входящие соединения через брандмауэр Windows (открыть порт) можно с использованием контекста firewall утилиты netsh

```
netsh firewall set portopening protocol=TCP port=27015 name=MyServer mode=ENABLE scope=ALL
```

или

```
netsh firewall set portopening TCP 27015 MyServer ENABLE ALL protocol
```

Протокол порта. TCP (Transmission Control Protocol), UDP (User Datagram Protocol), ALL - Все протоколы. port - Номер порта. name - Имя порта (необязательно) mode - Режим порта. ENABLE - Пропускать через брандмауэр (по умолчанию). DISABLE - Не пропускать через брандмауэр. scope - Область порта (необязательно). ALL - Пропускать через брандмауэр весь трафик (по умолчанию). SUBNET - Пропускать через брандмауэр только трафик локальной сети (подсети). CUSTOM - Пропускать через брандмауэр только указанный трафик.

С учетом значений по умолчанию и необязательных параметров открыть TCP порт 27015 в брандмауэре Windows можно командой netsh firewall set portopening TCP 27015

В Windows Vista/Windows7 пока поддерживается синтаксис приведенный в примере выше, однако в последующих версиях операционных систем он будет полностью заменен на контекст netsh advfirewall - управление улучшенным брандмауэром. Подсказку по использованию можно получить при вводе команды с параметром ? (знак вопроса):

```
netsh advfirewall ?
```

В контексте правил для брандмауэра:

```
netsh advfirewall firewall ?
```

Для открытия порта 27015 в Windows 7 с учетом нового синтаксиса правильнее использовать команду:

```
netsh advfirewall firewall add rule name="Open Port 27015" dir=in action=allow protocol=TCP localport=27015
```

add rule - добавить правило name - название правила. Название может быть произвольным, и если текст содержит пробелы - заключаться в двойные кавычки. Имя правила не должно принимать значение all dir - направление обмена данными (in-входящий трафик, out- исходящий) action - действие по отношению к попадающему под правило соединению (allow - разрешить, block - запретить) protocol - разновидность протокола. (TCP - протокол TCP, UDP - протокол UDP, ANY - любой протокол). Если параметр protocol не указан, то используется значение по умолчанию - ANY) localport - номер порта на локальном компьютере. Можно указывать диапазон портов 0 -65535 или any - любой порт или номера через запятую - 67,69.

Примеры правил брандмауэра Windows 7

По сравнению с предыдущими версиями Windows синтаксис правил стал немного сложнее, но и возможности брандмауэра значительно расширились.

Краткий список возможных параметров правил:

```
add rule name=<строка> dir=in|out action=allow|block|bypass [program=<путь к
программе>] [service=<краткое имя службы>|any] [description=<строка>]
[enable=yes|no (по умолчанию - yes)] [profile=public|private|domain|any[,...]]
[localip=any||<подсеть>|<диапазон>|<список>]
[remoteip=any|localsubnet|dns|dhcp|wins|defaultgateway|
||<подсеть>|<диапазон>|<список>] [localport=0-65535||<диапазон
портов>[,...]]RPC|RPC-EPMap|IPHTTPS|any (по умолчанию - any)]
[remoteport=0-65535|<диапазон портов>[,...]]any (по умолчанию - any)]
[protocol=0-255|icmpv4|icmpv6|icmpv4:тип,код|icmpv6:тип,код| tcp|udp|any (по
умолчанию - any)] [interfacetype=wireless|lan|ras|any]
[rmtcomputergrp=<строка SDDL>] [rmtusrgrp=<строка SDDL>]
[edge=yes|deferapp|deferuser|no (по умолчанию - no)]
[security=authenticate|authenc|authdynenc|authnoencap|notrequired (по
умолчанию – notrequired)]
```

Некоторые правила применения параметров:

Параметры могут следовать в произвольном порядке - dir=in action=allow и action=allow dir=in являются допустимыми значениями. Если указана удаленная группа пользователей или компьютеров, для параметра security необходимо установить значение authenticate, authenc, authdynenc или authnoencap. Установка authdynenc в качестве значения параметра security позволяет системам динамически согласовывать использование шифрования трафика, соответствующего данному правилу брандмауэра Windows. Шифрование согласуется в соответствии со свойствами существующего правила безопасности соединения. Этот параметр позволяет компьютеру принять первый пакет TCP или UDP входящего соединения IPsec, при условии, что он защищен, но не зашифрован, с помощью IPsec. Как только первый пакет будет обработан, сервер повторно согласует соединение и обновит его, чтобы все последующие соединения были полностью зашифрованы.

Если action=bypass, должна быть указана группа удаленных компьютеров, если dir=in.

Короткое имя службы можно посмотреть в ее свойствах, в поле Имя службы. Так, для службы "DNS-клиент" короткое имя - Dnscache . Если service=any, правило действует только для служб.

Значением кода или типа ICMP может быть any - любой ICMP трафик. Параметр edge можно указывать только для правил входящего трафика (dir=in) .

AuthEnc и authnoencap нельзя использовать вместе. Если задан параметр authnoencap, то параметр security=authenticate задавать необязательно.

Параметр Authdynenc допустим только в том случае, если значение dir равно in.

Примеры:

Добавление правила для входящего трафика для программы qip.exe:

```
netsh advfirewall firewall add rule name="allow QIP" dir=in  
program="c:\programfiles\qip\qip.exe" action=allow
```

Добавление правила, запрещающего исходящий трафик для TCP порта 80:

```
netsh advfirewall firewall add rule name="allow80" protocol=TCP dir=out  
localport=80 action=block
```

Добавление правила входящего трафика с требованием безопасности и шифрования для трафика через TCP-порт 80:

```
netsh advfirewall firewall add rule name="Require Encryption for Inbound  
TCP/80" protocol=TCP dir=in localport=80 security=authdynenc action=allow
```

Добавление правила входящего трафика для messenger.exe с требованием безопасности:

```
netsh advfirewall firewall add rule name="allow messenger" dir=in  
program="c:\program files\messenger\msmsgs.exe" security=authenticate  
action=allow
```

Добавление правила обхода брандмауэра с проверкой подлинности для группы acmedomain\scanners, определяемой строкой SDDL:

```
netsh advfirewall firewall add rule name="allow scanners" dir=in  
rmtcomputergrp=<строка SDDL> action=bypass security=authenticate
```

Добавление правила разрешения исходящего трафика для локальных портов 5000-5010 для udp:

```
netsh advfirewall firewall add rule name="Allow port range" dir=out protocol=udp  
localport=5000-5010 action=allow
```

Для просмотра всех правил брандмауэра используется команда:
netsh advfirewall firewall show rule name=all netsh advfirewall firewall show rule name=all | more - с выдачей результатов на экран в постраничном режиме
netsh advfirewall firewall show rule name=all > C:\firewallrules.txt - с выдачей результатов в файл

Для просмотра конкретного правила указывается его имя. Для удаления правила используется параметр delete:
netsh advfirewall firewall show rule name=TEST просмотр правила с именем TEST netsh advfirewall firewall delete rule name=test - удаление правила с именем TEST

Для изменения значений в существующих правилах используется параметр set и new перед изменяемым значением:
netsh advfirewall firewall set rule name="Allow port range" new localport=50006000 изменить диапазон портов для правила "Allow port range"
Настройками по умолчанию, в режиме повышенной безопасности брандмауэр Windows 7 блокирует все входящие подключения, не соответствующие ни одному правилу и разрешает исходящие.

4. Wi-Fi точка доступа стандартными средствами Windows 7

В операционной системе Windows 7 реализована технология Virtual WiFi, позволяющая легко создавать программную точку доступа (Software Access Point - SoftAP) . В отличие от полноценных беспроводных точек доступа, реализуемая таким образом SoftAP, позволяет создать только один виртуальный адаптер, который будет работать только в режиме точки доступа, и может быть использовано шифрование только по WPA2-PSK/AES. Тем не менее, этого вполне достаточно для создания функциональной беспроводной сети без реально существующей точки доступа. Такая сеть, обозначается как Wireless Hosted Network, или просто Hosted Network (Размещенная Сеть).

Для создания размещенной сети используется команды сетевой оболочки netsh.exe в контексте wlan:
netsh wlan set hostednetwork [mode=]allow|disallow - разрешить или запретить использование размещенной сети. netsh wlan set hostednetwork [ssid=]<идентификатор_SSID> [key=]<парольная_фраза> [keyUsage=]persistent|temporary - задать параметры размещённой сети. ssid - идентификатор SSID сети, другими словами - имя беспроводной сети; key - ключ безопасности, используемый в данной сети, т.е. парольная фраза,

используемая при подключении клиентов к виртуальной точке доступа. Ключ должен быть строкой символов ASCII длиной от 8 до 63 знаков. keyUsage - указывает, является ключ безопасности постоянным или временным. По умолчанию, ключ является постоянным (persistent) и используется при каждом включении размещенной сети.

Примеры:

set hostednetwork mode=allow set hostednetwork ssid=ssid1 set hostednetwork key=passphrase keyUsage=persistent set hostednetwork mode=allow ssid=MyWiFi key=MyPassWord Или - одной командной строкой: netsh wlan set hostednetwork mode=allow ssid=MyWiFi key=MyPassWord - создать виртуальную точку доступа Wi-Fi с именем MyWiFi и паролем MyPassWord

Созданная программная точка доступа не будет запущена автоматически. Для запуска потребуется выполнить команду:

netsh wlan start hostednetwork

Для остановки - netsh wlan stop hostednetwork

При использовании команд управления размещенной сетью требуются права администратора. Для организации доступа в Интернет с использованием размещенной сети можно воспользоваться совместным подключением через, созданный после выполнения команды создания размещенной сети, виртуальный сетевой адаптер - Адаптер мини-порта виртуального WiFi Microsoft (Microsoft Virtual WiFi miniport adapter) . Если же данный адаптер не обнаруживается в диспетчере устройств и отсутствует в списке сетевых адаптеров, то наиболее вероятно, что драйвер реального Wi-Fi устройства не сертифицирован для использования в операционной системе Windows 7 и не поддерживает технологию Virtual WiFi.

4.Задания для выполнения

1. Запустить командную консоль.
2. Попыаться выполнить все приведенные сетевые команды с различными параметрами.
3. Исследовать, как параметры влияют на результат выполнения команд.
4. Написать программу на любом языке программирования для взаимодействия пользователя со следующими утилитами: утилита для настройки TCP/IP config, ping, tracert.

5.Варианты индивидуальных заданий

К написанной программе (п. 3 №4) добавить следующие утилиты (возможности):

1. NETSTAT, редактирование файла hosts.
2. Настройка firewall.
3. Полная реализация утилиты ipconfig, редактирование firewall.
4. Настройка Wi-Fi точки доступа.
5. NBTSTAT, GETMAC.
6. ROUTE, NET.
7. NETSH.
8. ARP, NSLOOKUP

6. Список литературы

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
Владимирский государственный университет имени А.Г. и Н.Г. Столетовых

Кафедра информационных систем
и программной инженерии

**ЛАБОРАТОРНАЯ РАБОТА №2
УСТАНОВКА И НАСТРОЙКА
ОДНОРАНГОВОЙ СЕТИ НА ОСНОВЕ ОС
WINDOWS NT/2000/XP/2003/7.
ИЗУЧЕНИЕ ОСНОВНЫХ СЕТЕВЫХ УТИЛИТ**

Владимир 2015 г.

1. Цель работы

1. Выяснить назначение виртуальных машин. Научиться использовать их для работы с операционными системами семейства Windows.
2. Знать базовые принципы IP-адресации.
3. Научиться настраивать стек протоколов TCP/IP и уметь организовывать одноранговую сеть.
4. Ознакомиться и уметь использовать основные консольные утилиты для получения информации о сетевой конфигурации системы и тестирования работоспособности сети.

2. Общие сведения

2.1 Понятие локальных вычислительных сетей

Локальная вычислительная сеть, ЛВС (англ. Local Area Network, LAN) – компьютерная сеть, покрывающая относительно небольшую территорию, такую как дом, офис, или небольшую группу зданий, например, институт.

Компьютеры могут соединяться по различным протоколам, таким как wi-fi или Ethernet. ЛВС может иметь шлюзы с другими локальными сетями; быть частью или иметь подключение к глобальной вычислительной сети, например, к Интернет.

Классические архитектуры организации локальных сетей

Одноранговые, или децентрализованные сети – это компьютерные сети, основанные на равноправии участников. В таких сетях отсутствуют выделенные серверы, а каждый узел (peer) является как клиентом, так и сервером (см. рис. 1).

К основным достоинствам одноранговых сетей можно отнести:

- низкая стоимость;
- высокая надежность;
- простота реализации, настройки, сопровождения для относительно небольших проектов.

Но данной архитектуре присущи и недостатки:

- зависимость эффективности работы сети от количества станций;
- сложность управления сетью при большом количестве узлов;
- сложность обеспечения защиты информации;
- трудности обновления и изменения программного обеспечения станций.

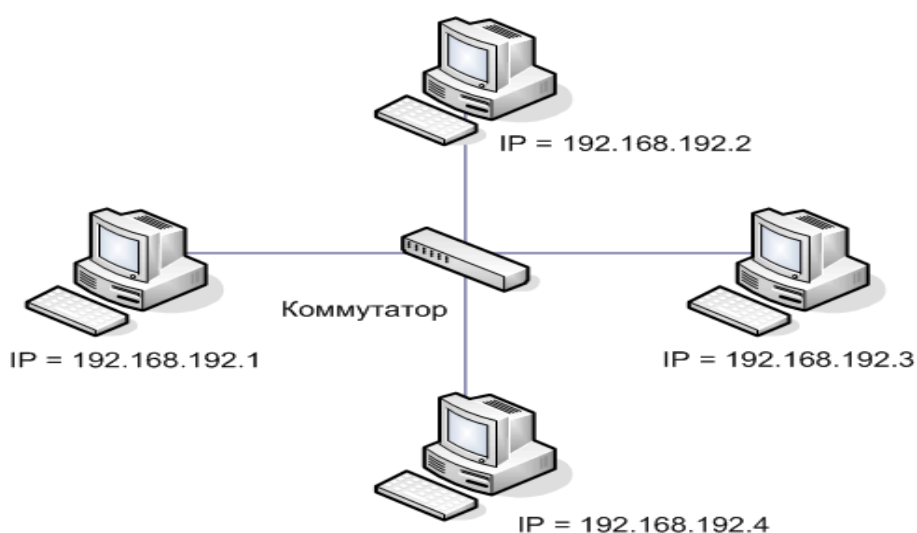


Рис. 1. Пример одноранговой сети

Сеть с выделенным сервером – это локальная вычислительная сеть (LAN), в которой сетевые устройства централизованы и управляются одним или несколькими серверами. Индивидуальные рабочие станции или клиенты (такие, как ПК) должны обращаться к ресурсам сети через сервер(ы) (см. рис. 2).

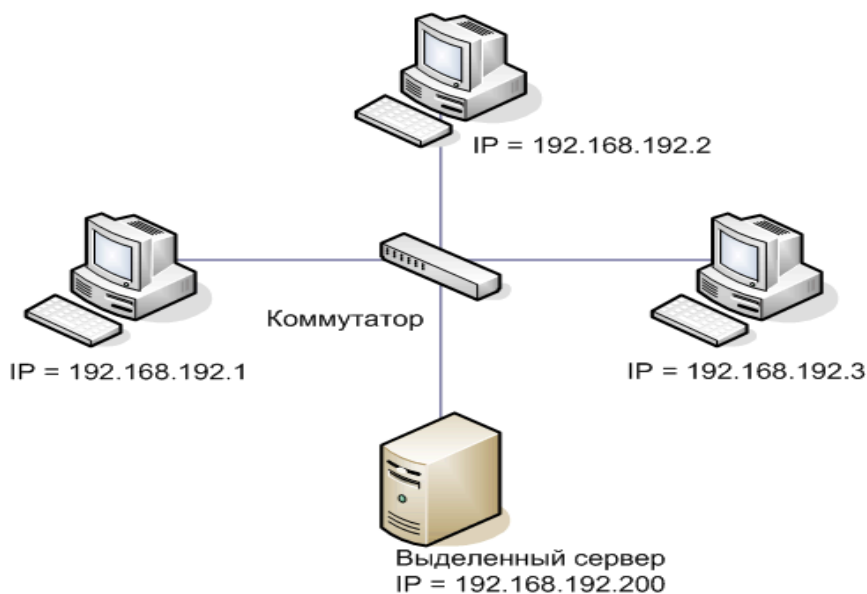


Рис. 2. Пример сеть с выделенным сервером

Достоинства сети с выделенным сервером:

- надежная система защиты информации;
- высокое быстродействие;
- отсутствие ограничений на число рабочих станций;
- простота управления по сравнению с одноранговыми сетями.

Недостатки данной архитектуры:

- более высокая стоимость из-за выделения одного компьютера под сервер;
- зависимость быстродействия и надежности сети от сервера;
- меньшая гибкость по сравнению с одноранговой сетью.

2.2 Виртуальные машины

Виртуальная машина – это вид программного обеспечения, эмулирующего работу аппаратного обеспечения компьютера. Как и в случае с реальной машиной, вы можете установить на виртуальную машину операционную систему, причем неважно Windows или *nix. Таким образом вы можете тестировать различные операционные системы не покидая своей. У виртуальной машины эмулируются BIOS, жесткий диск (на базе – файлов), CD-ROM (физический CD-привод или подключенный ISO-образ), сетевые адаптеры для соединения с вашей реальной машиной, сетевыми ресурсами или другими виртуальными машинами и т.д. Вы можете без проблем обмениваться файлами между основной операционной системой (host) и гостевой операционной системой (guest). Это осуществляется различными способами от общих («расшаренных») каталогов до простого перетаскиванием файлов из файлового менеджера клиента в окно гостевой системы или в обратном направлении.

Удобство виртуальной машины для тестирования автоматической установки просто неопределимо. Достаточно просто подключить загрузочный ISO-образ вместо CD-ROM в настройках виртуальной машины, и установка системы пойдет точно так же, как и на реальной машине.

В ходе изучаемого курса «Сети ЭВМ и телекоммуникации» мы неоднократно будем использовать возможности виртуализации для настройки, тестирования и изучения принципов работы сетевых операционных систем, протоколов и т.д.

Настройка Microsoft Virtual PC (в качестве альтернативного ПО можете использовать Oracle VM VirtualBox)

На данный момент для пользователей доступно на бесплатной основе несколько различных виртуальных машин. Наиболее популярные программы виртуализации: Microsoft Virtual PC, VmWare, Innotek VirtualBox.

В дальнейшем на занятиях предлагается использовать программу Microsoft Virtual PC 2007. Она бесплатна и очень легко настраивается. К тому же для упрощения работы на лабораторных занятиях предлагаются предустановленные виртуальные машины для ряда операционных систем. В частности, Windows 2000 SP2.

Инсталляция программы доступна на кафедральном ftp-сервере.

Основное рабочее окно программы после установки выглядит аналогично рис. 3.

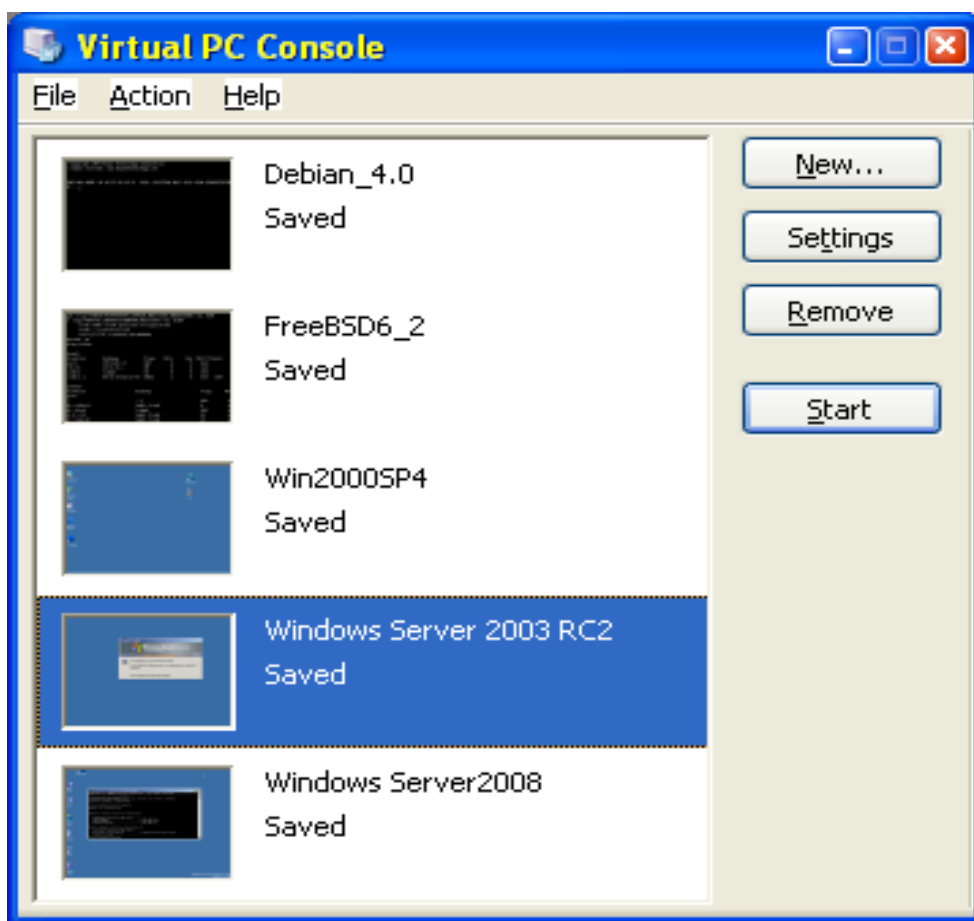


Рис. 3. Менеджер виртуальных машин

Заметьте, что сразу после установки список доступных виртуальных машин будет пуст.

Рассмотрим с вами процесс подключения и конфигурирования виртуальных машин на примере предустановленной ОС Windows 2000sp2 (образы доступны на кафедральном сервере).

Для начала необходимо скачать на локальную машину 2 файла: *.vhd (образ жесткого диска) и *.vmc (конфигурационный файл).

Далее для подключения образа необходимо выбрать пункт меню «File->New Virtual Machine Wizard». После чего в мастере создания виртуальных машин выбрать пункт «Add an existing virtual machine» (см. рис. 4.)

После выбора соответствующего конфигурационного файла (рис. 5), машина появляется в списке доступных для запуска (рис. 6). Вы также можете изменить настройки машины (рис. 7). Например, изменить объем выделяемой оперативной памяти, добавить или удалить сетевой адаптер и т.д.

После установки и настройки виртуальной машины, ее можно запустить посредством кнопки «Start».

Работа в виртуальной среде практически полностью идентичная работе с host-машиной. Имеющиеся отличия заключаются в интерпретации комбинаций служебных клавиш. Информацию об этом можно легко почерпнуть из меню «File», где для каждой служебной команды сопоставлено сочетание клавиш.

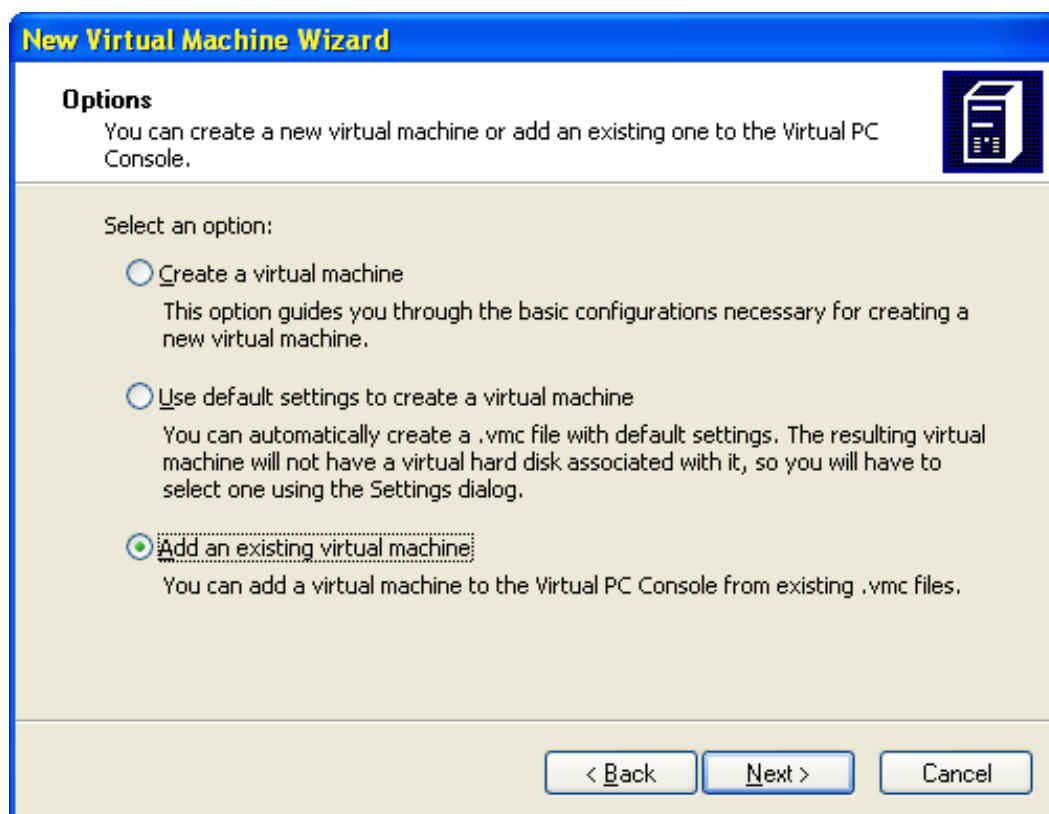


Рис. 4. Подключение виртуальной машины

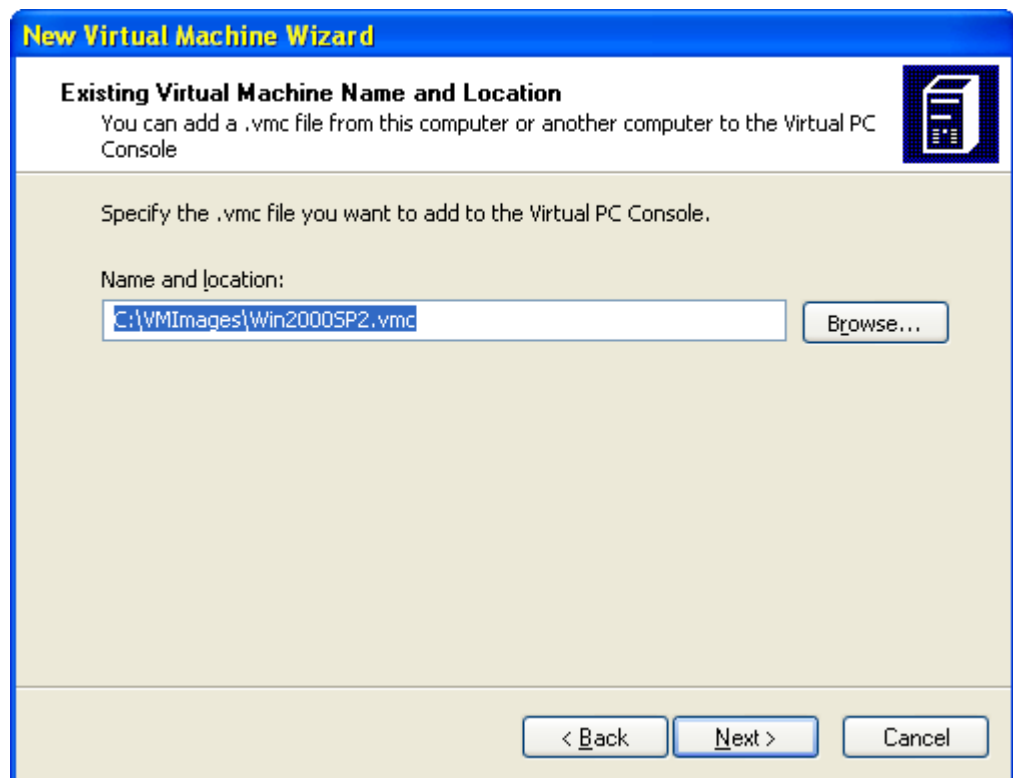


Рис. 5. Выбор конфигурационного файла

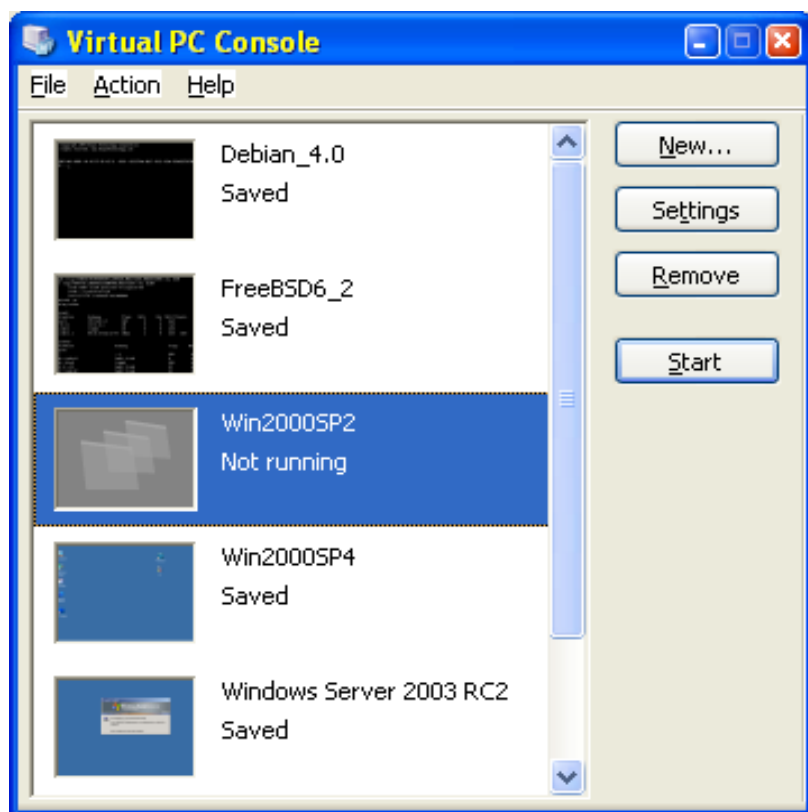


Рис. 6. Обновленный список виртуальных машин

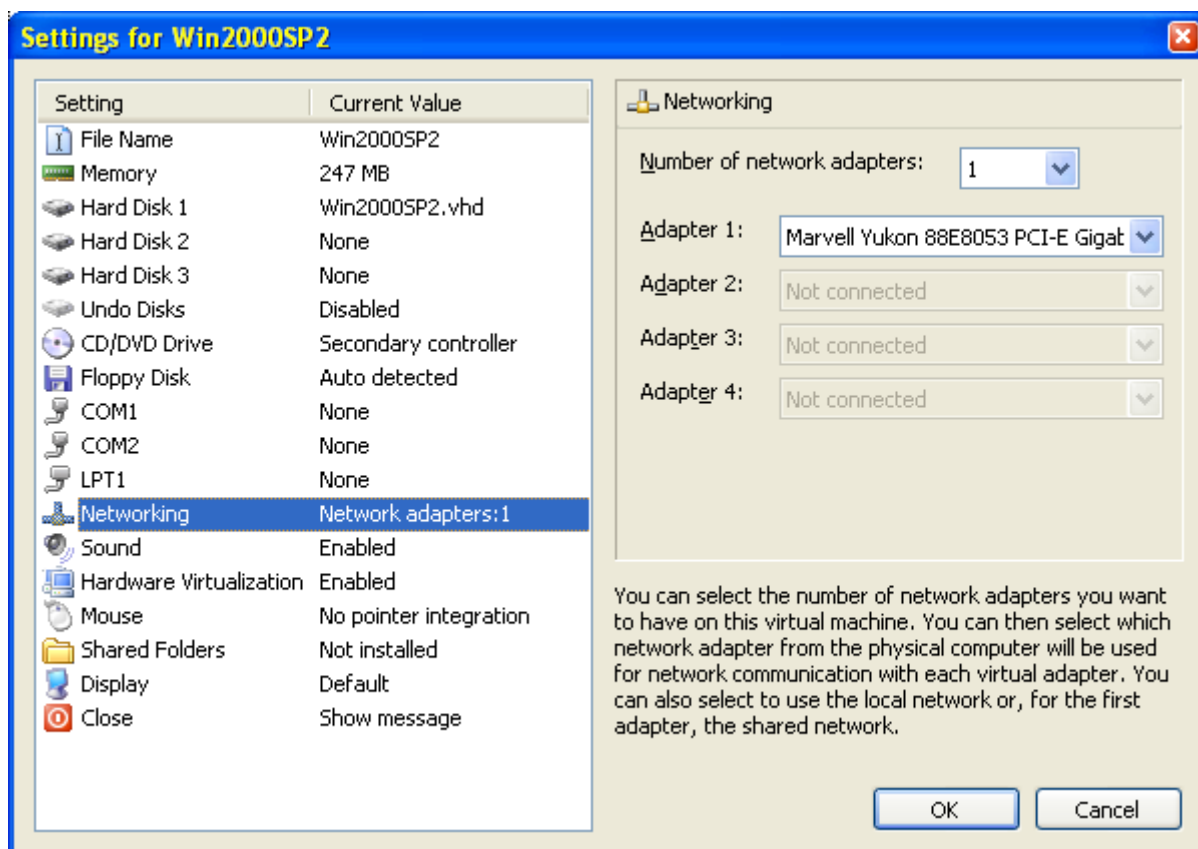


Рис. 7. Настройка виртуальной машины

2.3 Локальные сети по технологии Ethernet

На данный момент самой распространенной аппаратной технологией организации локальных сетей является Ethernet. Сейчас локальные сети стоят преимущественно строят по звездообразной топологии с использованием сетевых карт, коммутаторов и витой пары. Далее мы рассмотрим перечисленные компоненты более подробно.

Сетевая карта (сетевой адаптер, Ethernet-адаптер, NIC (network interface card)) – печатная плата, позволяющая подключаться к компьютерной сети (см. рис. 8).

Обычно, сетевая плата идет как отдельное устройство и вставляется в слоты расширения материнской платы (в основном, PCI). На современных материнских платах, сетевой адаптер все чаще является встроенным, таким образом, покупать отдельную плату не нужно до тех пор, пока не требуется организация ещё одного сетевого интерфейса.

На сетевой плате для подключения к локальной сети имеются разъёмы для подключения кабеля витой пары (например, RJ-45), а также несколько информационных светодиодов, сообщающих о наличии подключения и передаче информации.



Рис. 8. Сетевой Ethernet-адаптер с разъемом RJ-45

Сетевой коммутатор, или свитч (switch - переключатель) – устройство, предназначенное для соединения нескольких узлов компьютерной сети в пределах одного сегмента. В отличие от концентратора (hub), который распространяет трафик от одного подключенного устройства ко всем остальным, коммутатор передает данные только непосредственно получателю. Это повышает производительность и безопасность сети, избавляя остальные сегменты сети от необходимости (и возможности) обрабатывать данные, которые им не предназначались.



Рис. 9. Пример 24-х портового коммутатора

Витая пара (англ. twisted pair) – вид кабеля связи, представляет собой одну или несколько пар изолированных проводников, скрученных между собой (с небольшим числом витков на единицу длины), для уменьшения взаимных наводок при передаче сигнала, и покрытых пластиковой оболочкой

(см. рис 10). Один из компонентов современных структурированных кабельных систем. Используется в телекоммуникациях и в компьютерных сетях в качестве сетевого носителя во многих технологиях, таких как Ethernet, ARCNet и Token ring. В настоящее время, благодаря своей дешевизне и лёгкости в установке, является самым распространённым для построения локальных сетей.

Кабель подключается к сетевым устройствам при помощи соединителя RJ-45 (см. рис. 11), немного бОльшим, чем телефонный соединитель RJ11.

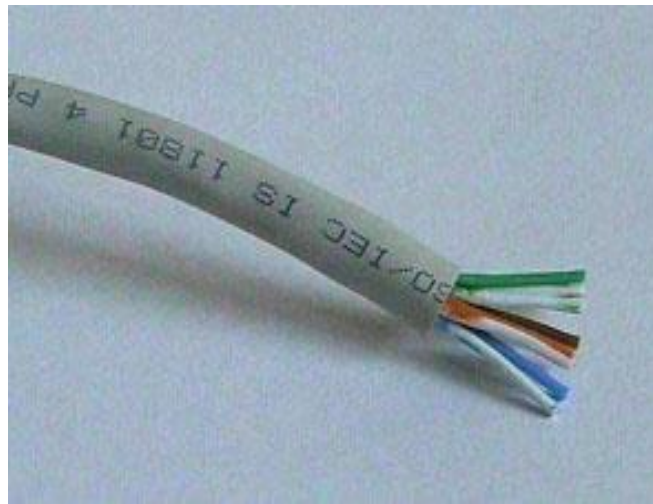


Рис. 10. Неэкранированная витая пара

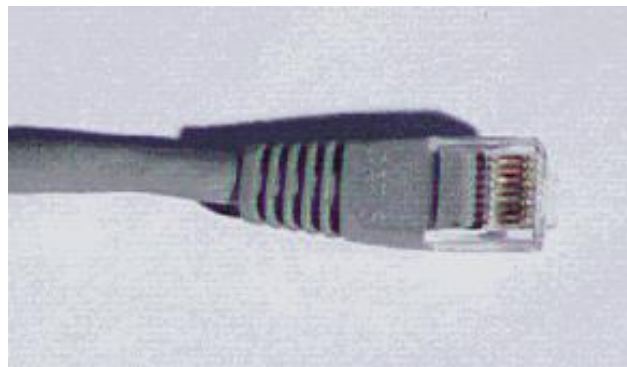


Рис. 11. Витая пара с коннектором RJ-45

2.4 Адресация в локальных сетях

Для взаимодействия узлов на аппаратном уровне каждый активный участник сети (например, сетевой адаптер) должен быть адресуем некоторым образом для любого иного участника сети.

Аппаратные адреса в технологии Ethernet носят названия «MAC-адреса».

MAC-адрес (от англ. Media Access Control – управление доступом к носителю) – это уникальный идентификатор, сопоставляемый с различными типами оборудования для компьютерных сетей. Большинство сетевых протоколов канального уровня используют одно из трёх пространств MAC-адресов, управляемых IEEE: MAC-48, EUI-48 и EUI-64. Адреса в каждом из пространств теоретически должны быть глобально уникальными. Не все протоколы используют MAC-адреса, и не все протоколы, использующие MAC-адреса, нуждаются в подобной уникальности этих адресов.

В ширококвещательных сетях (таких, как сети на основе Ethernet) MAC-адрес позволяет уникально идентифицировать каждый узел сети и доставлять данные только этому узлу. Таким образом, MAC-адреса формируют основу сетей на канальном уровне, которую используют протоколы более высокого (сетевого) уровня. Для преобразования MAC-адресов в адреса сетевого уровня и обратно применяются специальные протоколы (например, ARP и RARP в сетях TCP/IP).

Адреса типа MAC-48 наиболее распространены; они используются в таких технологиях, как Ethernet, Token ring, FDDI и др. Они состоят из 48 бит.

Визуально MAC-адреса обычно записываются в виде 6 шестнадцатеричных чисел. Например, **00-06-29-C9-57-1C**.

Примечание. Для корректной работы устройств в рамках одной локальной сети MAC-адреса устройств должны быть уникальны. В виду того, что в учебных аудиториях предлагается использовать виртуальные машины с одного образа, после запуска машин аппаратные адреса виртуальных машин будет совпадать. Для решения этой проблемы необходимо завершить работу запущенной машины, открыть в текстовом редакторе (хорошо подходит notepad, т.к. поддерживает кодировку unicode) конфигурационный файл виртуальной машины *.vmx и изменить строку, задающую аппаратный адрес,

например поменяв последнее число на номер компьютера в аудитории (см. рис. 12).

```
<ethernet_adapter>
  <controller_count type="integer">1</controller_count>
  <ethernet_controller id="0">
    <virtual_network>
      <id type="bytes">76544E39925111DA821390A33B23363A</id>
      <name type="string">IBM 10/100 EtherJet PCI Management Adapter</name>
    </virtual_network>
    <ethernet_card_address type="bytes">0003FFC85703</ethernet_card_address>
  </ethernet_controller>
</ethernet_adapter>
```

Рис. 12. Задание аппаратного адреса сетевого адаптера виртуальной машины

2.5 Стек протоколов TCP/IP. IP-адресация

На программном уровне взаимодействие между участниками сети осуществляется через так называемые стеки протоколов. Самым распространенным стеком протоколов является TCP/IP, который используется как в локальных сетях, так и сети Internet.

Адресация машин, использующих стек TCP/IP осуществляется на базе IP-адресов.

IP-адрес (ай-пи адрес, Internet Protocol Address) — уникальный идентификатор (адрес) устройства, подключённого к локальной сети или интернету.

IP-адрес представляет собой 32-битовое (по версии IPv4) или 128-битовое (по версии IPv6) двоичное число. Удобной формой записи IP-адреса (IPv4) является запись в виде четырёх десятичных чисел (от 0 до 255), разделённых точками, например, 192.168.0.1. (или 128.10.2.30 — традиционная десятичная форма представления адреса, а 10000000 00001010 00000010 00011110 — двоичная форма представления этого же адреса).

IP-адреса представляют собой основной тип адресов, на основании которых сетевой уровень протокола IP передаёт пакеты между сетями. IP-адрес назначается либо статически (администратором), либо динамически (управляется некоторым выделенным сервером; чаще всего по протоколу DHCP) во время конфигурирования сетевых настроек.

IP-адрес состоит из двух частей: номера сети и номера узла. В случае изолированной сети её адрес может быть выбран администратором из специально зарезервированных для таких сетей блоков адресов (192.168.0.0/16, 172.16.0.0/12 или 10.0.0.0/8).

Номер сети может быть выбран администратором произвольно, либо назначен по рекомендации специального подразделения Internet (Internet Network Information Center, InterNIC), если сеть должна работать как составная часть Internet. Обычно поставщики услуг Internet получают диапазоны адресов у подразделений InterNIC, а затем распределяют их между своими абонентами.

Рассмотрим более подробно то, как интерпретируются IP-адреса.

IP-адреса: классы, подсети и узлы

Некоторое время назад Интернет было разделено на три основных класса сетей: класс А, класс В и класс С (классовая адресация). Сети класса А имели то свойство, что первый октет (восемь битов) IP-адреса определял собственно сеть, а оставшиеся биты использовались организацией, которая управляла сетью, для того, чтобы различать узлы сети. Большинство организаций, управляющих сетями класса А, разделяли их на подсети, добавляя к схеме адресации еще один уровень иерархии. В сетях класса В два первых октета использовались для определения сети, а оставшиеся два - для определения отдельных узлов, а в сетях класса С для определения сети отводилось три октета и лишь один для определения узлов (см. рис. 13).



Рис. 13. Классы IP-адресов

К сожалению, эта система мелких, средних и крупных сетей не всегда была удобна. Многие организации были достаточно велики, чтобы выйти за пределы сети класса C, которая могла содержать максимум 254 узла, но недостаточно велики, чтобы занять целый класс B, сеть которого могла вместить 65534 узла. Но многие из этих организаций получили все-таки сети класса B в свое распоряжение. Как следствие, свободные сети класса B были занесены в красную книгу.

Для решения этой проблемы и создания сетей, которые имели бы соответствующий требованиям размер, была разработана бесклассовая междоменная маршрутизация (Classless Inter-Domain Routing, или CIDR (произносится как «сайдр»). Как видно из названия, CIDR избавляется от классов A, B и C. В системе CIDR для идентификации сети может использоваться не фиксированное число октетов (один, два или три), но любое число битов IP адреса. Так, к примеру, если организации нужно адресное пространство примерно в четыре раза большее, чем адресное пространство сети класса B, власти предрежащие могут определить длину идентификатора сети в 14 битов, таким образом, оставляя 18 битов (в четыре раза больше узлов, чем в сети класса B) на используемое адресное пространство.

Совершенно естественно, что пришествие CIDR сделало «классовую» терминологию устаревшей, хотя она до сих пор довольно часто используется в разговорах. Итак, чтобы обозначить конкретную CIDR-сеть, следует указать конкретное значение старших битов, присваиваемое организации в записи через точку, а также число битов, определяющих сеть. Две части записи разделяются символом «слеш». 15/8 – прежняя сеть класса A, которая «начинается» с восьмибитной последовательности 00001111. Прежняя сеть класса B 128.32.0.0 теперь идентифицируется как 128.32/16. А сеть 192.168.0.128/25 состоит из 128 IP-адресов, начиная с адреса 192.168.0.128 и заканчивая адресом 192.168.0.255.

Маска подсети может задаваться аналогично записи IP-адреса. Например, запись 192.168.128.1/24 и эквивалентна маске подсети 255.255.255.0.

Формально выделение сети и номера узла на основе ip-адреса и маски сети осуществляется как:

NET = IP & MASK,

HOST = IP & (~MASK), где все компоненты 32-х битные числа, «&» – операция логического умножения, а «~» – инверсия бит.

Особые ip-адреса. К особым адресам относят:

- 127.0.0.0/8 – для компьютера обозначает «самого себя»
- Адреса, в которых номер узла обозначен числом, в битовой записи которого установлены все биты. Например, 192.168.192.255/24
- Некоторые другие.

2.6 Настройка сетевого стека TCP/IP для ОС семейства Windows 2000+

Настройка сетевых параметров в ОС Windows 2000 и более поздних осуществляется через свойства соответствующего сетевого адаптера (см. рис 14).

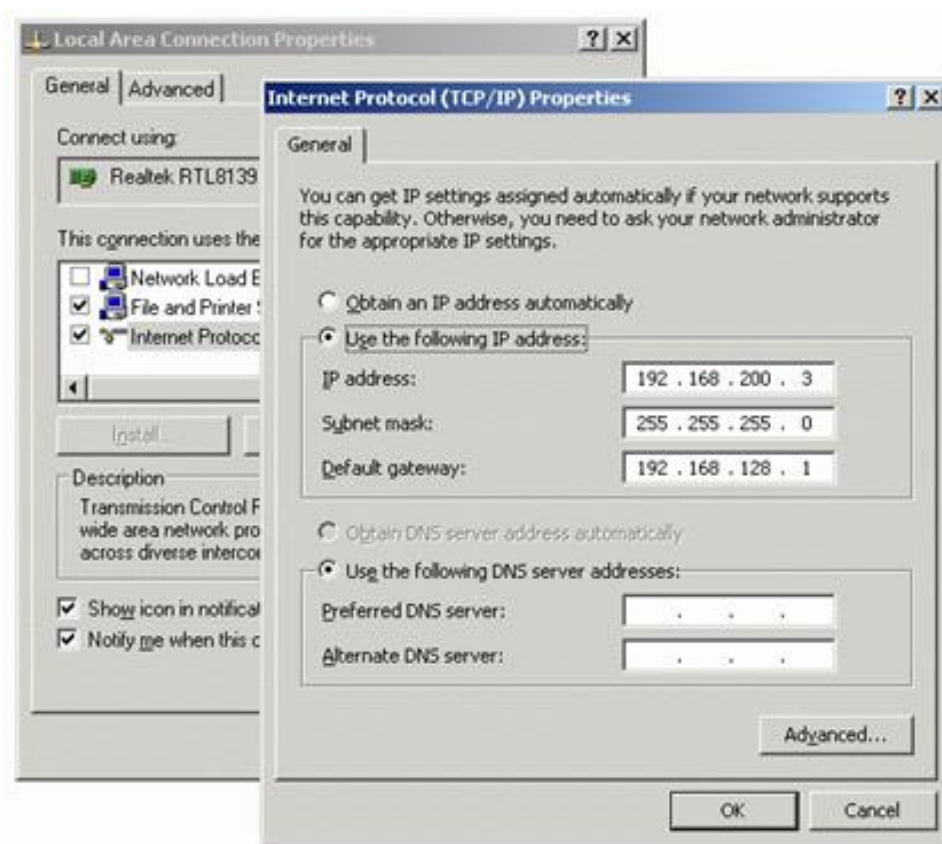


Рис. 14. Окно настройки основных параметров

В примере показано задание статических параметров. Для получения настроек динамически необходимо выбрать опции («Obtain an IP address automatically»)

Примечание. Также конфигурирование может быть осуществлено с использованием консольной утилиты netsh (например, см. «netsh interface /?»).

2.7 Сетевые консольные утилиты

Для работы с сетевыми настройками в консольном режиме (cmd.exe) служат несколько утилит.

Ipconfig – эта утилита инструментов для просмотра настроек сетевых соединений и устранения неисправностей TCP/IP. Она выводит информацию о каждом сетевом адаптере, в том числе о назначенном ему IP-адресе, маске подсети, адресе шлюза, MAC-адресе, адресе DNS-сервера и т. д. Чтобы получить основные сведения о сетевых устройствах, наберите в командной строке ipconfig.

Табл. 1. Параметры утилиты ipconfig

Ключи	Функции
/all	выводит полную информацию о настройках TCP/IP
/displaydns	выводит информацию из кэша DNS
/flushdns	очищает кэш DNS
/registerdns	обновляет все DHCP-адреса и регистрирует заново доменные имена
/release "<адаптер>"	освободить IP-адрес для указанного устройства
/renew"<адаптер>"	возобновляет адрес указанного устройства
/setclassid "<адаптер>"<новый код>	устанавливает код класса DHCP указанного адаптера
/showclassid "<адаптер>"	выводит код класса DHCP указанного адаптера

/?	отображение справки в командной строке

Пример использования:

➤ ipconfig /all

Windows IP Configuration

```
Host Name . . . . . stuff-ctam
Primary Dns Suffix . . . . . : ctam.tu-bryansk.ru
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : ctam.tu-bryansk.ru
```

Ethernet adapter Local Area Connection 1:

```
Connection-specific DNS Suffix . : ctam.tu-bryansk.ru
Description . . . . . : IBM 10/100 EtherJet PCI Management Adapter
Physical Address. . . . . : 00-06-29-C9-57-1C
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
IP Address. . . . . : 192.168.128.111
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.128.1
DHCP Server . . . . . : 192.168.128.1
DNS Servers . . . . . : 192.168.128.5
                        192.168.128.1
Primary WINS Server . . . . . : 192.168.128.1
Secondary WINS Server . . . . . : 192.168.128.1
```

Ping — эта команда, которая базируется на IP- и ICMP-протоколах пересылки дейтограмм и служит для проверки работоспособности каналов и узлов. Для

решения поставленной задачи PING использует отклики протокола ICMP (смотри также статьи о протоколах IP и ICMP).

Применяется PING и при отладке сетевых каналов. Ниже приведены параметры утилиты и пример использования команды Ping.

Табл. 2. Параметры утилиты ping

Ключи	Функции
-t	Отправка пакетов на указанный узел до команды прерывания
-a	Определение адресов по именам узлов
-n	Число отправляемых запросов
-l	Размер буфера отправки
-f	Установка флага, запрещающего фрагментацию пакета
-i TTL	Задание времени жизни пакета (поле "Time To Live")
-v TOS	Задание типа службы (поле "Type Of Service")
-r	Запись маршрута для указанного числа переходов
-s	Штамп времени для указанного числа переходов
-j список узлов	Свободный выбор маршрута по списку узлов
-k список узлов	Жесткий выбор маршрута по списку узлов
-w интервал	Интервал ожидания каждого ответа в миллисекундах

Пример использования:

➤ ping iipo.tu-bryansk.ru

Pinging iipo.tu-bryansk.ru [82.179.88.34] with 32 bytes of data:

Reply from 82.179.88.34: bytes=32 time=6ms TTL=58

Reply from 82.179.88.34: bytes=32 time=5ms TTL=58

Reply from 82.179.88.34: bytes=32 time=5ms TTL=58

Reply from 82.179.88.34: bytes=32 time=7ms TTL=58

Ping statistics for 82.179.88.34:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Approximate round trip times in milli-seconds:

Minimum = 5ms, Maximum = 7ms, Average = 5ms

Arp - эта утилита предназначена для просмотра таблицы соответствия IP-адресов сетевых машин аппаратным (MAC) адресам их сетевых адаптеров. Она также позволяет изменять эту таблицу.

Табл. 3. Параметры утилиты arp

Ключи	Функции
-a	Отображает локальную таблицу соответствия IP-адресов MAC-адресам. Если указан IP-адрес, то выводится информация из таблицы только для соответствующего компьютера. Если в системе установлено более одного сетевого адаптера, то выводится информация из таблицы ARP для всех сетевых адаптеров
-g	То же, что и -a
ip_адрес	IP-адрес
-N адрес_интер фейса	Указывает, что выводятся данные из таблицы ARP только указанного адаптера

-d	Удаляет указанный хост из таблицы ARP. При задании IP-адреса допустимо использование символа * для удаления нескольких адресов. Если адрес интерфейса не указан, то соответствующие записи будут удалены из таблиц всех интерфейсов
-s	Добавляет в таблицу ARP статическую запись. Если не указан адрес интерфейса, то запись будет добавлена в таблицы всех интерфейсов. Статические записи сохраняются только на время работы компьютера - после перезагрузки статические записи требуют повторного добавления
mac_адрес	MAC-адрес. Указывается в виде 6 шестнадцатеричных чисел, разделенных дефисами

Пример использования:

➤ arp -a

Interface: 192.168.50.111 --- 0x10003

Internet Address	Physical Address	Type
192.168.50.1	00-02-b3-50-8d-e7	dynamic
192.168.50.6	00-02-b3-e9-aa-c5	dynamic
192.168.50.11	00-14-85-31-f4-9f	dynamic
192.168.50.18	00-06-29-89-8a-66	dynamic
192.168.50.54	00-c0-f0-56-ae-cf	dynamic
192.168.50.68	00-06-29-89-73-5d	dynamic

3. Задания для выполнения

1. Необходимо установить ОС Windows (XP/2003/7 или Linux:, Ubuntu, Fedora, Knoppix, Debian, Alt linux...) на виртуальную машину Virtual PC или Oracle VM VirtualBox.
2. Настроить стек протоколов TCP/IP:
 - а. Установить взаимодействие между несколькими запущенными в локальной сети виртуальными машинами с использованием статически заданных IP адресов. При этом рекомендуется

использовать диапазон локальных адресов 192.168.0.XX и соответствующую маску подсети 255.255.255.0.

б. Ознакомиться с принципами динамического назначения адресов:

- при наличии DHCP сервера (адреса локальной сети лаборатории, например, подсеть 192.168.192.0, маска 255.255.255.0).
- в отсутствии DHCP сервера (автоматически назначаемые адреса с 169.254.0.1 по 169.254.255.254, маска 255.255.0.0). В данном случае DHCP-сервер должен быть недоступен.

3. Далее попытаться получить доступ к «основной сети» в лаборатории(ях). При этом реализовав как динамическое, так и статическое назначение адресов машинам.

4. Протестировать рассмотренные сетевые утилиты

Дополнительное задание:

1. Попробовать установить для нескольких виртуальных машин одинаковый MAC адрес (в конфигурационном файле образа для Virtual PC – *.vmc).
2. Задать для двух машин одинаковый IP адрес (использовать статическое назначение адресов).

Проанализировать полученные результаты.

Примечание: приветствуется выполнение работы группами при условии работы с несколькими виртуальными машинами.

4. Варианты индивидуальных заданий

Смотрите п. 3.

5. Контрольные вопросы

1. Каково назначение виртуальных машин?
2. Что такое топология «звезда»?
3. Что такое коммутатор (switch)? Чем отличается от классического концентратора (hub)?

4. Что такое и где применяется MAC-адрес? Почему в рамках одной локальной сети MAC-адреса должны быть уникальны?
5. Что такое IP-адрес? Что такое маска подсети?

6. Список рекомендуемой литературы

1. В.Г. Олифер, Н.А. Олифер, “Компьютерные сети”.
2. (ftp://iipo.tu-bryansk.ru/pub/Drozdov/Net/Books/book_comp_net_olifer.chm)
3. Компьютерные сети. 4-е изд. / Э. Таненбаум. – СПб.: Питер, 2006. – 992 с.: ил. – (Серия “Классика computer science”).
4. (<ftp://iipo.tu-bryansk.ru/pub/Drozdov/Net/Books/TanenbaumKomputernyeSeti.djvu>)

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
Владимирский государственный университет имени А.Г. и Н.Г. Столетовых

Кафедра информационных систем
и программной инженерии

**ЛАБОРАТОРНАЯ РАБОТА №3
ВЗАИМОДЕЙСТВИЕ ПРИКЛАДНЫХ
ПРОГРАММ С ПОМОЩЬЮ ТРАНСПОРТНЫХ
ПРОТОКОЛОВ СЕТИ ИНТЕРНЕТ**

Владимир 2015 г

1. Цель работы

Изучение принципов работы транспортных протоколов Интернет, разработка прикладных программ, осуществляющих взаимодействие между собой на основе этих протоколов.

Для выполнения лабораторной работы требуется написать программу, которая выполняется под управлением ОС типа Windows или UNIX и использует для взаимодействия с другими программами заданный протокол сети Интернет. Для разработки программы рекомендуется использовать среду Delphi версии 3.0 или старше под управлением ОС типа Windows 95/98 или Windows NT/2000.

2. Общие сведения

2.1 Транспортный протокол TCP сети Internet

TCP (Transmission Control Protocol) - это один из самых широко распространенных протоколов транспортного уровня. Главная функция TCP заключается в доставке сообщений без потерь, чего не может гарантировать протокол более низкого уровня IP (Internet Protocol). Для доставки сообщений предварительно устанавливается соединение между процессом-отправителем и процессом-получателем. Данное соединение осуществляет надежную доставку дейтаграмм. Протокол TCP производит повторную передачу искаженного или утерянного пакета.

Выделение всех функций, необходимых для надежной доставки сообщений, в отдельный уровень освобождает разработчиков прикладных программ и утилит от решения задач управления потоком дейтаграмм. Протокол обеспечивает сквозную передачу данных от отправителя к получателю. Поскольку TCP ориентирован на установление соединения, то адресат, получивший дейтаграмму, должен уведомить отправителя об этом. Подразумевается, что между отправителем и получателем устанавливается виртуальный канал, где они обмениваются сообщениями, часть из которых есть подтверждения о получении данных либо коды ошибок. Виртуальный канал на самом деле может подразумевать несколько реальных физических каналов передачи данных, поскольку сообщение может проходить через один или несколько шлюзов.

Когда некоторое приложение (процесс) прикладного уровня отправляет сообщение другому приложению с помощью ТСП, предполагается, что сообщение является потоком, т.е. представляет собой поток байтов, передаваемых асинхронно. ТСП получает поток байтов и собирает его в пакеты (сегменты), добавляя заголовки в начало сегментов. Длина сегмента обычно определяется протоколом или выбирается администратором системы.

Процесс обмена данными начинается с передачи запроса на установление соединения от машины-отправителя к машине-получателю. В запросе содержится специальное целое число, называемое номером сокета (socket). В ответ получатель посылает номер своего сокета. Номера сокетов отправителя и получателя однозначно определяют соединение (конечно, соединение также невозможно без указания IP-адресов отправителя и получателя, но эта задача решается протоколами более низкого уровня - IP).

После установления соединения ТСП начинает передавать сегменты сообщения. На более низком IP-уровне отправителя сегменты разбиваются на одну или несколько дейтаграмм. Пройдя через сеть, дейтаграммы поступают к получателю, где IP-уровень снова собирает из них сегменты и передает их ТСП. ТСП собирает все сегменты в сообщение. От ТСП сообщение поступает к процессу-получателю, где обрабатывается протоколом прикладного уровня.

ТСП на машине-получателе собирает целое сообщение из сегментов, руководствуясь порядковыми номерами сегментов, которые записаны в их заголовке. Если какой-то сегмент сообщения потерян или поврежден (что проверяется по контрольной сумме в заголовке сегмента), то отправителю посылается сообщение, содержащее номер ошибочного сегмента. В этом случае отправитель повторно передает сегмент. Если сегмент успешно принят, то получатель посылает отправителю подтверждение-квитанцию (ACK - acknowledgement).

В ТСП применяется средство ограничения потока данных, называемое скользящим окном. Оно представляет собой фрагмент сообщения, которые адресат готов принять. При установлении соединения отправителю сообщается размер окна (размер окна кратен размеру сегмента). После того, как отправитель передал количество байтов, соответствующее размеру окна, он должен ждать квитанции. Как только будет получена квитанция на переданные сегменты, окно сдвигается вправо на соответствующее число байтов, и новые сегменты могут быть переданы. Отправитель может

передать без получения квитанций в сеть максимально столько сегментов, сколько их укладывается в скользящем окне. В процессе обмена данными получатель может присылать квитанции, в которых будет указан новый размер скользящего окна.

Важную роль в протоколе ТСР играют таймеры. Сегмент считается потерянным, если квитанция на него не поступила в течение заданного времени ожидания. При этом производится повторная передача сегмента. При получении квитанции таймер останавливается. Если получатель обнаруживает несколько правильных копий одного и того же сегмента, то все лишние копии просто отбрасываются и отправителю передается только одна квитанция.

2.2 Номера портов и сокет

Приложение (процесс), использующее ТСР однозначно определяется числом - номером порта (сокета). В принципе, номера портов можно назначать процессам произвольно, но для облегчения взаимодействия между различными программами прикладного уровня приняты соглашения о номерах портов, закрепленных за определенными службами Internet. Номера портов наиболее известных служб сети приведены в табл. 4.

Таблица 4 Номера портов наиболее употребительных служб сети Internet

Номер порта	Служба сети	Описание
0		Зарезервирован
7	Echo	Эхо-ответ на входящие сообщения
9	Discard	Сброс (поглощение) всех входящих сообщений
11	Users	Активные пользователи
13	Daytime	Отклик, содержащий время дня
19	Chargen	Генератор символов
20	ftp data	Передача данных по протоколу FTP
21	ftp	Передача управляющих команд по протоколу FTP

23	telnet	Порт подключения по протоколу TELNET
25	Smtп	Протокол передачи почтовых сообщений SMTP
37	Time	Отклик, содержащий время
42	Name	Сервер имен
43	Whois	Кто это
53	Domain	Сервер имен доменов
67	Boots	Протокол удаленной загрузки сервера
68	Bootc	Протокол удаленной загрузки клиента
69	Tftp	Упрощенный протокол передачи файлов TFTP
79	Finger	Протокол получения информации о пользователях FINGER
80	http	Протокол передачи гипертекста HTTP
109	Pop2	Протокол почтового ящика POP2
110	Pop3	Протокол почтового ящика POP3
111	Rpc	Протокол удаленного вызова процедур RPC
156	Sqlserv	Служба SQL
161	snmp	Управляющий протокол SNMP

В формате сообщения протокола TCP под номер порта отводится 16 бит, поэтому максимально возможным номером порта является число 65535. Номера портов от 0 до 255 строго зарезервированы под системные нужды, их не допускается использовать в прикладных программах. В интервале от 256 до 1023 многие порты также используются сетевыми службами, поэтому и их не рекомендуется применять для прикладных нужд. Как правило, большинство прикладных приложений, построенных на основе TCP/IP используют номера портов в диапазоне от 1024 до 5000. Рекомендуется использовать номера от 3000 до 5000, номера выше 5000 используются чаще всего для краткосрочного применения.

Любой канал связи в ТСП определяется двумя числами - эта комбинация называется сокетом. Таким образом, сокет определяется IP-адресом ЭВМ и номером порта, используемым программным обеспечением ТСП. При соединении любая машина однозначно определена IP-адресом, а каждый процесс - портом, поэтому соединение между двумя процессами однозначно определяется сокетом. Схема установления соединения по протоколу ТСП между тремя ЭВМ в сети отражена на рис. 6.

Взаимодействующие ЭВМ ведут таблицы всех активных портов отправителей и получателей. Если между двумя машинами происходит обмен данными, то порт, который в одной из них является отправителем, в другой будет получателем и наоборот. Если машина отправитель запрашивает несколько соединений, то у каждого из них свой порт-отправитель, а порт получателя может быть общим. Несколько машин могут одновременно использовать один и тот же порт получатель, это называется мультиплексированием. На рис. 6 мультиплексирование двух соединений выполняется на ЭВМ 3 по порту номер 23.

Когда устанавливается несколько соединений, то может случиться, что несколько машин пошлют запросы на соединение, в которых указаны одинаковые порты источники и получатели. Однако путаницы с соединениями не возникает, потому что IP-адреса у всех машин разные, следовательно, каждое соединение будет однозначно определено своим сокетом.

2.3 Программирование обмена данными на основе транспортных протоколов

ТСП должен взаимодействовать не только с протоколами нижележащего уровня, но и с протоколами и приложениями прикладного уровня. Связь с прикладным уровнем осуществляется с помощью набора сервисных примитивов. Сервисные примитивы определены в стандарте протокола, а для прикладных программ они доступны в форме библиотек работы с сокетами.

При установлении соединения каждая из сторон выполняет некоторые операции, называемые открытием соединения. Открытие может быть пассивным или активным. Как правило, одна из сторон производит активное открытие соединения, а другая - пассивное, тогда соединение

устанавливается. Оба режима подчиняются четким правилам. Пассивное соединение еще иногда называют серверным, а активное - клиентским.

При активном соединении процесс прикладного уровня передает программному обеспечению ТСП на той же ЭВМ сервисный примитив запроса на установление соединения с номером сокета, после чего ТСП отправляет получателю запрос на установление соединения, затем ждет ответа. После установления соединения активный процесс (клиент) может инициировать прием или передачу данных.

При пассивном соединении прикладная программа переводит программное обеспечение ТСП в режим ожидания запроса на соединение от удаленной системы. Когда поступает запрос, программное обеспечение ТСП осуществляет установку соединения, после чего пассивный процесс (сервер) готов принимать и передавать данные.

Программный интерфейс сокетов был разработан для ОС UNIX. Библиотека функций, поддерживающих этот интерфейс, входит в ядро всех ОС типа UNIX и Linux. Однако принципы работы с этим программным интерфейсом применимы к большинству ОС, поддерживающих TCP/IP (например, в семействе ОС и оболочек типа Windows программный интерфейс сокетов реализован в динамической библиотеке Winsock.dll).

Для протокола TCP пассивное (на стороне сервера) соединение с сокетом приводит к выполнению следующих функций:

- создание сокета и установление его типа (в ОС типа UNIX функция `socket`);
- настройка сокета на конкретное соединение (указывает адрес и номер порта - в ОС типа UNIX - функция `bind`);
- создание очередь клиентов (в ОС типа UNIX - функция `listen`);
- ожидание приходящего запроса на соединение с сокетом (в ОС типа UNIX - функция `accept`);
- прием и передача данных от клиента (в ОС типа UNIX - функции `read`, `write`, `send`, `recv` и их модификации);
- закрытие соединения с клиентом (в ОС типа UNIX - функция `close`).

Получив входящий запрос на соединение, сервер должен решать как бы две задачи одновременно: обслуживать уже установленное с клиентом соединение в соответствии с прикладным протоколом (принимать и отдавать данные клиенту) и ожидать поступления новых запросов на соединение от

других клиентов. Обычно в развитых ОС (а сюда подпадают все современные ОС за исключением, разве что, MS DOS) эта проблема решается за счет возможностей параллельного выполнения нескольких процессов. Сервер может породить новый процесс (или новую цепочку выполнения - thread), который и должен будет заняться обслуживанием уже установленного соединения, а сам основной процесс сервера может закрыть текущее соединение и вновь вернуться к ожиданию запросов на соединение от других клиентов. В ОС типа UNIX создание нового процесса решается с помощью функции `fork`, при этом за вновь созданным процессом сохраняются все соединения, сделанные в основном процессе.

Для протокола TCP активное (на стороне клиента) соединение с сокетом приводит к выполнению следующих функций:

- создание сокета и установление его типа (в ОС типа UNIX функция `socket`);
- установление соединения с сервером (указывает адрес и номер порта - в ОС типа UNIX - функция `connect`);
- прием и передача данных (в ОС типа UNIX - функции `read`, `write`, `send`, `recv` и их модификации);
- закрытие соединения с сервером (в ОС типа UNIX - функция `close`).

Клиент, как правило, не требует для своей работы параллельного выполнения нескольких процессов. О синхронных и асинхронных сокетах можно почитать в Интернете.

В среде программирования Borland Delphi существуют специальные классы, которые позволяют выполнять все те же действия, что и библиотека сокетов в ОС UNIX. Они взаимодействуют с библиотекой `Winsock.dll` на основе специальных технологий ОС (ActiveX технологии и COM-объекты). В среде Borland Delphi версии 3.0 для целей клиентского и серверного соединений служит класс объектов `TTCP`; а в среде Borland Delphi версии 5.0 для клиентского соединения существует класс объектов `TClientSocket`, а для серверного - `TServerSocket`. Естественно, пользователь может на основе базовых классов разрабатывать свои собственные классы, которые будут поддерживать соединения по определенным им самим прикладным протоколам.

Для того чтобы создать сокет, достаточно создать экземпляр объекта выбранного класса (`TTCP` - в среде Borland Delphi версии 3.0 на страничке компонент "Internet", `TClientSocket` или `TServerSocket` - в среде Borland Delphi

версии 5.0 также на страничке компонент "Internet"). Это можно выполнить при проектировании приложения в среде разработки или же средствами языка программирования при выполнении приложения. Чтобы специфицировать (настроить) сокет необходимо созданному экземпляру объекта присвоить нужные значения в указанные свойства (properties) - как правило, это свойства с именами вида "Port" и "Host" (имена и состав свойств зависят от версии среды разработки). Это тоже можно сделать как в режиме проектирования приложения, так и командами присвоения свойств объекта в тексте программы. После этого сокет инициализирован и с ним можно работать.

Для работы сокета клиента необходимо открыть сокет (процедура Open), затем использовать процедуры установления соединения, передачи и приема данных, а в конце работы закрыть сокет (процедура Close). При удалении экземпляра объекта автоматически прекратит существование и связанный с ним сокет.

Для работы сокета сервера после его создания надо вызвать процедуру ожидания соединения. При этом по установлению соединения наступит событие, которое программист должен соответствующим образом обработать. В среде Borland Delphi версии 3.0 программист сам должен создавать потоки выполнения для обслуживания соединения (для выполнения потоков служат экземпляры класса объектов TThread), а в среде Borland Delphi версии 5.0 это можно сделать в автоматическом или полуавтоматическом режиме. После окончательного завершения работы сокет сервера следует закрыть. При удалении экземпляра объекта автоматически прекратит существование и связанный с ним сокет.

Более подробная информация о функциях программного интерфейса с описанными компонентами фирмы Borland и достаточная для выполнения лабораторной работы написана в двух прилагаемых статьях (или можете найти сами в специальной литературе и Internet).

1- Программирование сокетов

С помощью следующего приложения мы осветим вопросы программирования сокетов как для протокола TCP, так и для протокола UDP. Приложение функционирует следующим образом.

1. Клиент считывает со стандартного устройства ввода (клавиатуры) строку символов и посылает эту строку серверу через свой сокет.
2. Сервер принимает строку через свой сокет.

3. Сервер переводит все символы строки в верхний регистр.
4. Сервер отправляет модифицированную строку клиенту.
5. Клиент получает строку и печатает ее с помощью стандартного устройства вывода (монитора).

Мы начнем с рассмотрения приложения, в котором взаимодействие клиента и сервера осуществляется через логическое соединение по протоколу TCP (рис. 2.20).

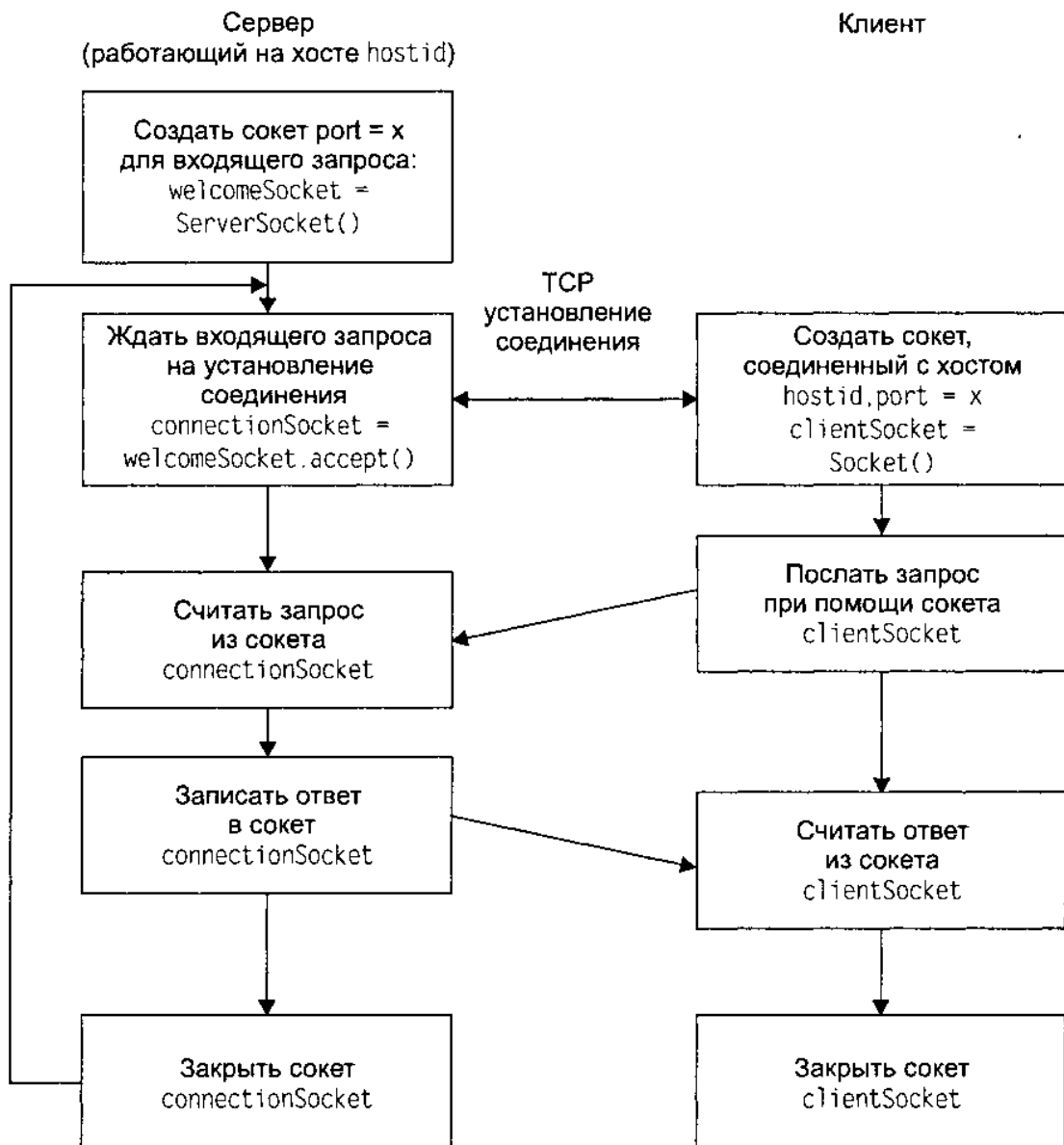


Рис. 2.20. Приложение клиент/сервер, использующее транспортные службы с установлением логического соединения

Ниже мы приведем тексты программ для клиентской и серверной сторон приложения и снабдим пояснениями каждую строку кода. Программа-клиент названа TCPClient.java, программа-сервер — TCPServer.java. Отметим, что приведенные программы лишь иллюстрируют ключевые фрагменты приложения и не претендуют на то, чтобы называться «хорошими» с точки

зрения практического применения. Для улучшения программ приведенные тексты следовало бы снабдить несколькими дополнительными строками.

После компиляции обеих программ (каждой на своем компьютере) первой запускается программа-сервер, поскольку для установления соединения серверный процесс должен ожидать запроса со стороны клиентского процесса. Клиентский процесс создается после запуска программы-клиента, и в его функцию входит инициирование TCP-соединения с сервером. После того как соединение установлено, пользователь, находящийся на клиентской стороне, может вводить строки символов и получать их обратно в верхнем регистре.

Ниже приведен текст программы TCPClient.java.

```
1. import java.io.*;
2. import java.net.*;
3. class TCPClient {
4.     public static void main(String argv[]) throws Exception
5.     {
6.         String sentence;
7.         String modifiedSentence;
8.         BufferedReader inFromUser =
9.             new BufferedReader(
10.                new InputStreamReader(System.in));
11.         Socket clientSocket = new Socket ("hostname". 6789);
12.         DataOutputStream outToServer =
13.             new DataOutputStream(
14.                clientSocket.getOutputStream());
15.         BufferedReader inFromServer =
16.             new BufferedReader(
17.                new InputStreamReader(
18.                    clientSocket.getInputStream()));
19.         sentence = inFromUser.readLine();
20.         outToServer.writeBytes(sentence + '\n');
21.         modifiedSentence = inFromServer.readLine();
22.         System.out.println "FROM SERVER: " +
23.             modifiedSentence);
24.         clientSocket.close();
25.     }
26. }
```

Как показано на рис. 2.21, программа TCPClient создает три потока данных и один сокет. Сокет имеет название clientSocket. Поток inFromUser является входным

поток данных программы и связан со стандартным устройством ввода (клавиатурой). Все символы, вводимые с клавиатуры пользователем, попадают в поток `inFromUser`. Другой входной поток данных программы, `inFromServer`, связан с сокетом и состоит из символов, передаваемых серверной стороной приложения. Выходной поток данных программы `TCPClient` имеет название `outToServer` и также связан с сокетом. Этот поток содержит символы, передающиеся серверу для обработки.

Теперь рассмотрим приведенный код подробнее. Первые две строки содержат имена двух Java-пакетов, `java.io` и `java.net`:

```
1. import java.io.*;
2. import java.net.*;
```

Пакет `java.io` содержит классы входных и выходных потоков данных. Обычно этими классами являются `BufferedReader` и `DataOutputStream`, которые и были использованы в программе. Пакет `java.net` хранит классы, поддерживающие работу с компьютерной сетью (`Socket` и `ServerSocket`). Объект `clientSocket` программы порожден классом `Socket`.

```
1. class TCPClient {
2. public static void main(String argv[]) throws Exception
3. {.....}
4. }
```

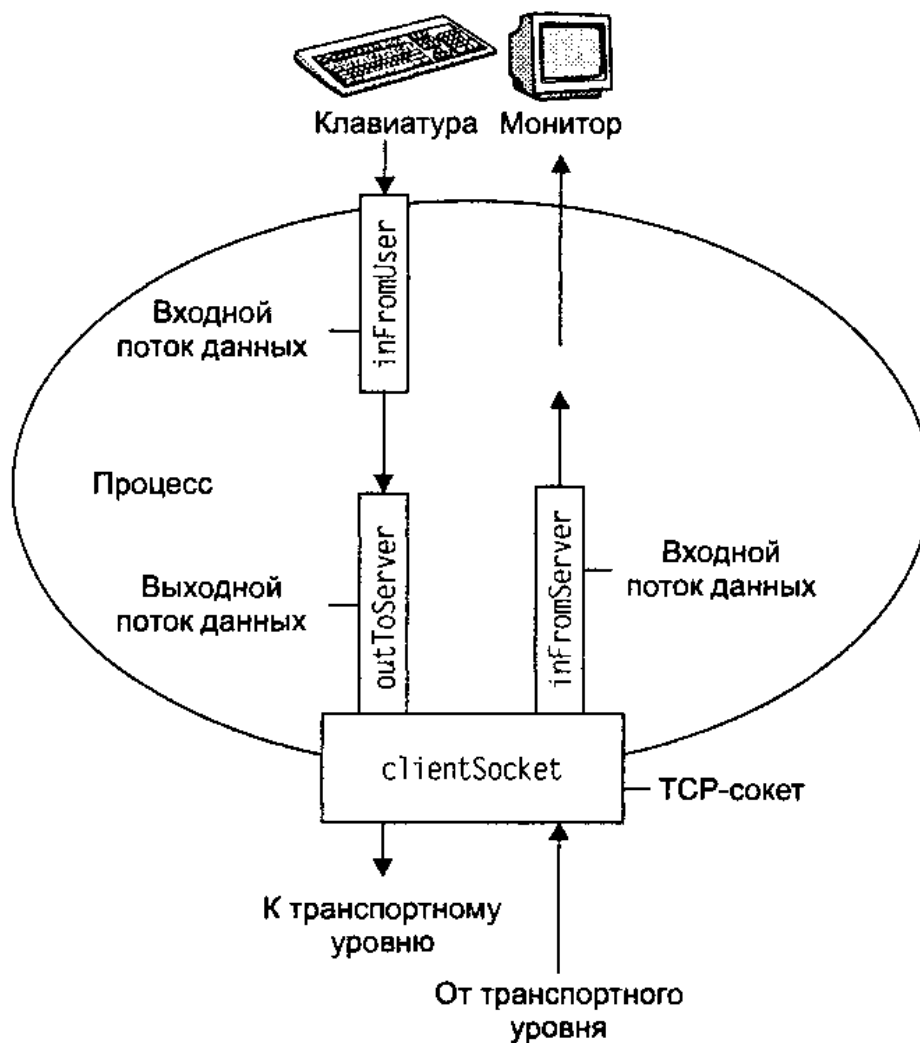


Рис. 2.21. Программа TCPClient организует три потока данных для символов

Конструкции, аналогичные приведенной выше, являются стандартными в практике программирования на Java. Первая строка представляет собой начало блока определения класса. В ней присутствует ключевое слово `class` и имя определяемого класса `TCPClient`. Класс может содержать переменные и методы; в объявлении класса они заключены в фигурные скобки и фактически составляют блок определения класса. В классе `TCPClient` нет переменных, он содержит единственный метод с именем `main()`. Методы похожи на процедуры и функции языков, подобных C; метод `main()` также играет роль, схожую с функцией `main()` в C или C++. Когда интерпретатор Java исполняет приложение (будучи вызванным управляющим классом приложения), он обращается к методу `main()` этого класса. Метод `mainQ` вызывает все остальные методы, используемые в приложении. Если в данный момент вы впервые сталкиваетесь с Java-приложением, можете не обращать внимания на ключевые слова `public`, `static`, `void`, `main` и `throws Exception`.

```
1. String sentence;
2. String modifiedSentence;
```

Приведенные две строки являются объявлениями объектов типа String. Объект `sentence` предназначен для хранения строки, вводимой пользователем и передаваемой серверу. Объект `modifiedSentence` содержит строку, принимаемую от сервера и выводимую на стандартное устройство вывода.

```
1. BufferedReader inFromUser =  
2. new BufferedReader(new InputStreamReader(System.in));
```

Здесь происходит создание потокового объекта `in From User` типа `BufferedReader`. Инициализация потока данных производится объектом `System.in`, связывающим поток со стандартным устройством ввода. После выполнения этой команды клиенту начинают передаваться символы, вводимые пользователем с клавиатуры.

```
1. Socket clientSocket = new Socket ("hostname", 6789);
```

В приведенной строке происходит создание объекта `clientSocket` типа `Socket`. Кроме того, при этом иницируется TCP-соединение между клиентом и сервером. Слово `hostname` необходимо заменить именем хоста, сохранив кавычки (например, `"_fling.seas.upenn.edu"`). Перед началом установки TCP-соединения клиент производит DNS-запрос IP-адреса хоста. Значение 6789 — это номер порта; вы можете выбрать другое число, однако необходимо помнить, что клиентская и серверная стороны должны использовать один и тот же номер порта. Как упоминалось ранее, IP-адрес хоста в совокупности с номером порта приложения идентифицирует серверный процесс.

```
1. DataOutputStream outToServer =  
2. new DataOutputStream(clientSocket.getOutputStream());  
3. BufferedReader inFromServer =  
4. new BufferedReader(new  
5. InputStreamReader(clientSocket.getInputStream()));
```

Здесь происходит создание потоковых объектов, связанных с сокетом. Поток `outToServer` обеспечивает вывод программы через сокет, а поток `inFromServer` предназначен для приема данных от серверной стороны (см. рис. 2.21).

```
1. sentence = inFromUser. readLine();
```


Эта строка помещает последовательность символов, вводимых пользователем, в переменную `sentence`. Ввод оканчивается нажатием пользователем клавиши `Enter`. Как видим, для помещения символов в переменную используется потоковый объект `inFromUser`.

```
1. outToServer.writeBytes(sentence + '\n');
```

Здесь строка `sentence`, снабженная символом возврата каретки, помещается в выходной поток `outToServer`. После этого она будет передана через сокет в канал, соединяющий клиента с сервером.

```
modifiedSentence = inFromServer.readLine();
```

Ответ сервера принимается во входной поток `inFromServer`, откуда принятая строка копируется в переменную `modifiedSentence`. Помещение символов в строку `modifiedSentence` продолжается до тех пор, пока не будет получен символ возврата каретки.

```
1. System.out.println("FROM SERVER: " + modifiedSentence);
```

Здесь происходит вывод принятой от сервера строки на монитор,

```
1. clientSocket.close();
```

Эта строка закрывает сокет, а следовательно, и TCP-соединение между клиентом и сервером. Как будет показано в главе 3, исполнение этой команды ведет к отправке TCP-клиентом сообщения TCP-серверу.

Ниже приведен текст программы `TCPServer.java`.

```
1. import java.io.*;
2. import java.net.*;
3. class TCPServer {
4.     public static void main(String argv[]) throws Exception
5.     {
6.         String clientSentence;
7.         String capitalizedSentence;
8.         ServerSocket welcomeSocket = new ServerSocket (6789);
9.         while (true) {
10.            Socket connectionSocket = welcomeSocket.
11.            accept();
12.            BufferedReader inFromClient =
13.            new BufferedReader(new InputStreamReaderC
14.            connectionSocket.getInputStream()));
15.            DataOutputStream outToClient =
16.            New DataOutputStreamC
```

```

17.    ConnectionSocket.getOutputStream());
18.    clientSentence = inFromClient.readLine();
19.    apitalizedSentence =
20.    clientSentence.toUpperCase() + '\n';
21.    CoutToClient.writeBytes(capitalizedSentence);
22.    }
23.    }
24.    }

```

Программа TCPServer имеет много общего с программой TCPClient; рассмотрим ее подробнее, опустив строки, общие с программой TCPClient и описанные выше.

```

1. ServerSocket welcomeSocket = new ServerSocket (6789);

```

Здесь происходит создание объекта welcomeSocket типа ServerSocket. Объект welcomeSocket представляет собой впускающий сокет, с помощью которого клиент устанавливает первоначальный контакт с сервером. Для этого сокета используется порт с номером 6789, совпадающим с номером порта сокета клиента (мы раскроем причины совпадения номеров портов сокетов клиента и сервера в главе 3). Протокол TCP устанавливает прямой виртуальный канал между сокетом clientSocket на клиентской стороне и connectionSocket на серверной стороне, после чего клиент и сервер могут свободно осуществлять обмен информацией. После создания сокета connectionSocket сервер может вновь использовать сокет welcomeSocket для инициирования других соединений с клиентами (приведенная программа не обрабатывает новых соединений, однако ее можно модифицировать соответствующим образом).

Далее программа создает несколько потоковых объектов, аналогичных clientSocket.

```

1. capitalizedSentence = clientSentence.toUpperCase() + '\n';

```

Та команда основная в приложении. Она выполняет получение строки, передаваемой клиентом, перевод строки в верхний регистр с помощью метода tollpperCase() и добавление в конец символа возврата каретки. Все остальные команды программы являются периферийными и не касаются взаимодействия сервера с клиентом.

Для того чтобы протестировать совместную работу наших программ, следует поместить их на разные хосты и указать в программе TCPClient имя хоста-сервера. Затем необходимо запустить программу TCPServer, чтобы создать серверный процесс. Серверный процесс будет находиться в состоянии ожидания до тех пор, пока клиентский процесс не иницирует TCP-соединение; для этого, в свою очередь, нужно запустить программу TCPClient. Наконец, чтобы проверить наше приложение в действии, следует ввести на стороне клиента строку, оканчивающуюся символом возврата каретки (он вводится нажатием клавиши Enter).

2- Создаем клиент-сервер на сокетах

1. Что такое сокеты?
2. Создаем сервер
3. Создаем клиент
4. Замечания

Что такое сокеты?

Для начала давайте определим что такое сервер и клиент. Итак, сервер - это специальная программа, обычно запущенная на отдельном компьютере (хосте, от слова host(eng.) - хозяин), и выполняющая некий круг задач. Клиент, в свою очередь - программа, которая запрашивает сервер выполнить то или иное действие (задачу) и вернуть полученные данные клиенту. На хосте для работы сервера обычно выделяется порт (port). К этому порту и должен будет обращаться клиент. Клиент для связи с портом хоста, который соединен в свою очередь с нужным сервером (программой), создает сокет.

В целом алгоритм работы системы клиент-сервер выглядит следующим образом:

1. Сервер подключается к порту на хосте и ждет соединения с клиентом;
2. Клиент создает сокет и пытается соединить его с портом на хосте;
3. Если создание сокета прошло успешно, то сервер переходит в режим ожидания команд от клиента;
4. Клиент формирует команду и передает ее серверу, переходит в режим ожидания ответа;
5. Сервер принимает команду, выполняет ее и пересылает ответ клиенту.
6. и т.д.

Теперь попробуем создать сервер и клиент.

Создаем сервер

Для создания сокетов и управления ими в Java есть специальные классы `java.net.Socket` и `java.net.ServerSocket`. Первый для клиента, второй для сервера. Так же нам будут необходимы два класса из пакета `java.io.*`: `BufferedReader` и `PrintWriter` для чтения/записи в сокет (думаю, что читатель уже знаком с этими классами).

Для начала подключимся к порту хоста. Сделать это можно с помощью конструктора класса `ServerSocket`. Обратите внимание, что конструктор выбрасывает исключение типа `IOException`, т.е. нам понадобится блок `try - catch`:

```
1. ServerSocket servers;
2. try {
3.     servers = new ServerSocket(4444);
4. } catch (IOException e) {
5.     System.out.println("Couldn't listen to port 4444");
6.     System.exit(-1);
7. }
```

После успешного подключения к порту сервер должен ждать подключения от клиента. Сделать это можно так:

```
1. Socket fromclient;
2. try {
3.     System.out.print("Waiting for a client...");
4.     fromclient= servers.accept();
5.     System.out.println("Client connected");
6. } catch (IOException e) {
7.     System.out.println("Can't accept");
8.     System.exit(-1);
9. }
```

Рассмотрим этот блок подробнее. Метод `servers.accept()` позволяет серверу следить за портом, или иначе говоря ждать подключения клиента. Как только клиент подключается - сокет для клиента сразу же создается. В противном случае выбрасывается исключение `IOException`.

Как только клиент подключился к серверу, сервер должен создать потоки ввода и вывода для связи с ним. Это можно сделать следующим образом:

```
1. BufferedReader in;
2. PrintWriter out;
3. in=new BufferedReader(new
4.   InputStreamReader(fromclient.getInputStream()));
5. out = new PrintWriter(fromclient.getOutputStream(),true);
```

И далее можно просто считывать данные из потока in и записывать данные в out.

Исходный код сервера:

```
1. import java.io.*;
2. import java.net.*;
3.
4. public class Server {
5.
6.   public static void main(String[] args) throws IOException {
7.     System.out.println("Welcome to Server side");
8.     BufferedReader in = null;
9.     PrintWriter out= null;
10.
11.     ServerSocket servers = null;
12.     Socket fromclient = null;
13.
14.     // create server socket
15.     try {
16.       servers = new ServerSocket(4444);
17.     } catch (IOException e) {
18.       System.out.println("Couldn't listen to port 4444");
19.       System.exit(-1);
20.     }
21.
22.     try {
23.       System.out.print("Waiting for a client...");
24.       fromclient= servers.accept();
25.       System.out.println("Client connected");
26.     } catch (IOException e) {
27.       System.out.println("Can't accept");
28.       System.exit(-1);
29.     }
30.
31.     in = new BufferedReader(new
32.       InputStreamReader(fromclient.getInputStream()));
```

```

32.         out = new
33.         PrintWriter(fromclient.getOutputStream(),true);
34.         String          input,output;
35.
36.         System.out.println("Wait for messages");
37.         while ((input = in.readLine()) != null) {
38.             if (input.equalsIgnoreCase("exit")) break;
39.             out.println("S ::: "+input);
40.             System.out.println(input);
41.         }
42.         out.close();
43.         in.close();
44.         fromclient.close();
45.         servers.close();
46.     }
47. }

```

Создаем клиент

Для создания клиента достаточно класса Socket и двух классов для ввода/вывода (см. Создаем сервер). Также необходимо знать имя компьютера (хоста), на котором запущен сервер и номер порта.

Конструктор сокета имеет два параметра: имя хоста и номер порта. Опять таки, конструктор выбрасывает исключение типа IOException.

```

1.     Socket fromserver = null;
2.     fromserver = new Socket("localhost",4444);

```

Далее аналогичным образом, как для сервера, создаем потоки ввода вывода. И можно записывать/считывать данные.

Исходный код клиента:

```

1.     import java.io.*;
2.     import java.net.*;
3.
4.     public class client {
5.         public static void main(String[] args)
6.             throws IOException {
7.
8.             System.out.println("Welcome to Client side");
9.
10.            Socket fromserver = null;
11.

```

```

12.         if (args.length==0) {
13.             System.out.println("use: client hostname");
14.             System.exit(-1);
15.         }
16.
17.         System.out.println("Connecting to... "+args[0]);
18.
19.         fromserver = new Socket(args[0],4444);
20.         BufferedReader in  = new
21.             BufferedReader(new
22.                 InputStreamReader(fromserver.getInputStream()));
23.         PrintWriter      out = new
24.             PrintWriter(fromserver.getOutputStream(),true);
25.         BufferedReader inu = new
26.             BufferedReader(new InputStreamReader(System.in));
27.
28.         String fuser,fserver;
29.
30.         while ((fuser = inu.readLine())!=null) {
31.             out.println(fuser);
32.             fserver = in.readLine();
33.             System.out.println(fserver);
34.             if (fuser.equalsIgnoreCase("close")) break;
35.             if (fuser.equalsIgnoreCase("exit")) break;
36.         }
37.
38.         out.close();
39.         in.close();
40.         inu.close();
41.         fromserver.close();
42.     }
43. }

```

Замечания

Клиент-сервер организован на примере Echo server (Эхо сервер).
Клиент получает обратно строку переданную серверу.

Несколько замечаний по кодам:

Обратите внимание на конструкцию методов main :

```

1. public static void main(String[] args) throws IOException {}

```

throws IOException позволит не использовать блоки try-catch для ловки исключения IOException в самом методе, что достаточно удобно.

В клиентской части используется параметр командной строки для указания имени хоста. Например, если Вы запускаете сервер и клиент на одном компьютере, то клиент надо запускать так:

```
1. java client localhost
```

3- Клиент-сервер на java

Пример клиент-серверного приложения на JAVA.

Для начала нужно создать два приложения Client и Server:

```
1. touch Client.java
2.
3. touch Server.java
```

Приложение Server будет висеть на определенном порту и слушать все обращения к нему, обрабатывать их и возвращать результат клиенту, делается это через socket-ы.

Содержимое файла Server.java:

```
1. import java.net.*;
2.
3. import java.io.*;
4.
5.
6. class Server {
7.
8. public static void main(String[] args) {
9.
10.  try{
11.
12.  ServerSocket server = null;
13.
14.  Socket client;
15.
16.  server = new ServerSocket(1234); // слушаем порт 1234
17.
18.  client = server.accept();
```



```
19.
20.
21. System.out.println("Got client");
22.
23. PrintWriter out = null;
24.
25. BufferedReader in = null;
26.
27. in = new BufferedReader(new
28. InputStreamReader(client.getInputStream()));
29.
30. out = new PrintWriter(client.getOutputStream(),true);
31.
32. String input, output;
33.
34.
35. while ((input = in.readLine()) != null) {
36.
37. if (input.equalsIgnoreCase("exit")) {
38.
39. break;
40.
41. }
42.
43.
44. // Ответ клиенту будет в виде Server:
45. сообщение_клиента
46.
47. out.println("Server:" + input);
48.
49. System.out.println(input);
50.
51. }
52.
53. out.close();
54.
55. in.close();
56.
57. client.close();
58.
59. server.close();
60.
61. System.out.println("Server closed");
62.
63. } catch(Exception e){
64.
```

```
65. System.out.println(e.getMessage());
66.
67. }
68.
69. }
70.
71. }
```

Приложение Client будет коннектиться к определенному серверу по ip, на определенный ранее порт.

Код файла Client.java:

```
1. import java.net.*;
2. import java.io.*;
3.
4. class Client {
5.
6. public static void main(String[] args) {
7.
8. try {
9.
10.     Socket clientSocket = null;
11.
12.     // 127.0.0.1 - IP где запущен Server, 1234 - порт
13.
14.     clientSocket = new Socket("127.0.0.1", 1234);
15.
16.     BufferedReader in = new BufferedReader(new
17.         InputStreamReader(clientSocket.getInputStream()));
18.     PrintWriter out = new
19.         PrintWriter(clientSocket.getOutputStream(), true);
20.
21.     BufferedReader inu = new BufferedReader(new
22.         InputStreamReader(System.in));
23.
24.     String fuser, fserver;
25.
26.     out = new PrintWriter(clientSocket.getOutputStream(),
27.         true);
28.
29.     in = new BufferedReader(new
30.         InputStreamReader(clientSocket.getInputStream()));
31.
32.     while ((fuser = inu.readLine()) != null) {
33.         out.println(fuser);
```

```
34.     fserver = in.readLine();
35.     System.out.println(fserver);
36.     if (fuser.equalsIgnoreCase("close")) {
37.         break;
38.     }
39.
40.     if (fuser.equalsIgnoreCase("exit")) {
41.         break;
42.     }
43.
44.     }
45.
46.     out.close();
47.     in.close();
48.     inu.close();
49.     clientSocket.close();
50.     } catch (Exception e) {
51.         System.out.println(e.getMessage());
52.     }
53.     }
54.     }
```

Теперь приложения нужно скомпилировать

```
1. javac Server.java
2. javac Client.java
```

Запустить сервер

```
1. java Server
```

Запустить клиент

```
1. java Client
2. message test
3. Server: message test
```

Полученные сервером сообщения можно обработать как угодно, переслать другому (всем) клиентам, запустить в системе на выполнение, etc.

3. Задания для выполнения

1. Получить вариант задания у преподавателя.
2. Разработать прикладную программу в соответствии с заданием.
3. Написать и отладить программу на ЭВМ.
4. Сдать работающую программу преподавателю.
5. Подготовить и защитить отчет.

4. Варианты индивидуальных заданий

1. Организовать взаимодействие типа клиент - сервер. Клиент делает запрос серверу на выполнение какой-либо команды. Сервер выполняет эту команду и возвращает результаты клиенту.
2. Организовать взаимодействие типа клиент - сервер. Клиент делает запрос серверу о передаче файлов с определенным расширением из указанной директории. Сервер сканирует указанную директорию и отправляет клиенту список файлов, удовлетворяющих запросу.
3. Организовать взаимодействие типа клиент - сервер. Сервер при подключении к нему нового клиента высылает список IP-адресов уже подключенных клиентов. А остальным клиентам рассылается сообщение в виде IP-адреса о том, что подключился такой-то клиент.
4. Организовать взаимодействие типа клиент - сервер. Клиент при входе в связь с сервером должен ввести пароль. Разрешено сделать три попытки. Если пароль не верен, сервер должен блокировать IP-адрес клиента на 5 минут.
5. Организовать взаимодействие типа клиент - сервер. Клиенты подключаются к первому серверу, и передают запрос на получение определенного файла. Если этого файла нет, сервер подключается ко второму серверу и ищет файл там. Затем либо найденный файл пересылается клиенту, либо высылается сообщение, то такого файла нет.
6. Организовать взаимодействие типа клиент - сервер. К серверу одновременно может подключиться только один клиент. Остальные клиенты заносятся в очередь, и им высылается сообщение об ожидании освобождения сервера.
7. Организовать взаимодействие типа клиент - сервер. Клиент отправляет строку серверу. Сервер отправляет данную строку на другие сервера,

список которых хранится в файле, а там уже осуществляется поиск файлов содержащих данную строку. Результаты поиска отсылаются клиенту.

8. Эмуляция DNS сервера. Клиент подсоединяется к серверу, IP которого хранится в файле dns.url и делает ему запрос на подключение к серверу "Имя сервера". DNS-сервер имеет список, хранящийся в файле о соответствии имен серверов и IP-адресов. Если в списке нет "имени сервера" запрошенного клиентом, то сервер DNS подключается последовательно к другим серверам, хранящимся в файле dns.url и т.д. Если сервер не найден, клиенту возвращается соответствующее сообщение.
9. Организовать чат. К серверу подключаются клиенты. При подключении клиента сервер спрашивает имя, под которым клиент будет известен в соединении. Сервер хранит IP-адреса подключаемых клиентов и их имена. Все сообщения каждого клиента рассылаются остальным в виде "имя клиента" - сообщение". Сообщения рассылаются сервером всем клиентам также при вхождении в связь нового клиента, и выходе какого-либо клиента.

5. Контрольные вопросы

1. Какова структура IP-адреса?
2. Как поместить и извлечь IP-адрес из структуры сокета?
3. В чем разница между моделями TCP-соединения и дейтаграмм?
4. Каковы основные шаги межпроцессного взаимодействия в модели TCP-соединения?
5. Каковы основные шаги межпроцессного взаимодействия в модели дейтаграмм?
6. Как занести в структуру сокета IP-адрес своего компьютера?
7. Каким образом извлечь информацию о клиенте после установки TCP-соединения?
8. Какова реакция системных вызовов отправки и приема сообщений в модели TCP-соединения при разрыве связи?

6. Список рекомендуемой литературы

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
Владимирский государственный университет имени А.Г. и Н.Г. Столетовых

Кафедра информационных систем
и программной инженерии

ЛАБОРАТОРНАЯ РАБОТА №4 ПРОТОКОЛЫ МАРШРУТИЗАЦИИ

Владимир 2015 г

1. Цель работы

Изучить процесс маршрутизации в сети, его задачи, маршруты, протоколы, алгоритмы, а также рассмотреть алгоритм Беллмана-Форда.

2. Общие сведения

2.1. Основы маршрутизации

Сети соединяются между собой специальными устройствами, называемыми маршрутизаторами. **Маршрутизатор** – это устройство, которое собирает информацию о топологии межсетевых соединений и пересылает пакеты сетевого уровня в сеть назначения.

Чтобы передать сообщение от отправителя, находящегося в одной сети, получателю, находящемуся в другой сети, нужно совершить некоторое количество транзитных передач между сетями, или **хопов** (от слова hop - прыжок), каждый раз выбирая подходящий маршрут. Таким образом, **маршрут** представляет собой последовательность маршрутизаторов, через который проходит пакет.

Сетевой уровень должен обеспечить доставку пакета:

- между любыми двумя узлами сети с произвольной топологией;
- между любыми двумя сетями в составной сети.

Сеть – совокупность компьютеров, использующих для обмена данными единую сетевую технологию.

Маршрут – последовательность прохождения пакетом маршрутизаторов в составной сети.

2.2. Задачи маршрутизации

Проблема выбора наилучшего пути называется маршрутизацией, и ее решение является одной из главных задач сетевого уровня. Эта проблема осложняется тем, что **самый короткий путь - не всегда самый лучший**. Критерием при выборе маршрута может служить *время передачи данных*. Время зависит от пропускной способности каналов связи и интенсивности трафика, которая может с течением времени изменяться. Выбор маршрута может осуществляться и по другим критериям, таким как надежность

передачи. Функции сетевого уровня шире, чем функции передачи сообщений по связям с нестандартной структурой, которые мы рассмотрели на примере объединения нескольких локальных сетей. Сетевой уровень также решает задачи согласования разных технологий, упрощения адресации в крупных сетях и создания надежных и гибких барьеров на пути нежелательного трафика между сетями.

2.3. Маршруты движения пакетов

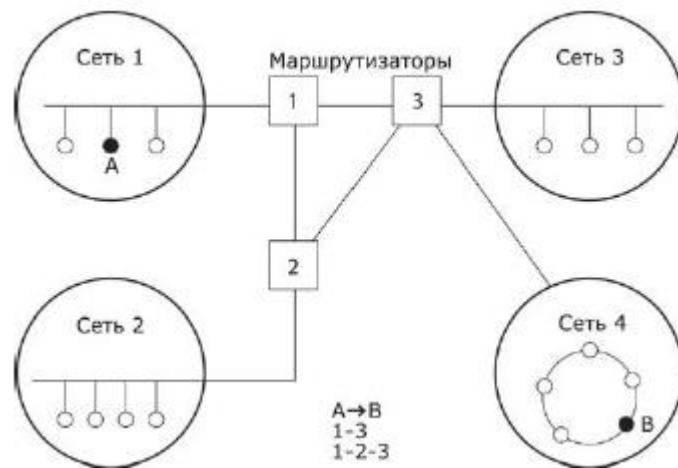


Рисунок 4.1.

На рисунке 4.1. показаны четыре сети, связанные тремя маршрутизаторами. Между узлами А и В данной сети пролегает два маршрута:

- первый - через маршрутизаторы 1 и 3;
- второй - через маршрутизаторы 1, 2 и 3.

2.4. Маршрутизация

Маршрутизатор - это устройство, распределяющее пакеты по сети с помощью информации сетевого уровня. Маршрутизатор извлекает данные об адресации сетевого уровня из пакета данных. В маршрутизаторе также имеются алгоритмы, называемые протоколами маршрутизации, с помощью которых он строит таблицы. В соответствии с этими таблицами и определяется тот маршрут, по которому должен быть направлен пакет, чтобы достичь конечного пункта назначения. Если маршрутизатор является многопротокольным, т.е. понимает несколько форматов адресов сетевого уровня и может работать с несколькими протоколами маршрутизации, к каковым и относятся маршрутизаторы компании Cisco, то он хранит

отдельные таблицы маршрутизации для каждого из протоколов сетевого уровня, маршрутизация которого осуществляется.

Оценивая возможные пути, протоколы маршрутизации используют информацию о топологии сетей. Эта информация может конфигурироваться сетевым администратором или собираться посредством динамических процессов, исполняемых в сети.

Маршрутизируемый протокол - любой сетевой протокол, который обеспечивает в адресе сетевого уровня достаточно информации, чтобы позволить передать пакет от одной хост-машины к другой на основе принятой схемы адресации. Маршрутизируемый протокол определяет формат и назначение полей внутри пакета. В общем случае пакеты переносятся от одной конечной системы к другой. Примером маршрутизируемого протокола является межсетевой протокол IP.

Протокол маршрутизации - поддерживает маршрутизируемый протокол за счет предоставления механизмов коллективного использования маршрутной информации. Сообщения протокола маршрутизации циркулируют между маршрутизаторами. Протокол маршрутизации позволяет маршрутизаторам обмениваться информацией с другими маршрутизаторами с целью актуализации и ведения таблиц. Примерами протоколов маршрутизации являются протокол маршрутной информации (RIP), протокол внутренней маршрутизации между шлюзами (IGRP), усовершенствованный протокол внутренней маршрутизации между шлюзами (EIGRP) и протокол маршрутизации с выбором кратчайшего пути (OSPF).

2.5. Статические и динамические маршруты

Статическая информация администрируется вручную. Сетевой администратор вводит ее в конфигурацию маршрутизатора. Если изменение в топологии сети требует актуализации статической информации, то администратор сети должен вручную обновить соответствующую запись о статическом маршруте.

Динамическая информация работает по-другому. После ввода администратором сети команд, запускающих функцию динамической маршрутизации, сведения о маршрутах обновляются процессом маршрутизации автоматически сразу после поступления из сети новой информации. Изменения в динамически получаемой информации

распространяются между маршрутизаторами как часть процесса актуализации данных.

Маршрут по умолчанию - запись в таблице маршрутизации, которая используется для направления кадров, которые не имеют в таблице маршрутизации явно указанного следующего перехода. Маршруты по умолчанию могут устанавливаться как результат статического конфигурирования, выполняемого администратором.

Вообще говоря, содержание сведений обо всех других сетевых комплексах, доступных через сеть Internet, излишне и неразумно, если не невозможно. Вместо сведений о каждой конкретной сети каждому маршрутизатору компании X сообщается маршрут по умолчанию, с помощью которого он может добраться до любого неизвестного пункта назначения, направляя пакет в сеть Internet.

2.6. Адаптация к изменениям топологии

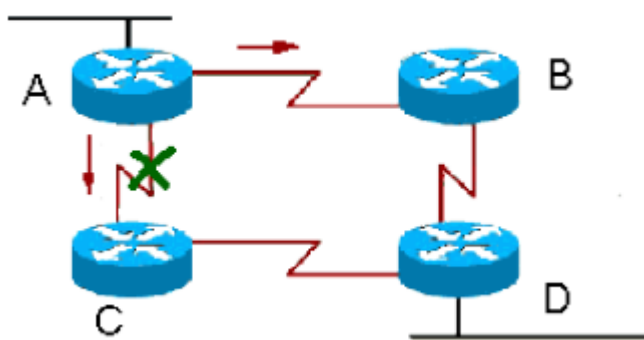


Рисунок 4.2.

Динамическая маршрутизация позволяет маршрутизаторам автоматически использовать резервные маршруты.

Сеть по-разному адаптируется к изменениям в топологии, в зависимости от того, используется статическая или динамическая информация.

Статическая маршрутизация позволяет маршрутизаторам правильно направлять пакет от сети к сети. Маршрутизатор просматривает свою таблицу маршрутизации и, следуя содержащимся там статическим данным, ретранслирует пакет маршрутизатору D (Рис. 4.2.). Маршрутизатор D делает то же самое и ретранслирует пакет маршрутизатору C. Маршрутизатор C доставляет пакет хост-машине получателя.

Но что произойдет, если путь между маршрутизаторами А и D становится непроходимым? Ясно, что маршрутизатор А не сможет ретранслировать пакет маршрутизатору D по статическому маршруту. Связь с сетью пункта назначения будет невозможна до тех пор, пока маршрутизатор А не будет реконфигурирован на ретрансляцию пакетов маршрутизатору В.

Динамическая маршрутизация обеспечивает более гибкое и автоматическое поведение. В соответствии с таблицей маршрутизации, генерируемой маршрутизатором А, пакет может достичь своего пункта назначения по предпочтительному маршруту через маршрутизатор D. Однако к пункту назначения возможен и другой путь через маршрутизатор В. Когда маршрутизатор А узнает, что канал на маршрутизатор D нарушен, он перестраивает свою таблицу маршрутизации, делая предпочтительным путь к пункту назначения через маршрутизатор В, а маршрутизаторы продолжают слать пакеты по этому каналу связи. Когда путь между маршрутизаторами А и D восстанавливается, маршрутизатор А может снова изменить свою таблицу маршрутизации и указать предпочтительным путь к сети пункта назначения против часовой стрелки через маршрутизаторы D и С. Протоколы динамической маршрутизации могут также перенаправлять трафик между различными путями в сети.

2.7. Операции динамической маршрутизации

Успех динамической маршрутизации зависит от двух основных функций маршрутизатора:

- Ведение таблицы маршрутизации;
- Своевременное распространение информации - в виде пакетов актуализации - среди других маршрутизаторов.

В обеспечении коллективного пользования информацией о маршрутах динамическая маршрутизация полагается на протокол маршрутизации. Протокол маршрутизации определяет набор правил, используемых маршрутизатором при его общении с соседними маршрутизаторами.

Например, протокол маршрутизации описывает следующее:

- как посылаются пакеты актуализации;
- какие сведения содержатся в таких пакетах актуализации;

- когда следует посылать эту информацию;
- как определять получателей этих пакетов актуализации;

2.8. Представление расстояние с помощью метрики

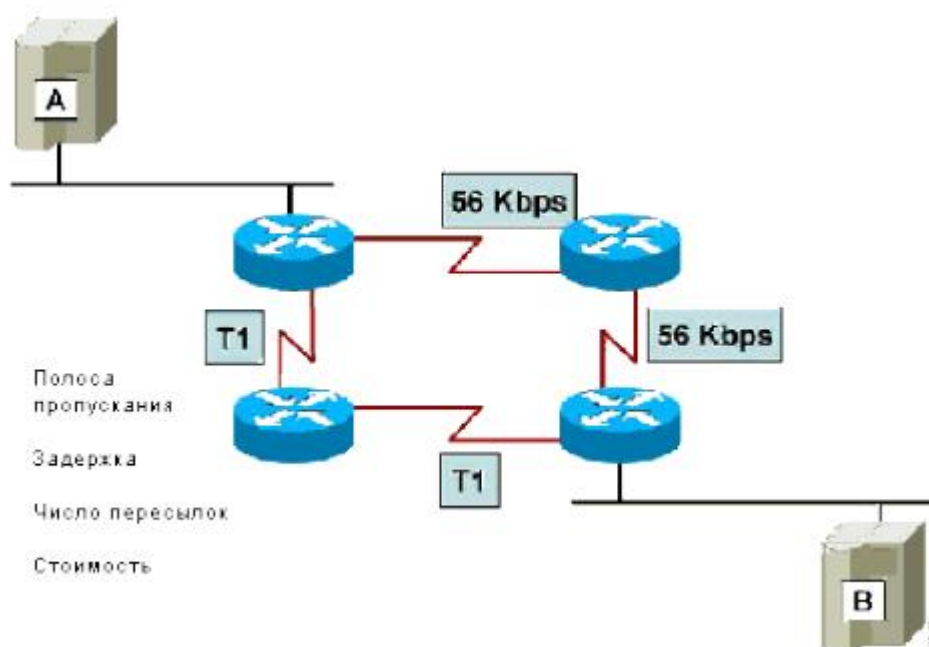


Рисунок 4.3.

Когда алгоритм маршрутизации обновляет таблицу маршрутизации, его главной целью является определение наилучшей информации для включения в таблицу. Каждый алгоритм маршрутизации интерпретирует понятие «наилучшая» по-своему. Для каждого пути в сети алгоритм генерирует число, называемое метрикой. Как правило, чем меньше величина этого числа, тем лучше путь.

Метрики могут рассчитываться на основе одной характеристики пути. Объединяя несколько характеристик, можно рассчитывать и более сложные метрики. Как показано на рисунке 4.3., при вычислении значения метрики используется несколько характеристик пути.

Наиболее общеупотребительными метриками, используемыми маршрутизаторами, являются следующие:

- Количество переходов - количество маршрутизаторов, через которые должен пройти пакет, чтобы дойти до получателя. Чем меньше

количество переходов, тем лучше путь. Для обозначения суммы переходов до пункта назначения используется термин длина пути.

- Полоса пропускания - пропускная способность канала передачи данных. Например, для арендуемой линии 64 Кбит/с обычно предпочтительным является канал типа T1 с полосой пропускания 1,544 Мбит/с.
- Задержка - продолжительность времени, требующегося для перемещения пакета от отправителя получателю.
- Нагрузка - объем действий, выполняемый сетевым ресурсом, например маршрутизатором или каналом.
- Надежность - темп возникновения ошибок в каждом сетевом канале. Тик - задержка в канале передачи данных, определяемая в машинных тактах IBM-подобного ПК (приблизительно 55 миллисекунд).
- Стоимость - произвольное значение, обычно основанное на величине полосы пропускания, денежной стоимости или результате других измерений, которое назначается сетевым администратором.

2.9. Протоколы маршрутизации

Большинство алгоритмов маршрутизации можно свести к трем основным алгоритмам.

- ✓ Подход на основе маршрутизации по вектору расстояния, в соответствии с которым определяются направление (вектор) и расстояние до каждого канала в сети.
- ✓ Подход на основе оценки состояния канала (также называемый выбором наикратчайшего пути), при котором воссоздается точная топология всей сети (или, по крайней мере, той части, где размещается маршрутизатор).
- ✓ Гибридный подход, объединяющий аспекты алгоритмов с определением вектора расстояния и оценки состояния канала.

Алгоритм маршрутизации является основой динамической маршрутизации. Как только вследствие роста, реконфигурирования или отказа изменяется топология сети, база знаний о сети должна изменяться тоже; это прерывает маршрутизацию.

Необходимо, чтобы знания отражали точное и непротиворечивое представление о новой топологии. В том случае, когда все маршрутизаторы используют непротиворечивое представление топологии сети, имеет место сходимости. Говорят, что сетевой комплекс сошелся, когда все имеющиеся в

нем маршрутизаторы работают с одной и той же информацией. Процесс и время, требующиеся для возобновления сходимости маршрутизаторов, меняются в зависимости от протокола маршрутизации. Для сети желательно обладать свойством быстрой сходимости, поскольку это уменьшает время, когда маршрутизаторы используют для принятия решений о выборе маршрута устаревшие знания, и эти решения могут быть неправильными, расточительными по времени или и теми и другими одновременно.

2.10. Алгоритмы маршрутизации по вектору расстояния

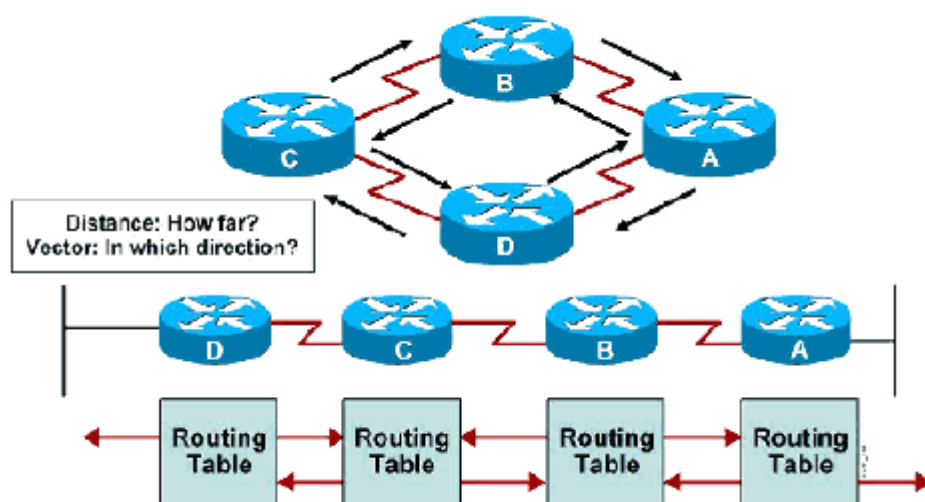


Рисунок 4.4.

Алгоритмы маршрутизации на основе вектора расстояния (также известные под названием алгоритмы Беллмана—Форда (Bellman-Ford algorithms)) предусматривают периодическую передачу копий таблицы маршрутизации от одного маршрутизатора другому. Регулярно посылаемые между маршрутизаторами пакеты актуализации сообщают обо всех изменениях топологии.

Каждый маршрутизатор получает таблицу маршрутизации от своего соседа. Например, на рисунке 4.4. маршрутизатор В получает информацию от маршрутизатора А. Маршрутизатор В добавляет величину, отражающую вектор расстояния (скажем, количество переходов), которая увеличивает вектор расстояния, и затем передает таблицу маршрутизации своему соседу - маршрутизатору С. Такой же процесс пошагово выполняется между соседними маршрутизаторами во всех направлениях.

Подобным образом алгоритм аккумулирует сетевые расстояния и поэтому способен поддерживать базу данных информации о топологии сети. Однако алгоритмы на основе вектора расстояния не позволяют маршрутизатору знать точную топологию всего сетевого комплекса.

2.11. Алгоритмы маршрутизации с учетом состояния канала связи

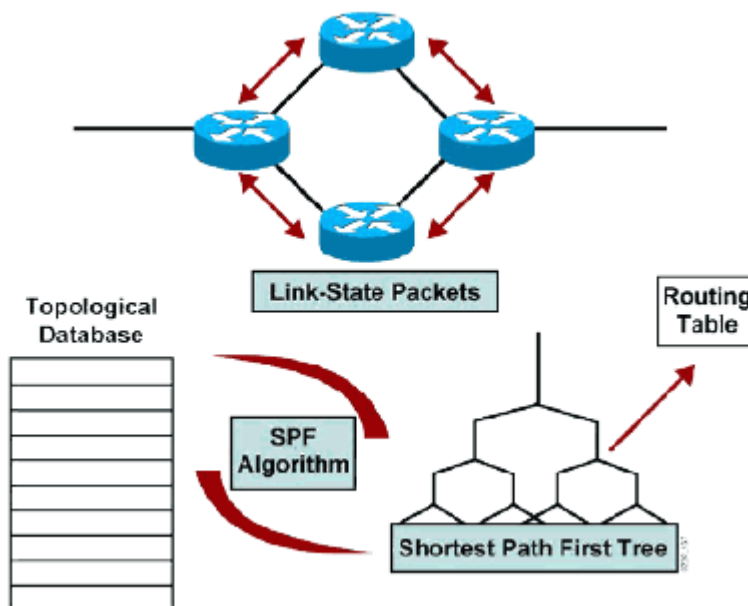


Рисунок 4.5.

Вторым основным алгоритмом, используемым для маршрутизации, является алгоритм с учетом состояния канала связи. Алгоритмы маршрутизации с учетом состояния канала связи, также известные под названием алгоритмов выбора первого кратчайшего пути (shortest path first (SPF) algorithms), поддерживают сложную базу данных топологической информации. И если алгоритмы с маршрутизацией по вектору расстояния работают с неконкретной информацией о дальних сетях, то алгоритмы маршрутизации с учетом состояния канала собирают полные данные о дальних маршрутизаторах и о том, как они соединены друг с другом.

Для выполнения маршрутизации с учетом состояния канала связи используются сообщения объявлений о состоянии канала (link-state advertisements, LSA), база данных топологии, SPF-алгоритм, результирующее SPS-дерево и таблица маршрутизации, содержащая пути и порты к каждой сети (рис. 4.5.).

2.12. Сравнение маршрутизации по вектору расстояния с учетом состояния канала связи

Сравнивать маршрутизацию по вектору расстояния и маршрутизацию с учетом состояния канала связи можно в нескольких ключевых областях:

- Процесс маршрутизации по вектору расстояния получает все топологические данные из информации, содержащейся в таблицах маршрутизации соседей. Процесс маршрутизации с учетом состояния канала связи получает широко представление обо всей топологии сетевого комплекса, собирая данные из всех необходимых LSA-пакетов.
- Процесс маршрутизации по вектору расстояния определяет лучший путь с помощью сложения получаемых метрик по мере того, как таблица движется от одного маршрутизатора к другому. При использовании маршрутизации с учетом состояния канала каждый маршрутизатор работает отдельно, вычисляя свой собственный кратчайший путь к пункту назначения.
- В большинстве протоколов маршрутизации по вектору расстояния пакеты актуализации, содержащие сведения об изменениях топологии, являются периодически посылаемыми пакетами актуализации таблиц маршрутизации. Эти таблицы передаются от одного маршрутизатора к другому, что обычно приводит к более медленной сходимости.
- В протоколах маршрутизации с учетом состояния канала связи пакеты актуализации обычно генерируются и рассылаются по факту возникновения изменения топологии. Относительно небольшие LSA-пакеты передаются всем другим маршрутизаторам, что, как правило, приводит к более быстрой сходимости при любом изменении топологии сетевого комплекса.

Маршрутизация по вектору расстояния	Маршрутизация с учетом состояния канала связи
Видит топологию сети глазами соседних маршрутизаторов	Получает общий вид топологии всей сети
Суммирует вектор расстояния от одного маршрутизатора к другому	Вычисляет кратчайший путь до других маршрутизаторов
Частые периодические обновления топологической информации, медленная сходимость	Обновления инициируются фактом изменения топологии; быстрая сходимость

Передаёт копии таблицы маршрутизации только соседним маршрутизаторам	Передаёт пакеты с информацией об актуальном состоянии канала связи всем другим маршрутизаторам
--	--

2.13. Алгоритм Беллмана-Форда

Описание алгоритма:

Предположим, что вершина 1 является вершиной-входом, и требуется найти длины кратчайших путей от вершины 1 до каждой другой вершины графа. Для этого алгоритма дуговые расстояния могут быть как положительными, так и отрицательными, но не должно быть циклов отрицательной длины. Предположим, что если в графе отсутствует та или иная дуга, то её вес равен бесконечно большому числу. Основная идея алгоритма Беллмана - Форда состоит в том, чтобы сначала найти длины кратчайших путей, при условии, что пути содержат не более двух дуг, не более трех дуг и т.д. Кратчайший путь, при условии, что он содержит не более h дуг, будем называть кратчайшим путем в графе. Согласно идее Беллмана - Форда, Р. Прим предложил алгоритм нахождения длин кратчайших путей, ведущих из произвольной вершины графа во все остальные его вершины (в которые есть пути из вершины-входа), состоящий в присвоении вершинам некоторых потенциалов. Признаком окончания алгоритма является остановка процесса изменения потенциалов.

Алгоритм, предложенный Р. Примом, состоит в выполнении следующих операций:

1) Вершине-входу присваиваем потенциал $\varphi_1 = 0$; остальным вершинам – потенциал $\varphi_i = \infty$, $i = 1 \dots n$, где n – число вершин в рассматриваемом графе.

2) Для каждой вершины i , смежной с вершиной j из массива вершин L , находим $\varphi_j + l(j, i)$ и проверяем неравенство

$$\varphi_j + l(j, i) < \varphi_i .$$

Если неравенство выполняется, то

$$\varphi_i := \varphi_j + l(j, i).$$

Вершину i заносим в массив L_1 .

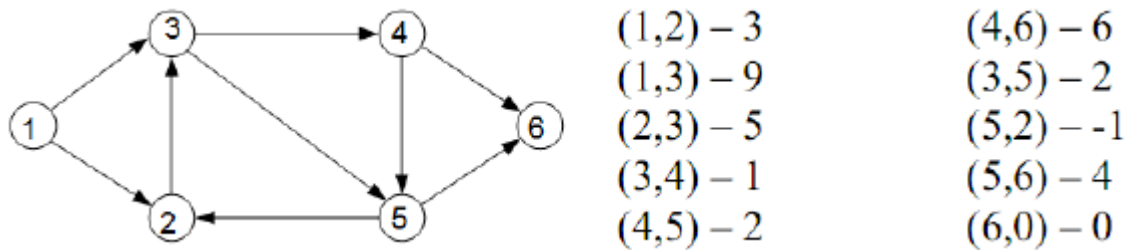
3) Проверяем массив $L_I \neq III$. Если неравенство выполняется $L := L_I$, осуществляем переход к п. 2.

4) Конец алгоритма нахождения кратчайших путей.

Пример:

Рассмотрим работу алгоритма на примере.

Дано: Допустим, имеется граф, представленный списком дуг:



Решение: Результаты работы алгоритма по шагам вычислений сведем в таблицу.

Шаг вычислений	Массив L	Потенциалы вершин						Массив L ₁
		1	2	3	4	5	6	
0	III	0	∞	∞	∞	∞	∞	1
1	1	0	3	9	∞	∞	∞	2, 3
2	2, 3	0	3	8	10	11	∞	3, 4, 5
3	2, 4, 5	0	3	8	9	10	15	4, 5, 6
4	4, 5, 6	0	3	8	9	10	14	6
5	6	0	3	8	9	10	14	III

На нулевом шаге производится инициализация согласно п.1 алгоритма. Далее на первом шаге вычислений рассматриваются вершины, смежные с вершиной-входом, потенциалы рассматриваемых вершин становятся равными длине соединяющей их дуги, если таковое существует, в противном случае – потенциал вершины остается равным ∞ .

На втором шаге рассматриваем вершины, достижимые из вершины-входа при помощи пути, состоящего из двух дуг. Таковыми являются:

- 1) 1 – 2 – 3: потенциал вершины 3 изменяется $\varphi_3 = 8$;
- 2) 1 – 3 – 4: потенциал вершины 4 изменяется $\varphi_4 = 10$;
- 3) 1 – 3 – 5: потенциал вершины 5 изменяется $\varphi_5 = 11$.

Третий шаг:

- 1) 1 – 3 – 4 – 6: потенциал вершины 6 изменяется $\varphi_6 = 16$;
- 2) 1 – 2 – 3 – 5: потенциал вершины 5 изменяется $\varphi_5 = 10$;
- 3) 1 – 3 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 10$);
- 4) 1 – 3 – 4 – 5: потенциал вершины 5 не изменится ($\varphi_5 < 12$);
- 5) 1 – 3 – 5 – 6: потенциал вершины 6 изменяется $\varphi_6 = 15$.

Четвертый шаг:

- 1) 1 – 2 – 3 – 5 – 6: потенциал вершины 6 изменяется $\varphi_6 = 14$;
- 2) 1 – 2 – 3 – 4 – 5: потенциал вершины 5 не изменится $\varphi_5 < 11$;
- 3) 1 – 2 – 3 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 9$);
- 4) 1 – 3 – 5 – 2 – 3: потенциал вершины 3 не изменится ($\varphi_3 < 15$);
- 5) 1 – 3 – 4 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 11$).

Пятый шаг:

- 1) 1 – 3 – 5 – 2 – 3 – 5: потенциал вершины 5 не изменится ($\varphi_5 < 17$);
- 2) 1 – 3 – 5 – 2 – 3 – 4: потенциал вершины 4 не изменится ($\varphi_4 < 16$);
- 3) 1 – 3 – 4 – 5 – 2 – 3: потенциал вершины 3 не изменится ($\varphi_3 < 14$);
- 4) 1 – 2 – 3 – 4 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 10$);
- 5) 1 – 2 – 3 – 4 – 5 – 6: потенциал вершины 6 не изменится ($\varphi_6 = 14$);
- 6) 1 – 2 – 3 – 5 – 2 – 3: потенциал вершины 3 не изменится ($\varphi_3 < 14$).

На пятом шаге вычислений не произошло изменений ни в одном из потенциалов вершин графа, делаем вывод об окончании работы алгоритма.

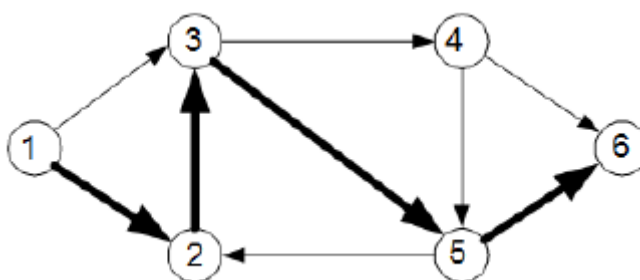
Восстановим кратчайший путь из вершины 6 в вершину 1. Для этого для каждой вершины j определим такую вершину i , что будет выполняться правило

$$\varphi_i := \varphi_j + l(j, i).$$

Следуя правилу, последовательно находим кратчайший путь:

- 1) вершина 6: $i = 5$;
- 2) вершина 5: $i = 3$;
- 3) вершина 3: $i = 2$;
- 4) вершина 2: $i = 1$.

Кратчайшим путем является путь: 1 – 2 – 3 – 5 – 6



3. Задания для выполнения

Дан список дуг с указанием их длин. Необходимо составить по нему рисунок ориентированного графа. Найдите для графа наименьший путь от вершины-входа до вершины с максимальным номером. ($\varphi = 0$)

4. Варианты индивидуальных заданий

- | | |
|--|--|
| <p>1. $(0;1) - 3, (0;2) - 9, (1;2) - 5,$
 $(2;4) - 1, (1;3) - 8, (2;3) - 2,$
 $(3;5) - 4, (4;5) - 6.$</p> <p>2. $(0;1) - 4, (0;2) - 5, (1;2) - 8,$
 $(2;4) - 3, (1;3) - 11, (2;3) - 5,$
 $(3;5) - 3, (4;5) - 6.$</p> <p>3. $(0;1) - 3, (0;2) - 9, (1;2) - 12,$
 $(2;4) - 1, (1;3) - 2, (2;3) - 3,$</p> | <p>9. $(0;1) - 2, (0;2) - 7, (2;1) - 1,$
 $(2;5) - 6, (1;5) - 12, (5;4) - 10,$
 $(5;3) - 5, (3;4) - 4, (4;6) - 2,$
 $(3;6) - 7.$</p> <p>10. $(0;1) - 4, (0;2) - 2, (2;1) - 1,$
 $(2;5) - 7, (1;5) - 5, (5;4) - 4,$
 $(5;3) - 1, (3;4) - 4, (4;6) - 3,$
 $(3;6) - 7.$</p> |
|--|--|

- (3;5) – 10, (4;5) – 5.
4. (0;1) – 6, (0;2) – 2, (2;1) – 3,
(2;4) – 6, (1;3) – 1, (2;3) – 5,
(3;5) – 8, (4;5) – 7.
 5. (0;1) – 6, (0;2) – 5, (1;2) – 1,
(2;4) – 6, (1;3) – 7, (2;3) – 6,
(3;5) – 8, (4;5) – 7.
 6. (0;1) – 3, (0;2) – 2, (2;1) – 1,
(2;5) – 3, (1;5) – 4, (5;4) – 8,
(5;3) – 5, (3;4) – 3, (4;6) – 2,
(3;6) – 4.
 7. (0;1) – 10, (0;2) – 5, (2;1) – 4,
(2;5) – 8, (1;5) – 3, (5;4) – 4,
(5;3) – 2, (3;4) – 1, (4;6) – 5,
(3;6) – 7.
 8. (0;1) – 3, (0;2) – 2, (2;1) – 2,
(2;5) – 12, (1;5) – 8, (5;4) – 2,
(5;3) – 6, (3;4) – 1, (4;6) – 8,
(3;6) – 3.
 11. (0;2) – 2, (0;1) – 7, (2;1) – 4,
(2;4) – 9, (1;3) – 3, (3;4) – 1,
(4;6) – 2, (3;5) – 8, (6;5) – 4,
(6;7) – 10, (5;7) – 5.
 12. (0;2) – 10, (0;1) – 5, (2;1) – 1,
(2;4) – 4, (1;3) – 3, (3;4) – 5,
(4;6) – 3, (3;5) – 10, (6;5) – 10,
(6;7) – 5, (5;7) – 1.
 13. (0;2) – 4, (0;1) – 6, (2;1) – 4,
(2;4) – 6, (1;3) – 3, (3;4) – 2,
(4;6) – 4, (3;5) – 7, (6;5) – 3,
(6;7) – 8, (5;7) – 5.
 14. (0;2) – 3, (0;1) – 1, (2;1) – 4,
(2;4) – 2, (1;3) – 6, (3;4) – 4,
(4;6) – 3, (3;5) – 6, (6;5) – 4,
(6;7) – 12, (5;7) – 7.
 15. (0;2) – 8, (0;1) – 12, (2;1) – 3,
(2;4) – 6, (1;3) – 5, (3;4) – 4,
(4;6) – 10, (3;5) – 4, (6;5) – 6,
(6;7) – 10, (5;7) – 6.

5. Контрольные вопросы

1. Что такое маршрутизатор? Задачи маршрутизации.
2. Статическая и динамическая маршрутизации. Отличия.
3. Алгоритмы маршрутизации. Сравнение.
4. Алгоритм Беллмана-Форда.

6. Список рекомендуемой литературы

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
Владимирский государственный университет имени А.Г. и Н.Г. Столетовых

Кафедра информационных систем
и программной инженерии

**ЛАБОРАТОРНАЯ РАБОТА №5
ИЗУЧЕНИЯ СРЕДСТВ МОНИТОРИНГА И
АНАЛИЗА СЕТЕВОГО ТРАФИКА.
СНИФФЕР WIRESHARK (ETHEREAL)**

Владимир 2015 г

1. Цель работы

- 1) Знать принципы анализа сетевого трафика.
- 2) Научиться использовать сетевой анализатор (сниффер Wireshark).
- 3) Научиться анализировать сетевой трафик на примере протоколов ARP, IP и ICMP.

2. Общие сведения

2.1. Анализ сетевого трафика и пакетов с использованием сниффера “ETHEREAL”

Sniffer (от англ. to sniff – нюхать) – это сетевой анализатор трафика, программа или программно-аппаратное устройство, предназначенное для перехвата и последующего анализа, либо только анализа сетевого трафика, предназначенного для других узлов.

Перехват трафика может осуществляться:

- обычным «прослушиванием» сетевого интерфейса (метод эффективен при использовании в сегменте концентраторов (хабов) вместо коммутаторов (свичей), в противном случае метод малоэффективен, поскольку на сниффер попадают лишь отдельные фреймы);
- подключением сниффера в разрыв канала;
- ответвлением (программным или аппаратным) трафика и направлением его копии на сниффер;
- через анализ побочных электромагнитных излучений и восстановление таким образом прослушиваемого трафика;
- через атаку на канальном (2-й) или сетевом (3-й) уровне, приводящую к перенаправлению трафика жертвы или всего трафика сегмента на сниффер с последующим возвращением трафика в надлежащий адрес.

В начале 1990-х широко применялся хакерами для захвата пользовательских логинов и паролей. Широкое распространение хабов позволяло захватывать трафик без больших усилий в больших сегментах сети.

Снифферы применяются как в благих, так и в деструктивных целях. Анализ прошедшего через сниффер трафика, позволяет:

- Отслеживать сетевую активность приложений.
- Отлаживать протоколы сетевых приложений.
- Локализовать неисправность или ошибку конфигурации.
- Обнаружить паразитный, вирусный и закольцованный трафик, наличие которого увеличивает нагрузку сетевого оборудования и каналов связи.
- Выявить в сети вредоносное и несанкционированное ПО, например, сетевые сканеры, флудеры, троянские программы, клиенты пиринговых сетей и другие.
- Перехватить любой незашифрованный (а порой и зашифрованный) пользовательский трафик с целью узнавания паролей и другой информации.

Постепенно из инструментов, предназначенных только для диагностики, снифферы постепенно превратились в средства для исследований и обучения. Например, они постоянно используются для изучения динамики и взаимодействий в сетях. В частности, они позволяют легко и наглядно изучать тонкости сетевых протоколов. Наблюдая за данными, которые посылает протокол, вы можете глубже понять его функционирование на практике, а заодно увидеть, когда некоторая конкретная реализация работает не в соответствии со спецификацией.

На сегодняшний момент существует достаточно большое количество хороших реализаций снифферов. Вот некоторые из них:

1. Tcpdump (<http://www.tcpdump.org/>) – консольный вариант сниффера. Портирован почти подо все наиболее распространенные ОС;
2. Wireshark (<http://www.wireshark.org/>) до недавнего момента был известен под названием Ethreal;
3. WinDump <http://www.winpcap.org/windump>;
4. и др.

2.2. Сниффер WireShark

Программа Wireshark является одной из самых удобных реализаций снифферов. Портирована на большое количество платформ. Распространяется абсолютно бесплатно. В дальнейшем на лабораторных работах мы будем постоянно использовать данный сниффер для изучения и анализа сетевых протоколов.

Но для начала рассмотрим базовый принцип работы sniffера как раз на примере Wireshark.

2.2.1. Базовый принцип работы sniffеров

Давайте рассмотрим с вами рисунок 5.1. На нем схематично изображена структура сетевой подсистемы ОС. Вся базовая инфраструктура реализована в виде драйверов и работает в режиме ядра. Пользовательские процессы и реализации прикладных протоколов, в частности интерфейс sniffера работают в пользовательском режиме.

На рисунке 5.1 отображены два пользовательских процесса («сетевой процесс 1» и «сетевой процесс 2»).

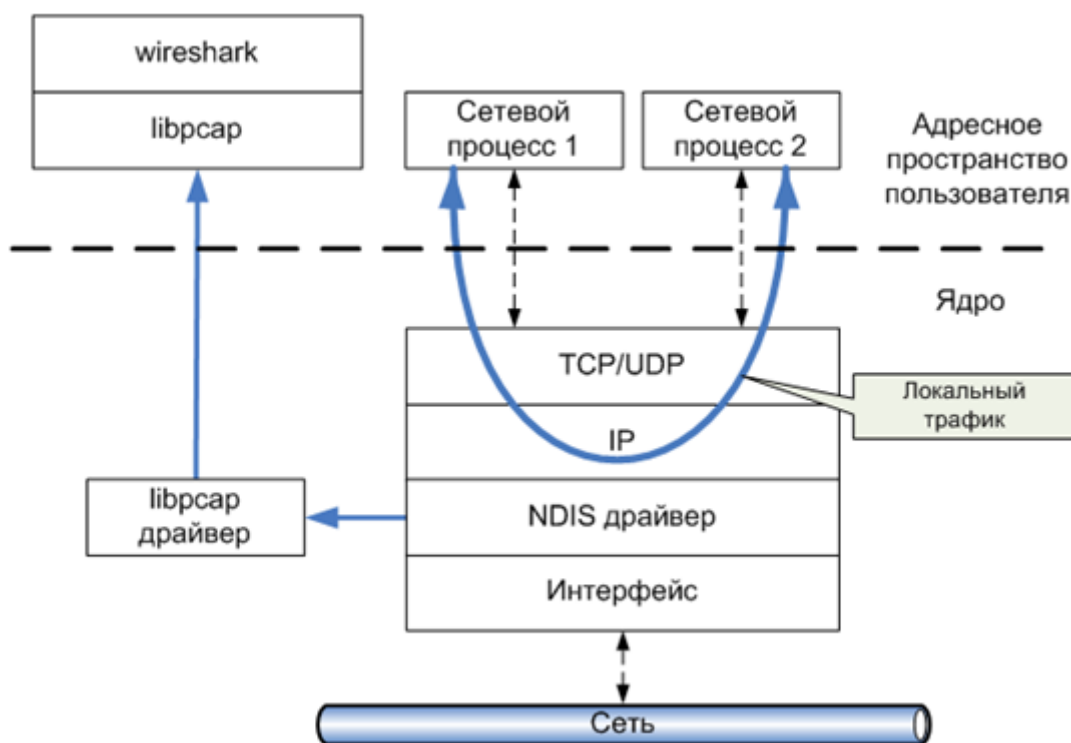


Рисунок 5.1. Принцип «захвата» sniffером сетевого трафика.

Основными компонентами sniffера являются: драйвер для захвата пакетов (libpcap драйвер), интерфейсная библиотека (libpcap) и интерфейс пользователя (Wireshark). Библиотека libpcap (реализация под ОС Windows носит название WinPcap - <http://www.winpcap.org>) – универсальная сетевая библиотека, самостоятельно реализующая большое количество сетевых протоколов и работающая непосредственно с NDIS (Network Driver Interface Specification) драйверами сетевых устройств. На базе данной библиотеки

реализовано большое количество сетевых программ, в частности сниффер Wireshark.

Сниффер использует библиотеку в режиме «захвата» пакетов, т.е. может получать копию ВСЕХ данных проходящих через драйвер сетевого интерфейса. Изменения в сами данные не вносятся!

Основной нюанс использования сниффера заключается в том, что он не позволяет производить анализ локального трафика, т.к. он не проходит через драйвер сетевого устройства (см. рис. 5.1.). Если вы захотите проанализировать сниффером трафик между двумя сетевыми процессами на локальной машине (например, ftp-сервер и ftp-клиент), то вас ждет разочарование. Однако при использовании виртуальных машин, сниффер будет работать без проблем, т.к. виртуальные машины эмулируют реальную среду и сетевые адаптеры, поэтому трафик идет через драйвера, как и в нормальной ситуации при взаимодействии с другими физическими сетевыми машинами.

Также к недостаткам большинства снифферов стоит отнести и тот факт, что они не могут указать, какое именно приложение генерирует или получает его, позволяя анализировать трафик, проходящий через сетевой интерфейс. Это объясняется тем, что информация об этом хранится на сетевом (например, IP) уровне сетевого стека, а большинство снифферов использует собственную реализацию стека протоколов (например, библиотеку WinPcap), которая (как уже было показано) работает непосредственно с драйверами устройств.

Стоит также отметить, что снифферы вносят дополнительную нагрузку на процессор, т.к. могут обрабатывать достаточно объемный сетевой трафик, в особенности для высокоскоростных соединений (Fast Ethernet, Gigabit Ethernet и др.).

2.2.2. Использование программы *WireShark*

Данный сниффер (в дальнейшем в примерах и описании подразумевается версия 0.99.6a (SVN Rev 22276)) позволяет в режиме реального времени захватывать пакеты из сети и анализировать их структуру. Также можно анализировать структуру пакетов из файла, содержащего трафик, полученный, например, программой «tcpdump» (unix/linux).

На рисунке 5.2. изображено основное окно программы Wireshark. В стандартном режиме окно сниффера делится на 3 фрейма (панели): список захваченных пакетов, «анализатор» протоколов и исходные данные пакетов. Размер каждого фрейма можно менять по своему усмотрению.

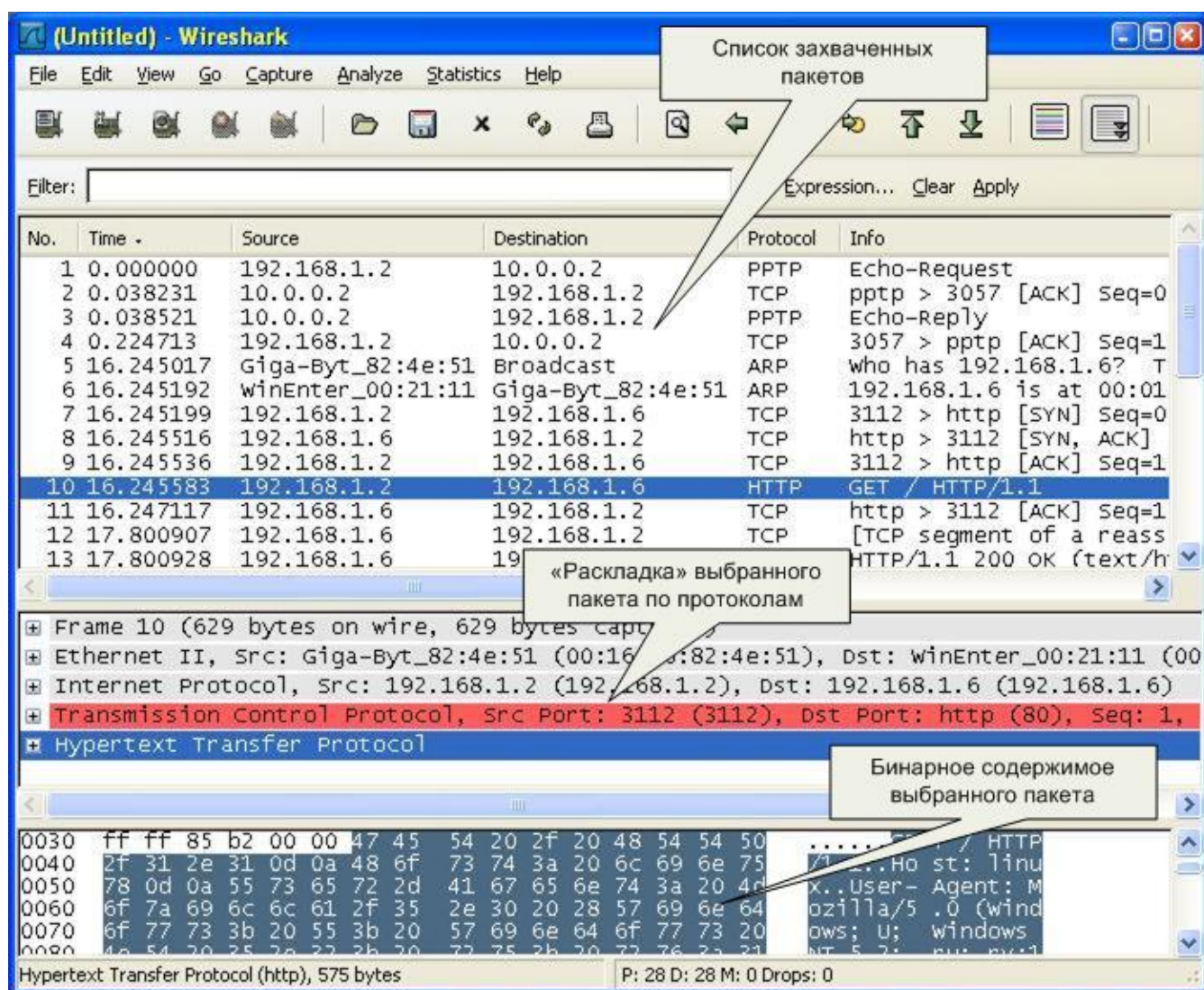


Рисунок 5.2. Основное окно sniffера Wireshark

Верхняя панель содержит список пакетов, захваченных из сети. Список можно отсортировать по любому полю (в прямом или обратном порядке). Для этого нажать на заголовок соответствующего поля.

Каждая строка содержит следующие поля (по умолчанию):

- порядковый номер пакета (No.);
- время поступления пакета (Time);
- источник пакета (Source);
- пункт назначения (Destination);
- протокол (Protocol);
- информационное поле (Info).

Список отображаемых полей настраивается в Edit/Perferencis/Columns. Для того чтобы изменения возымели эффект, необходимо перезапустить программу, предварительно нажав на кнопку Save.

При нажатии правой кнопки мыши на том или ином пакете, появится контекстное меню. Нажатием на среднюю кнопку мыши можно пометить группу интересующих нас пакетов.

Средняя панель содержит «дерево протоколов» для выбранного в верхнем окне пакета. В этой панели в иерархическом виде отображается вложенность протоколов в соответствии с моделью взаимодействия открытых систем OSI. Нажав на правую кнопку мыши, вызывается контекстное меню. При «раскрытии» каждого из протокола нажатием на значок «+» слева, выводятся поля данных соответствующих протоколов.

Нижняя панель содержит шестнадцатеричное представление выбранного пакета. При выборе того или иного поля в средней панели автоматически будет подсвечиваться соответствующий участок 16-ого представления.

2.3. *Захват пакетов*

Для начала захвата пакетов необходимо задать параметры захвата. В частности, указать сетевой интерфейс, с которого и будет осуществляться захват. Это действие доступно через меню как «Capture→Options» или комбинации клавиш CTRL+K (см. рис. 5.3.). Интерфейс, задаваемый в поле «*Interface:*» можно выбрать из соответствующего поля. На рисунке 5.3. показано, что доступны 3 интерфейса: физический сетевой адаптер («Marvel...»), и интерфейсы для виртуальных каналов, установленного VPN-соединения («WAN (PPP/SLIP)...»). В большинстве случаев подходит выбор интерфейса сетевого адаптера.

В качестве дополнительных параметров захвата можно указать следующие:

- «*Capture Filter*» – фильтр захвата. По нажатию на соответствующую кнопку можно применить тот или иной фильтр отбора (из ранее сохраненных). Если таковых не имеется, его можно указать явно в строке редактирования.
- «*Update list of packets in real time*» – обновление списка захваченных пакетов в режиме реального времени.
- «*Stop Capture*» – набор параметров, позволяющих задать то или иное значение при достижении, которого процесс захвата пакетов прекратится.
- «*Name Resolution*» – набор параметров разрешения имен позволяет определить какие из способов разрешения имен должны использоваться.

Для начала мониторинга сетевой активности нужно нажать на кнопку «Start». После выбора интерфейса в дальнейшем можно начинать и останавливать захват пакетов через соответствующие команды в меню «Capture».

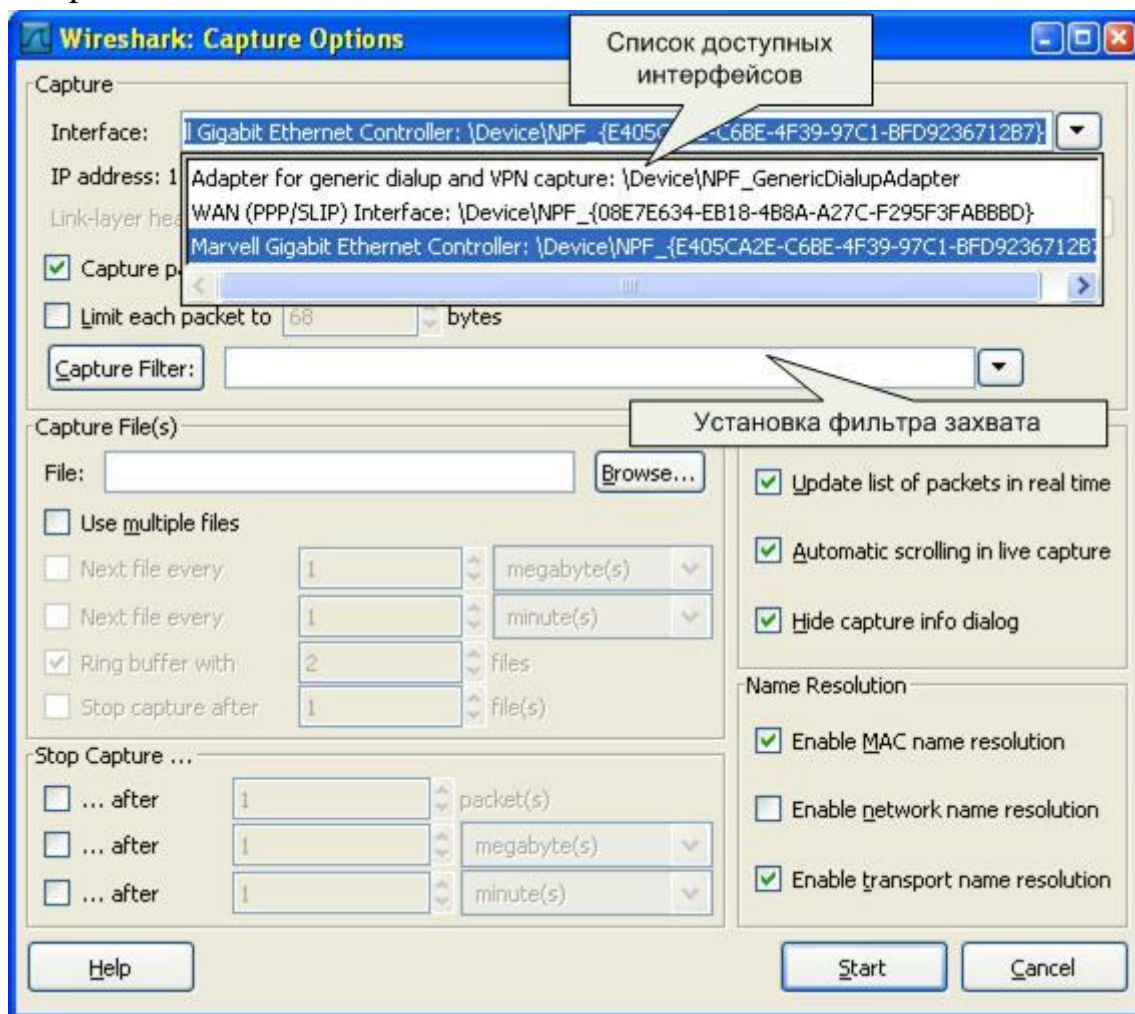


Рисунок 5.3. Выбор интерфейса и параметров захвата пакетов

2.4. Фильтрация пакетов

Если запустить сниффер без дополнительных настроек, он будет «захватывать» все пакеты, проходящие через сетевые интерфейсы (см. рисунок 5.1). Для общего ознакомления с процессами, происходящими в сети, полезно пронаблюдать активность сетевых протоколов в реальных условиях работы системы в сети и все разнообразие протоколов, запросов, ответов и др. событий.

При целенаправленном использовании сниффера очень часто необходимо выборочно отображать или захватывать пакеты по некоторым заданным критериям. Для этих целей служат фильтры отображения и захвата, соответственно.

2.4.1. Типы фильтрации трафика

Существует два варианта фильтрации пакетов: на этапе захвата и на этапе отображения пользователю. В первом случае эффективность работы сниффера и потребляемые им системные ресурсы значительно ниже, нежели во втором случае. Это объясняется тем, что при достаточно интенсивном сетевом трафике и продолжительном времени захвата все пакеты должны быть захвачены и сохранены либо в память, либо на дисковое устройство. Самые простые подсчеты могут показать, что даже для 100-мегабитной сети системных ресурсов хватит на непродолжительное время. Фильтрация захвата уже на момент получения пакета гораздо эффективнее, однако, в таком случае, она должна быть реализована на уровне самых драйверов захвата. Данный факт, естественно, усложняет реализацию сниффера. Wireshark поддерживает оба варианта фильтрации.

2.4.2. Фильтры отображения

Фильтры отображения представляют собой достаточно мощное средство отображения трафика. Фильтры задаются в строке, располагающейся вверху основного экрана («Filter:»). Простейший фильтр отображения позволяет отобрать пакеты по тому или иному протоколу. Для этого требуется указать в строке название протокола (например, HTTP) и нажать кнопку «Apply». После этого в верхнем окне останутся пакеты, принадлежащие этому протоколу. Кнопкой «Reset» действие фильтра отменяется.

Для работы с фильтрами можно вызвать окно «Analyze/Display Filters», сохранять созданные выражения под определенными именами для последующего использования и т.д.

С помощью логических операций (синтаксис языка Си) можно составлять логические выражения. Логическая истина - 1, ложь - 0.

Список поддерживаемых логических операций:

eq	==	равенство
ne	!=	не равно
gt	>	больше чем
Lt	<	меньше чем
ge	>=	больше или равно
Le	<=	меньше или равно

Например: tcp.port == 80 (см. рисунок 5.4.).

Filter: tcp.port==80 Expression... Clear Apply					
No.	Time	Source	Destination	Protocol	Info
5	8.689788	192.168.1.2	192.168.1.1	TCP	4810 > http [SYN] Seq=0
6	8.691675	192.168.1.1	192.168.1.2	TCP	http > 4810 [SYN, ACK] S
7	8.691717	192.168.1.2	192.168.1.1	TCP	4810 > http [ACK] Seq=1
8	8.691877	192.168.1.2	192.168.1.1	HTTP	GET /html/js/alphaindex.
9	8.699198	192.168.1.1	192.168.1.2	TCP	http > 4810 [ACK] Seq=1
10	8.699230	192.168.1.1	192.168.1.2	TCP	[TCP segment of a reasse
11	8.699246	192.168.1.1	192.168.1.2	HTTP	HTTP/1.1 404 Not Found (
12	8.699266	192.168.1.2	192.168.1.1	TCP	4810 > http [ACK] Seq=46
13	8.699556	192.168.1.2	192.168.1.1	TCP	4810 > http [FIN, ACK] S
14	8.701682	192.168.1.1	192.168.1.2	TCP	http > 4810 [ACK] Seq=45

Рисунок 5.4. Пример задания простого фильтра отображения

Мастер построения фильтров отображения доступен через кнопку «Expression...» (см. рисунок 5.5.).

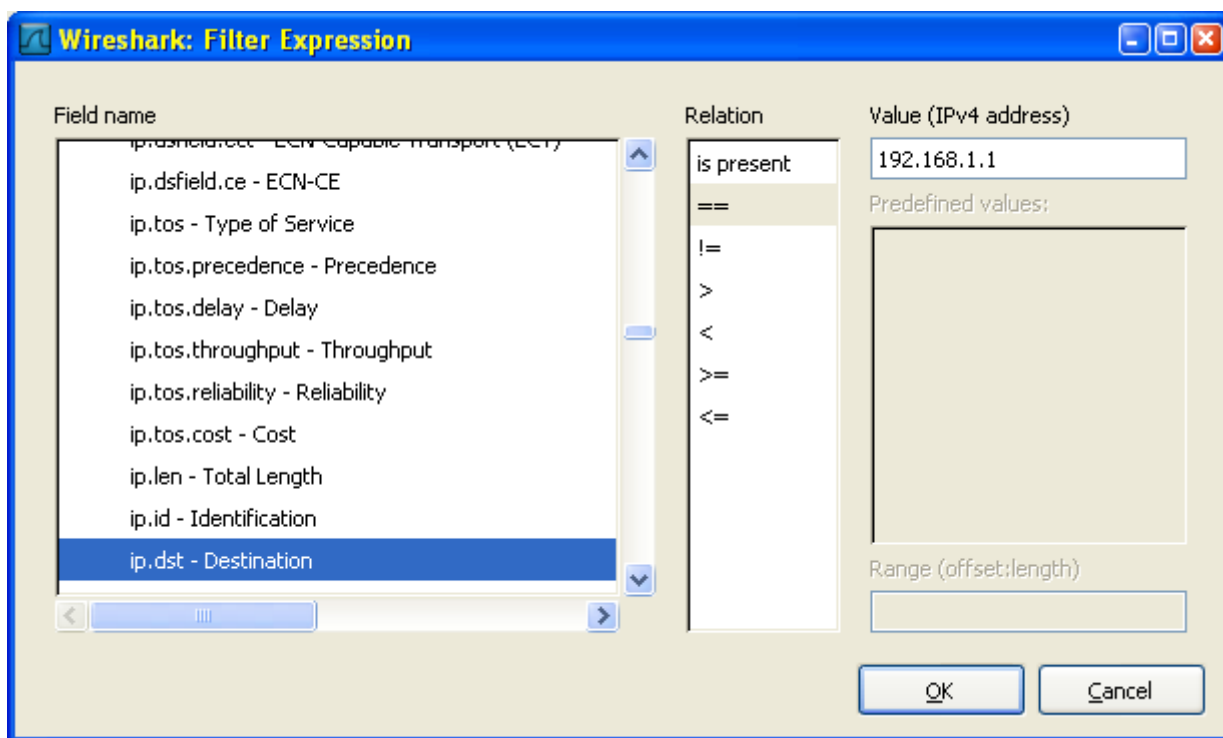


Рисунок 5.5. Построение фильтров отображения

2.4.3. Фильтры захвата

С помощью данных фильтров можно захватывать из сети только те пакеты, которые подходят под критерий отбора. Если не задано никакого фильтра (по умолчанию), то будут захватываться все пакеты. Синтаксис фильтров захвата несколько отличается от синтаксиса фильтров отображения. Выражение состоит из одного или более примитивов, разделенных пробельными символами. На рисунке 5.6. приведен пример

фильтра для захвата пакетов, адресованных на 80-й порт (http) узла с ip-адресом 10.197.0.11.

Существует три различных типа примитивов: *type*, *dir*, *proto*.

Спецификатор *type* определяет тип параметра.

Возможные параметры: **host**; **net**; **port**.

Например:

- host linux
- net 192.168.128
- port 80

Если не указано никакого типа предполагается, что это параметр **host**.

Спецификатор *dir* определяют направление передачи. Возможные направления: **src**; **dst**; **src or dst**; **src and dst**.

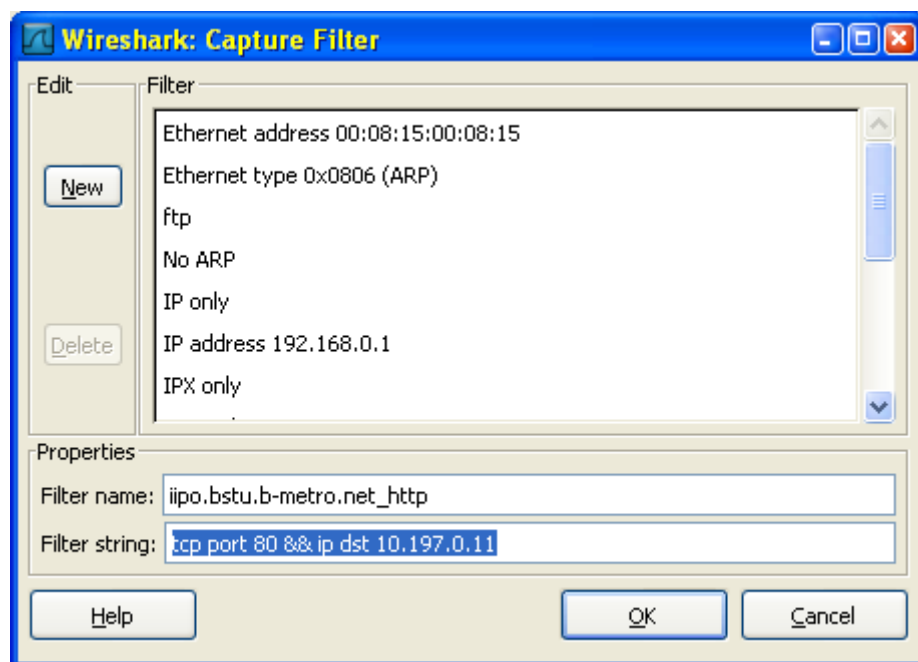


Рисунок 5.6. Пример фильтра захвата

Например:

- src linux
- dst net 192.168.128
- src or dst port http

Если не определено направление, то предполагается направление «**src or dst**». Для протоколов типа point-to-point используются спецификаторы inbound и outbound.

Спецификатор *proto* определяют тип протокола, которому принадлежит пакет.

Возможные протоколы: **ether; fddi; tr; ip/ipv6; arp/rarp; decent; tcp; udp.**

Например:

- ether src linux
- arp net 192.168.128
- tcp port 80

Если протокол не определен, то будут захватываться пакеты всех протоколов. Для «src linux» - это «(ip or arp or rarp) src linux»; для «net ctam» - «(ip or arp or rarp) net ctam»; для «port 53» - «(tcp or udp) port 53».

Также существует несколько специальных спецификаторов, которые не попадают в описанные выше случаи:

- *gateway;*
- *broadcast;*
- *less;*
- *greater;*
- *арифметические выражения.*

Сложные фильтры захвата строятся с использованием логических выражений.

Список операций:

not	!	отрицание
and	&&	конкатенация (логическое И)
or		альтернатива (логическое ИЛИ)

2.4.4. Примеры фильтров захвата

Ниже рассмотрены некоторые примеры построения фильтров захвата.

- Захват всех пакетов на сетевом интерфейсе хоста 192.168.1.2:
host 192.168.1.2
- Захват трафика между хостом host1 И хостами host2 ИЛИ host3:
host host1 and (host2 or host3)
- Захват всех IP-пакетов между хостом host1 и каждым хостом за исключением hostX:
ip host host1 and not hostX
- Захват пакетов ни сгенерированных, ни адресованных локальными хостами:
ip and not net localnet

- Захват IP-пакетов размером больше чем 576 байт, проходящих через шлюз snup:
`gateway snup and ip[2:2] > 576`
- Захват всех ICMP пакетов, за исключением пакетов ping:
`icmp[0] != 8 and icmp[0] != 0`

2.4.5. Статистическая обработка сетевого трафика

Сниффер Wireshark позволяет выполнять различную статистическую обработку полученных данных. Все доступные операции находятся в меню «Statistics».

Общая статистика — количество полученных/переданных пакетов, средняя скорость передачи и т.д. доступны через пункт «Statistics->Summary».

Получить информацию по статистике обработанных протоколов в полученных пакетах можно через пункт «Statistics->Protocol Hierarchy».

Статистику по типу ip-пакетов, их размеру и порту назначения можно получить, выбрав подпункты меню «IP-address...», «Packet length» и «Port type» соответственно.

Одной из наиболее интересных возможностей является генерация диаграммы взаимодействия между узлами, которая доступна, выбрав пункт меню «Flow Graph...». В результате можно наблюдать в достаточно наглядной форме процесс взаимодействия на уровне протоколов. На рисунке 5.7. приведена диаграмма взаимодействия при получении узлом win2003 статической web-страницы с сервера <http://linux>.

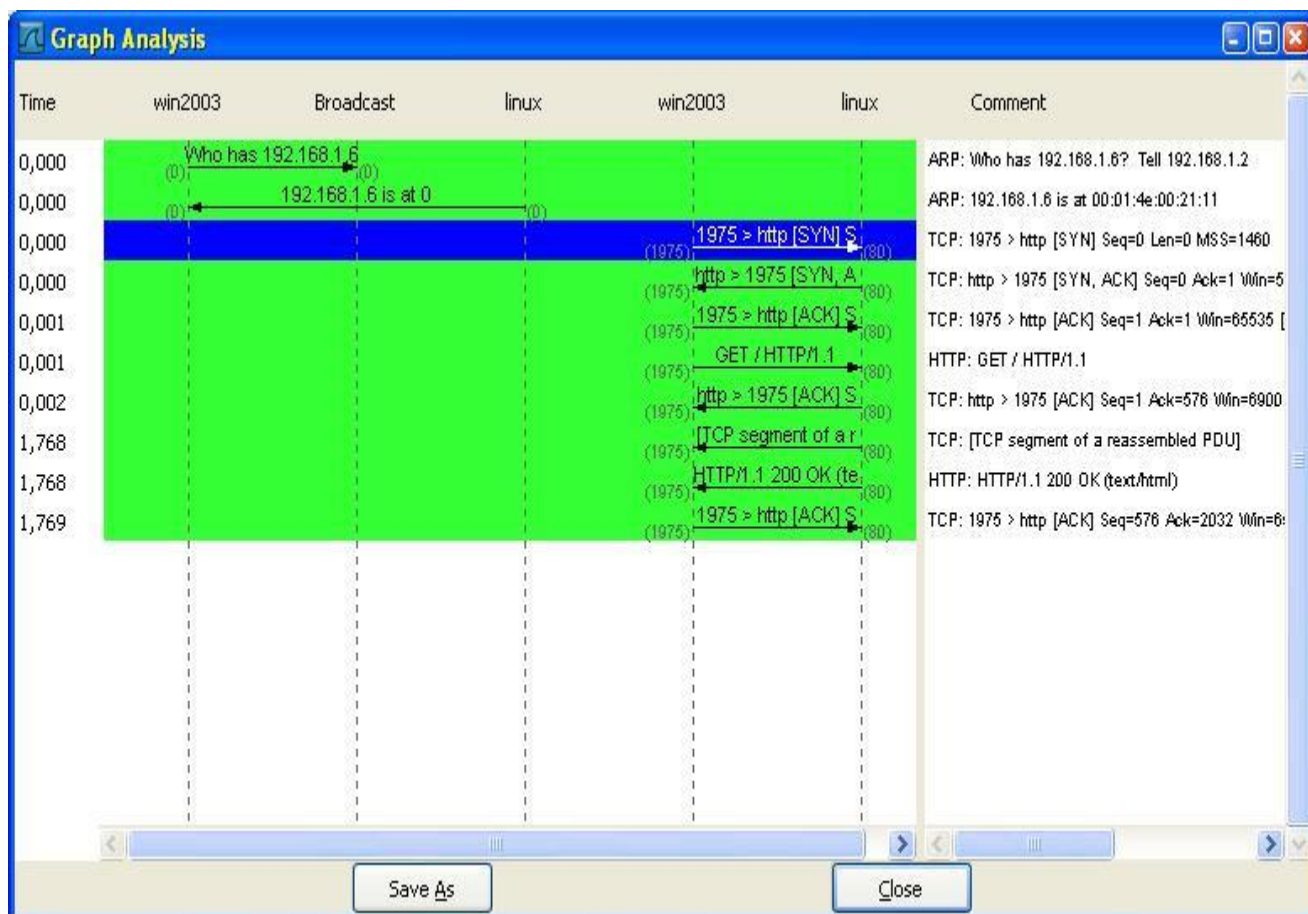


Рисунок 5.7. Диаграмма взаимодействия

Программы-снифферы – это незаменимый инструмент для изучения того, что происходит в сети. Если знать, что в действительности посылается или принимается «по проводам», то трудные, на первый взгляд, ошибки удастся легко найти и исправить. Снiffer представляет собой также важный инструмент для исследований динамики сети.

2.5. Пример исследование протокола с использованием сниффера

В качестве примера исследования некоторого протокола с использованием сниффера рассмотрим протокол ARP.

ARP (англ. Address Resolution Protocol — протокол разрешения адресов) — сетевой протокол, предназначенный для преобразования IP-адресов (адресов сетевого уровня) в MAC-адреса (адреса канального уровня) в сетях TCP/IP. Он определён в **RFC 826**.

Данный протокол очень распространенный и чрезвычайно важный. Каждый узел сети имеет как минимум два адреса, физический адрес и логический адрес. В сети Ethernet для идентификации источника и

получателя информации используются оба адреса. Информация, пересылаемая от одного компьютера к другому по сети, содержит в себе физический адрес отправителя, IP-адрес отправителя, физический адрес получателя и IP-адрес получателя. ARP-протокол обеспечивает связь между этими двумя адресами. Существует четыре типа ARP-сообщений: ARP-запрос (ARP request), ARP-ответ (ARP reply), RARP-запрос (RARP-request) и RARP-ответ (RARP-reply). Локальный хост при помощи ARP-запроса запрашивает физический адрес хоста-получателя. Ответ (физический адрес хоста-получателя) приходит в виде ARP-ответа. Хост-получатель вместе с ответом шлет также RARP-запрос, адресованный отправителю, для того чтобы проверить его IP адрес. После проверки IP-адреса отправителя, начинается передача пакетов данных.

Перед тем как создать подключение к какому-либо устройству в сети, IP-протокол проверяет свой ARP-кеш, чтобы выяснить, не зарегистрирована ли в нём уже нужная для подключения информация о хосте-получателе. Если такой записи в ARP-кеше нет, то выполняется широковещательный ARP-запрос. Этот запрос для устройств в сети имеет следующий смысл: «Кто-нибудь знает физический адрес устройства, обладающего следующим IP-адресом?» Когда получатель примет этот пакет, то должен будет ответить: «Да, это мой IP-адрес. Мой физический адрес следующий: ...» После этого отправитель обновит свой ARP-кеш и будет способен передать информацию получателю.

RARP (англ. Reverse Address Resolution Protocol – обратный протокол преобразования адресов) – выполняет обратное отображение адресов, то есть преобразует аппаратный адрес в IP-адрес.

Протокол применяется во время загрузки узла (например, компьютера), когда он посылает групповое сообщение-запрос со своим физическим адресом. Сервер принимает это сообщение и просматривает свои таблицы (либо перенаправляет запрос куда-либо ещё) в поисках соответствующего физического IP-адреса. После обнаружения найденный адрес отсылается обратно на запросивший его узел. Другие станции также могут «слышать» этот диалог и локально сохранить эту информацию в своих ARP-таблицах.

RARP позволяет разделять IP-адреса между не часто используемыми хост-узлами. После использования каким-либо узлом IP-адреса он может быть освобождён и выдан другому узлу. RARP является дополнением к ARP, и описан в **RFC 903**.

Для просмотра ARP-кеша можно использовать одноименную утилиту `arp` с параметром «-а».

Например:

```
D:\>arp -a
```

```
Interface: 192.168.1.2 --- 0x10003
```

Internet Address	Physical Address	Type
192.168.1.1	00-15-e9-b6-67-4f	
dynamic		
192.168.1.6	00-01-4e-00-21-11	
dynamic		

Из данного результата команды `arp` видно, что в кеше на данный момент находится две записи и видны соответственно `ip`-адреса машин и `MAC`-адреса их сетевых адаптеров.

Записи в `ARP`-кеше могут быть статическими и динамическими. Пример, данный выше, описывает динамическую запись кеша. Хост-отправитель автоматически послал запрос получателю, не уведомляя при этом пользователя. Записи в `ARP`-кеш можно добавлять вручную, создавая статические (`static`) записи кеша. Это можно сделать при помощи команды:

```
arp -s <IP адрес> <MAC адрес>
```

Также можно удалять записи из `ARP`-кеша. Это осуществляется путем следующего вызова:

```
arp -d <IP адрес>
```

После того как `IP`-адрес прошёл процедуру разрешения адреса, он остается в кеше в течение двух минут. Если в течение этих двух минут произошла повторная передача данных по этому адресу, то время хранения записи в кеше продлевается ещё на 2 минуты. Эта процедура может повторяться до тех пор, пока запись в кеше просуществует до 10 минут. После этого запись будет удалена из кеша и будет отправлен повторный `ARP`-запрос.

`ARP` изначально был разработан не только для `IP` протокола, но и в настоящее время в основном используется для сопоставления `IP`- и `MAC`-адресов.

Посмотрим же на практике, как работает протокол `ARP/RARP`. Для этого воспользуемся сниффером для захвата сетевого трафика.

Рассмотрим пример работы протокола `ARP` при обращении к машине с адресом 192.168.1.5, выполнив запрос с машины с адресом 192.168.1.2. Для успешного эксперимента предварительно очистим `arp`-кеш командой

```
arp -d 192.168.1.5
```

Для фильтрации ARP/RARP трафика воспользуемся фильтром захвата. В нашем случае это будет простой фильтр

```
arp or rarp
```

Далее запустим захват трафика командой «Start» и выполним обращение к заданной машине, например, «пропинговав» ее:

```
D:\>ping 192.168.1.5
Pinging 192.168.1.5 with 32 bytes of data:
Reply from 192.168.1.5: bytes=32 time<1ms TTL=64
Reply from 192.168.1.5: bytes=32 time<1ms TTL=64
Reply from 192.168.1.5: bytes=32 time<1ms TTL=64
Reply from 192.168.1.5: bytes=32 time<1ms TTL=64
Ping statistics for 192.168.1.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0%
loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Так как на момент начала работы утилиты ping в arp-кеше не было информации о MAC-адресе соответствующего узла, то первоначально система должна выполнить определение это самого MAC-адреса, сгенерировав ARP-запрос и отослав его в сеть широковещательным пакетом. После чего она будет ожидать ответа от заданного узла.

Посмотрим же, что мы получим на практике. После остановки sniffера мы должны увидеть результат схожий с тем, что отображен на рисунке 5.8. В нашем случае мы видим два захваченных пакета: ARP-запрос и ARP-ответ.

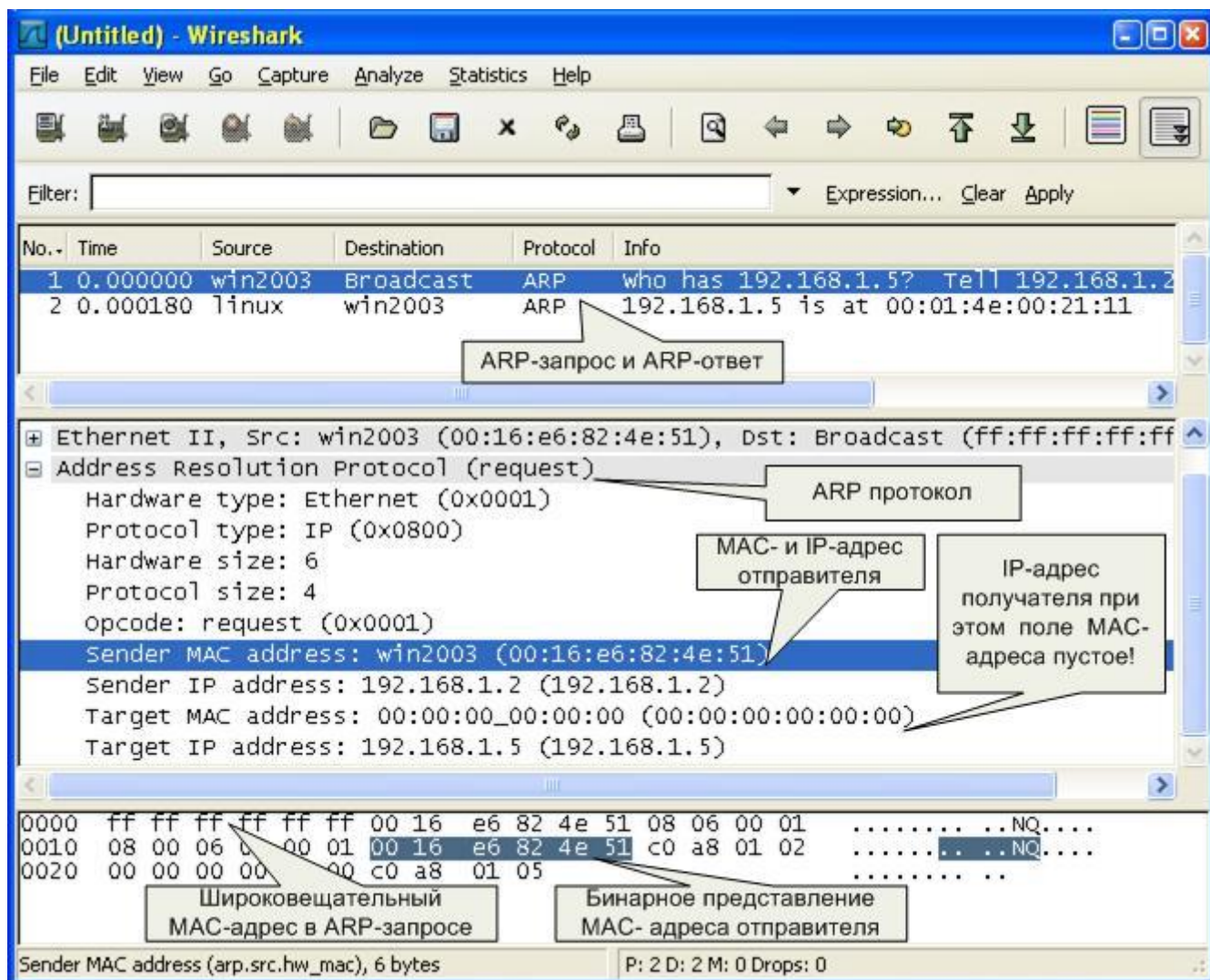


Рисунок 5.8. Анализ ARP-запроса.

Проанализируем полученные пакеты. Сначала рассмотрим ARP-запрос (пакет №1). Выделив пакет курсором, мы получаем его раскладку по протоколам (Ethernet+ARP) в среднем окне. Wireshark очень наглядно «раскладывает» заголовок протокола по полям.

Мы можем видеть, что в пакете указаны MAC- и IP-адреса отправителя («Sender MAC address» и «Sender IP address» соответственно). Это параметры машины, с которой выполняется запрос. В данном случае, запрос направлен на получение («Opcode: request» – запрос) MAC-адреса машины, у которой IP-адрес («Protocol type: IP») 192.168.1.5 («Target IP address»). При этом поле «Target MAC address» обнулено. Так как получатель ARP-запроса на момент запроса не известен, Ethernet-пакет отправляется всем машинам в данном локальном сегменте, о чем сигнализирует MAC адрес Ethernet-пакета «ff:ff:ff:ff:ff:ff».

Примечание. Обратите внимание, что пакет представляет собой бинарную последовательность, и сниффер выполняет большую работу по преобразованию полей из бинарного представления в удобочитаемый вариант.

Все работающие машины в сети получают пакет с ARP-запросом, анализируют его, а ответ отправляет только та машина, чей IP-адрес соответствует IP-адресу в запросе. Таким образом, второй полученный пакет является ARP-ответом (см. рисунок 5.9.). Это следует из параметра поля «Opcode: reply». Обратите внимание, что данный пакет был отправлен именно той машиной, чей MAC-адрес нас и интересовал («Sender IP address: 192.168.1.5»). При этом поле «Sender MAC address» заполнено значением «00:01:4E:00:21:11».

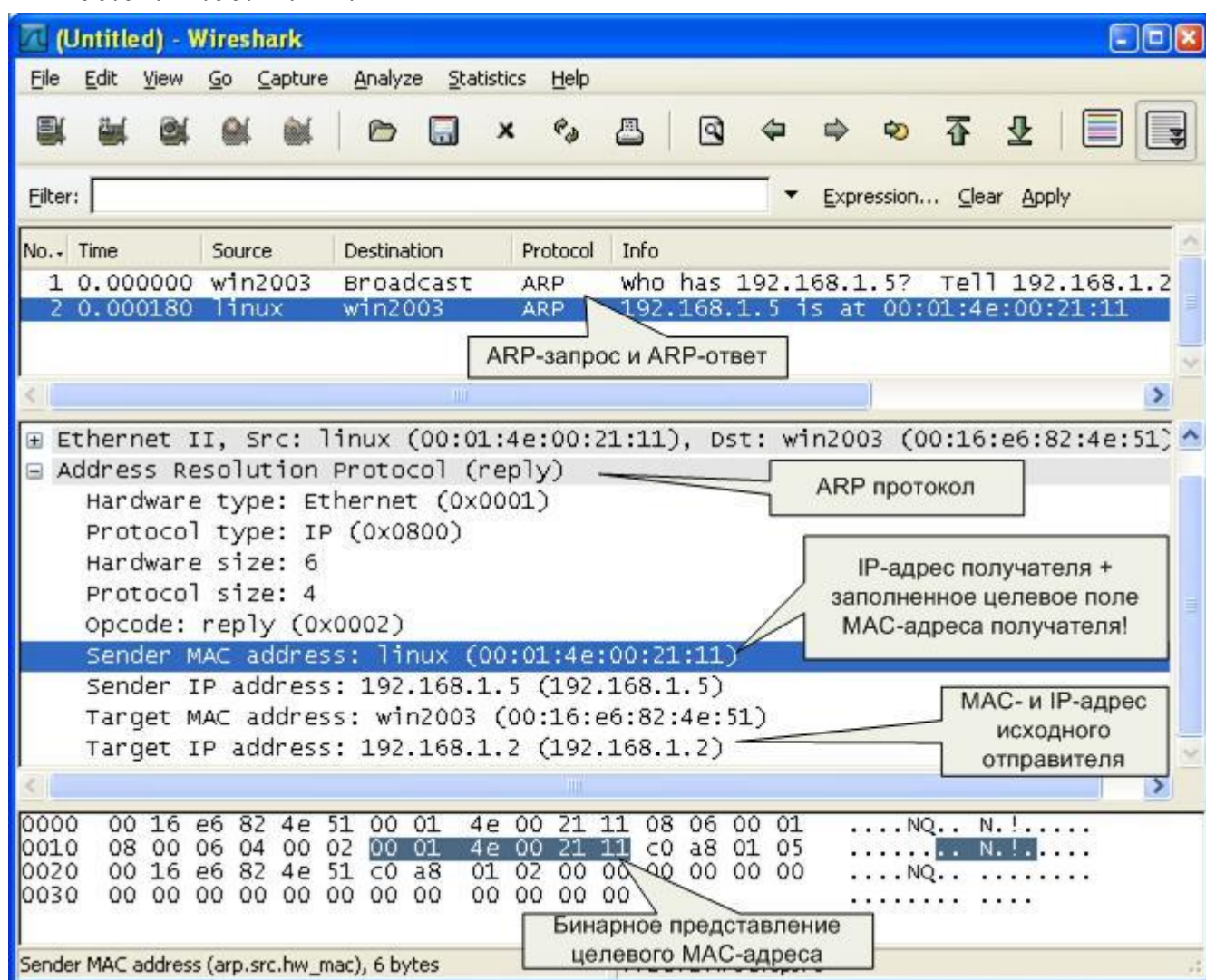


Рисунок 5.9. Анализ ARP-ответа.

Примечание. Обратите внимание на поле «Info» в списке захваченных пакетов. Сниффер и тут упрощает анализ сетевого трафика, подсказывая назначение пакетов.

Теперь мы можем повторно просмотреть ARP-кеш и сверить данные в нем с данными, которые мы узнали из анализа пакетов ARP-запрос/ответа:

```
D:\>arp -a
Interface: 192.168.1.2 --- 0x10003
    Internet Address      Physical Address      Type
    192.168.1.5           00-01-4e-00-21-11
dynamic
```

Стоит отметить, что в реальных условиях в локальной сети с большим количеством машин arp/rarp трафик бывает гораздо более интенсивным.

3. Задания для выполнения

1. Изучить интерфейс программы Wireshark
https://www.wireshark.org/docs/wsug_html_chunked/
2. Захватить 100 произвольных пакетов. Определить статистические данные:
 - процентное соотношение трафика разных протоколов в сети;
 - среднюю скорость кадров/сек;
 - среднюю скорость байт/сек;
 - минимальный, максимальный и средний размеры пакета;
 - степень использования полосы пропускания канала (загрузку сети).
3. Зафиксировать 20 IP-пакетов. Определить статистические данные:
 - процентное соотношение трафика разных протоколов стека tcp/ip в сети;
 - средний, минимальный, максимальный размеры пакета.
4. Выполнить анализ ARP-протокола по примеру из методических указаний.
5. На примере любого IP-пакета указать структуры протоколов Ethernet и IP, Отметить поля заголовков и описать их.
6. Проанализировать и описать принцип работы утилиты *ping*. При этом описать все протоколы, используемые утилитой. Описать все поля протоколов. Составить диаграмму взаимодействия машин при работе утилиты *ping*.
Примечание. Данная утилита использует протокол ICMP (RFC 792 и RFC 960).

4. Контрольные вопросы

- 1) Каковы основные цели мониторинга сетевого трафика?
- 2) Чем отличается мониторинг трафика от фильтрации?
- 3) Каково назначение класса программ-снифферов?
- 4) Какие основные функции выполняют снифферы?

- 5) Зачем используются фильтры отображения и фильтры захвата сниффера Wireshark? В чем их отличие?
- 6) Какие базовые функции статистической обработки захваченных пакетов имеет сниффер Wireshark?
- 7) Какие задачи рассчитан решать протокол ARP?

5. Список рекомендуемой литературы

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
Владимирский государственный университет имени А.Г. и Н.Г. Столетовых

Кафедра информационных систем
и программной инженерии

ЛАБОРАТОРНАЯ РАБОТА №6 ЗАПУСК ВЕБ-СЕРВИСОВ В ЛОКАЛЬНЫХ СЕТЯХ

Владимир 2015 г

1. Цель работы

Получить практические навыки по развертыванию веб-сервера под управлением Apache (версия 2.x), включая установку httpd, основные настройки и конфигурирование виртуальных хостов. Получить навыки по работе и настроек программ MySQL и PHP. Получить навыки работы с запуском сайтов по локальной сети с помощью Apache.

2. План работы

- ✓ Скачать Apache HTTP Server 2.2.18 , MySQL 5.5.12 , PHP 5.2.16;
- ✓ Установить веб-сервер Apache, MySQL, PHP и выполните их настройку;
- ✓ Выполнить установку и настройку phpMyAdmin;
- ✓ Создать простой сайт (Hello World! или предложить свой);
- ✓ Выполнить запуск сайта в локальной сети;
- ✓ Подготовить и защитить отчет.

3. Указания к работе

На диске C создайте папку с именем **server** и поместите туда три другие папки (apache, php, MySQL), в которых мы сохраним и распакуем (если скачали архивы) одноименные дистрибутивы.

3.1. Установка и настройка локального сервера Apache

- 1) Запускаем инсталлятор Apache.
- 2) Жмем **Next**.
- 3) Принимаем лицензионное соглашение, жмем **Next** (рисунок 6.1.).

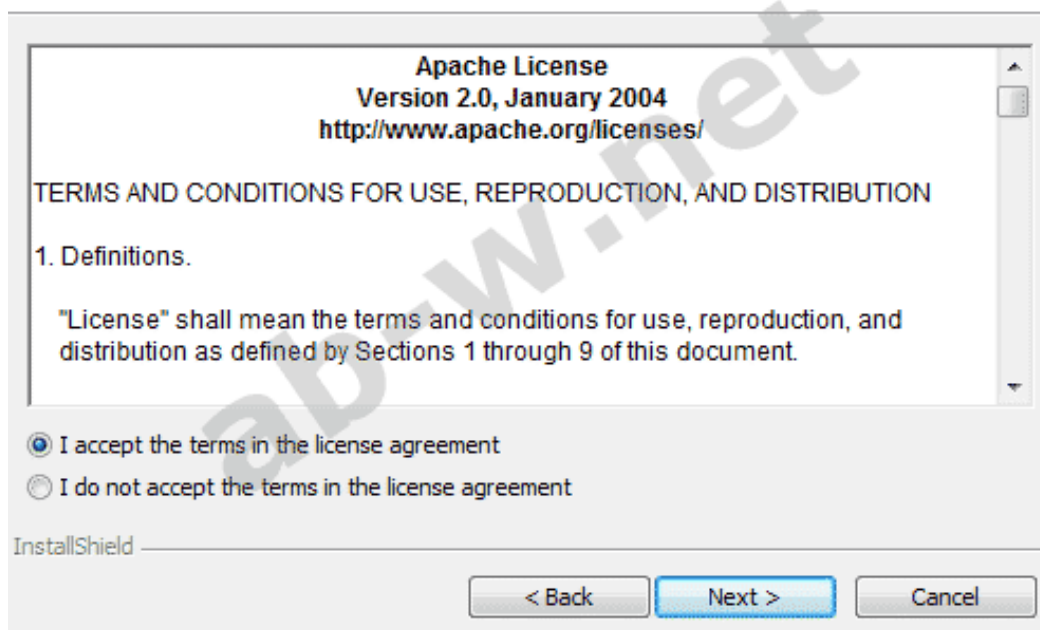


Рисунок 6.1.

4) **Next** (рисунок 6.2.).

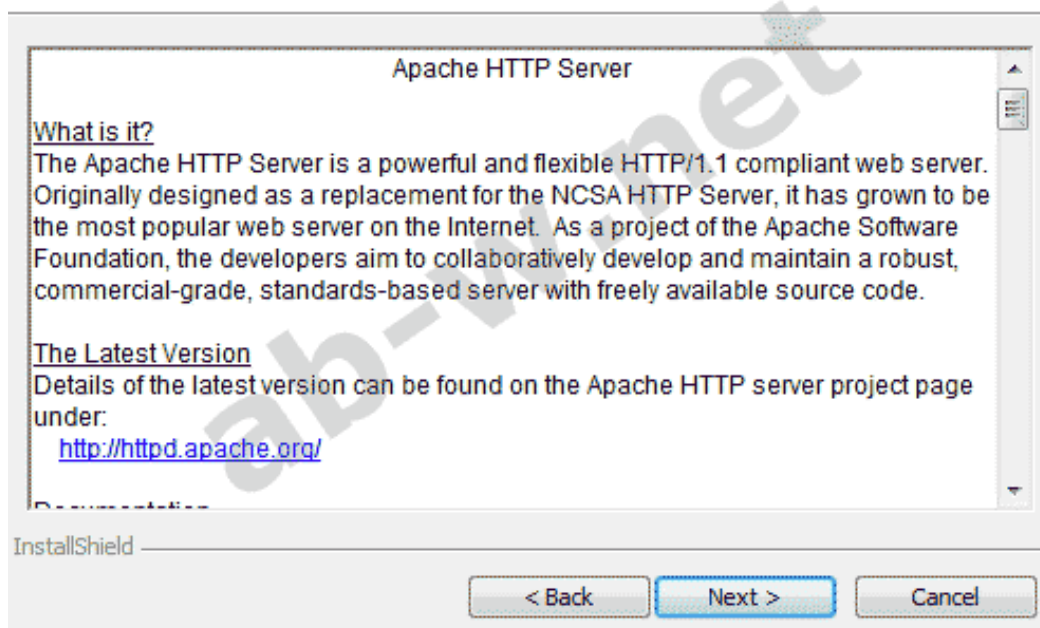


Рисунок 6.2.

5) Вводим информацию о нашем сервере. Так как сервер у нас локальный, прописываем стандартный для этого случая набор данных. В двух первых полях прописываем **localhost**, а в третьем — **admin@localhost** (рисунок 6.3.).

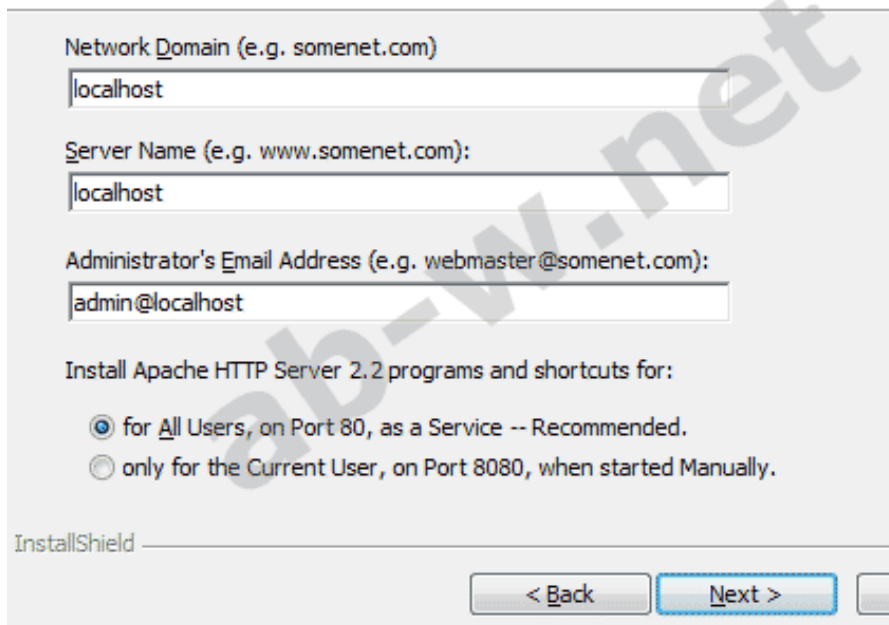


Рисунок 6.3.

6) Жмем **Next**.

7) Выбираем **Custom** (рисунок 6.4.).

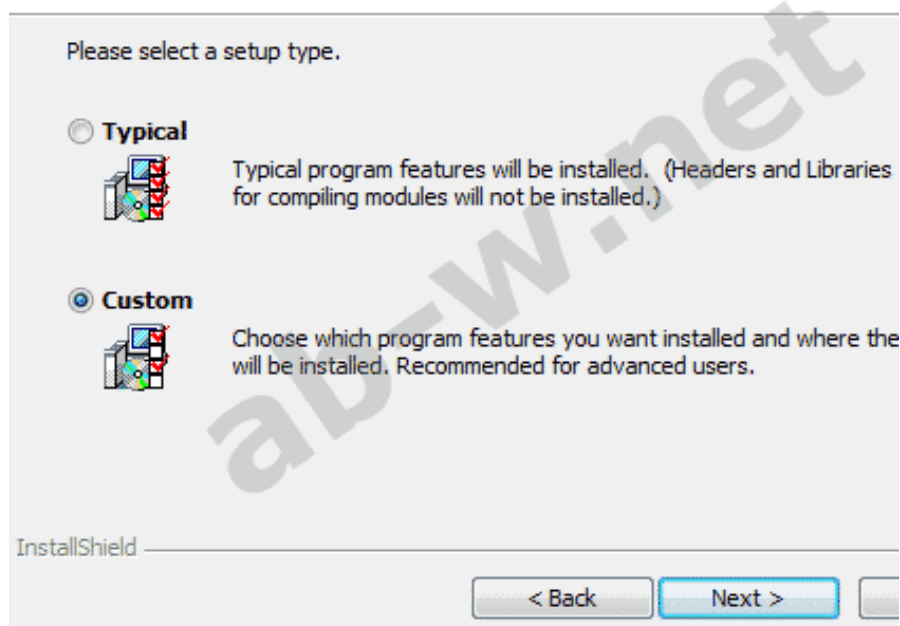


Рисунок 6.4.

8) Жмем **Next**.

9) В следующем окне (на рисунке 6.5.) жмем кнопку **Change...**, чтобы изменить путь установки:

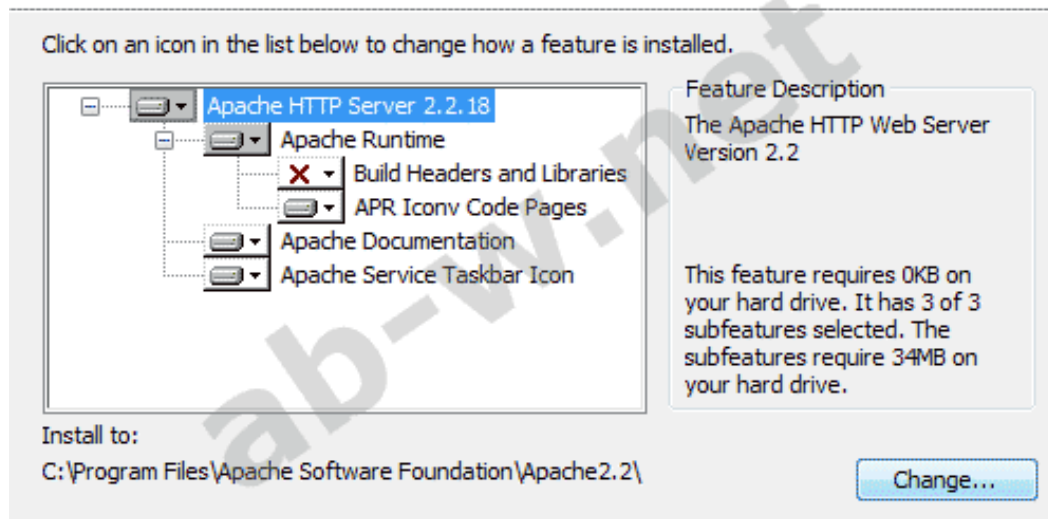


Рисунок 6.5.

10) Далее указываем папку, в которой у нас лежит дистрибутив Apache (рисунок 6.6.).

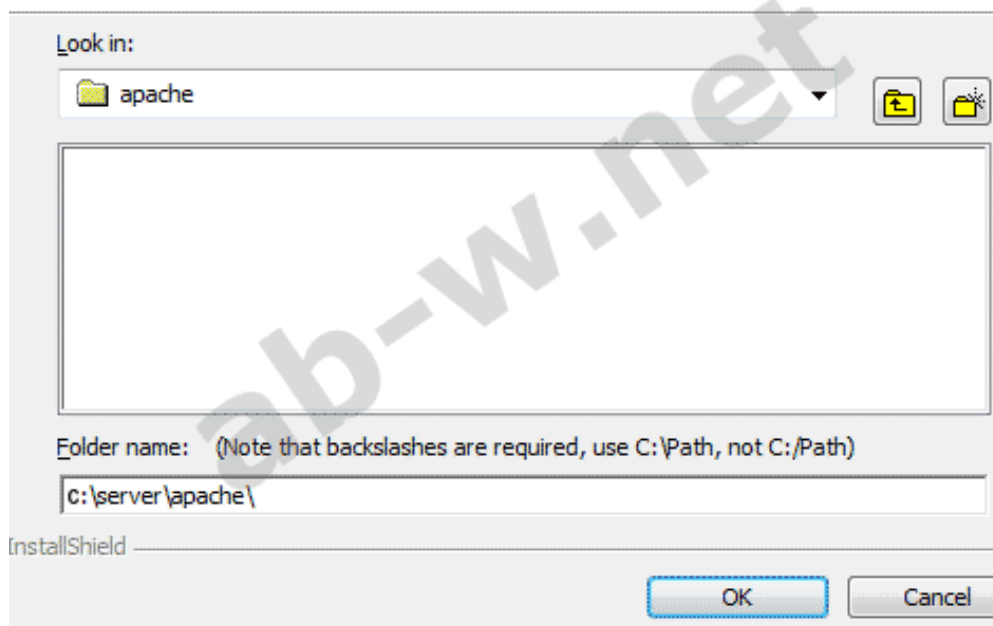


Рисунок 6.6.

11) Жмем **ОК**, затем **Next**.

12) Инсталлируем и в конце жмем **Finish** (рисунок 6.7.).

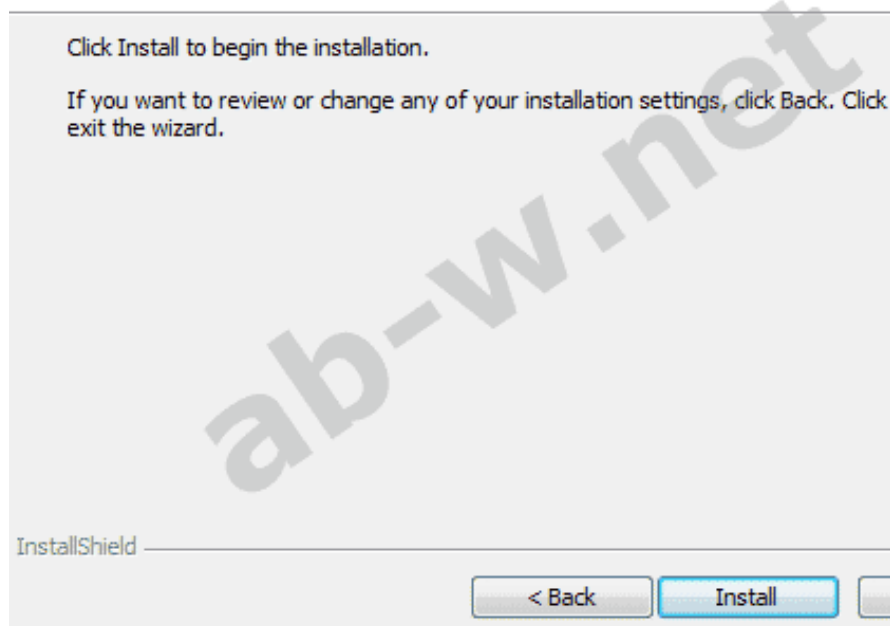


Рисунок 6.7.

Посмотрите на панель инструментов рабочего стола в правом нижнем углу (эта область в народе называется треем (system tray)). Там должен отобразиться значок установленного сервера Apache – розовое перо и белый круг, с зеленым треугольником в центре.

Проверим работает ли сервер. Открываем браузер, указываем адрес: `http://127.0.0.1/` или `http://localhost/`, жмем **Enter**. Страница с сообщением ***It works!*** говорит о том, что мы были внимательны и все сделали правильно (рисунок 6.8.).

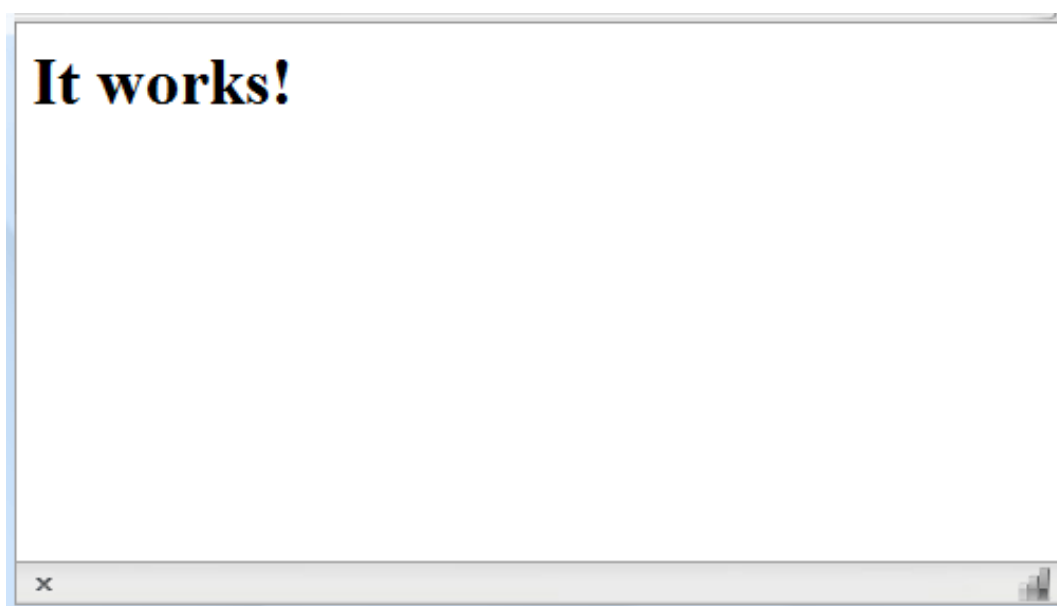


Рисунок 6.8.

3.2. Установка и настройка PHP интерпретатора

1) Запускаем инсталлятор php, жмем **Next** (рисунок 6.9.).

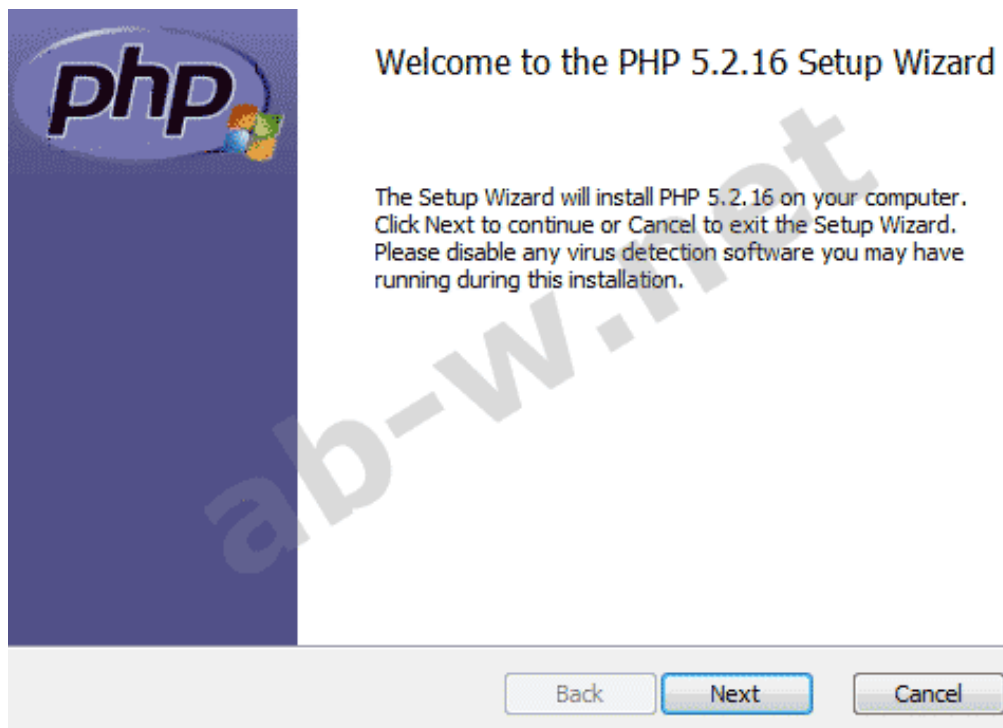


Рисунок 6.9.

2) Соглашаемся с лицензией, жмем **Next** (рисунок 6.10).

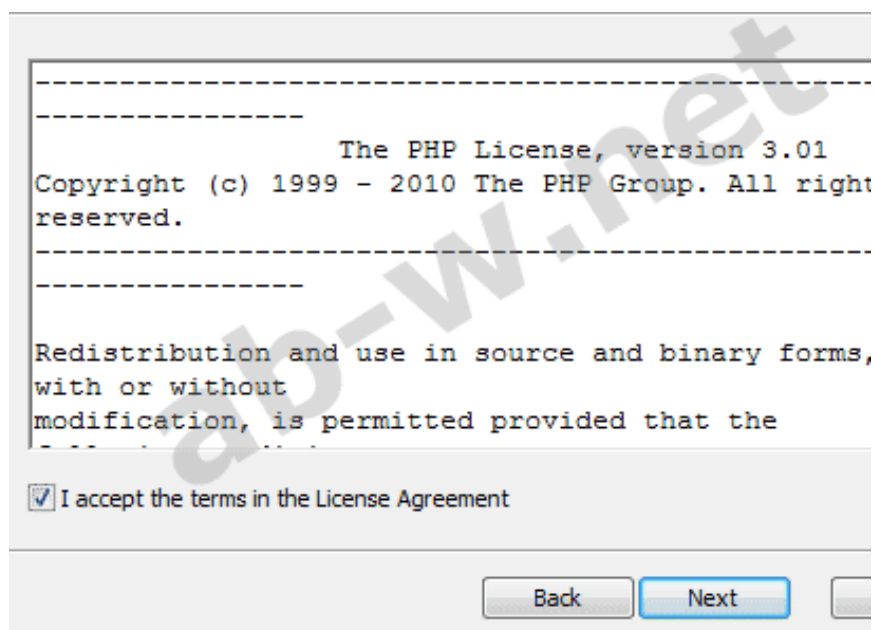


Рисунок 6.10.

3) Жмем **Browse...** (рисунок 6.11).



Рисунок 6.11.

4) Указываем путь установки (рисунок 6.12).

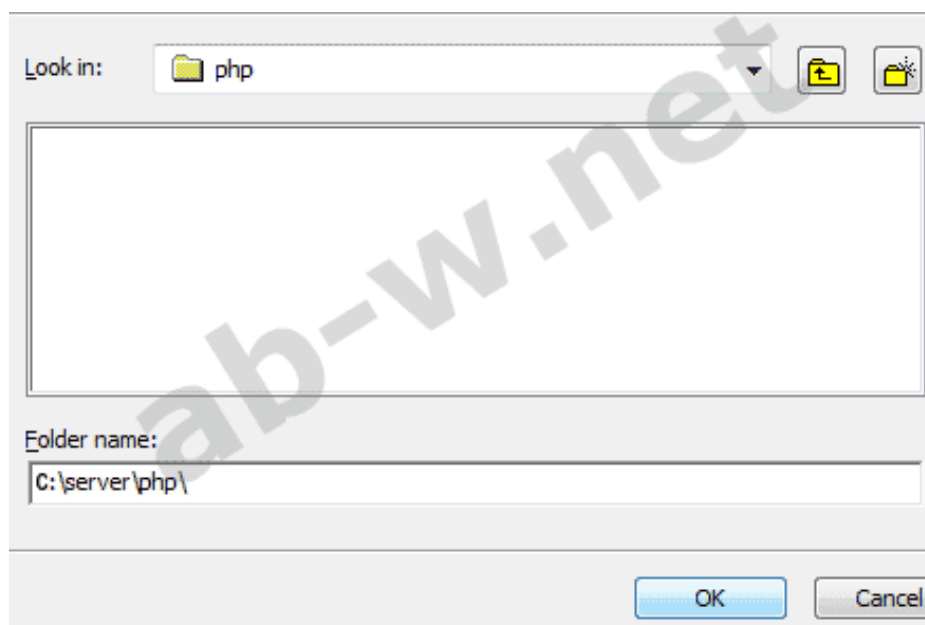


Рисунок 6.12.

5) **OK, Next.**

6) Выбираем тип модуля установленного сервера Apache (рисунок 6.13).

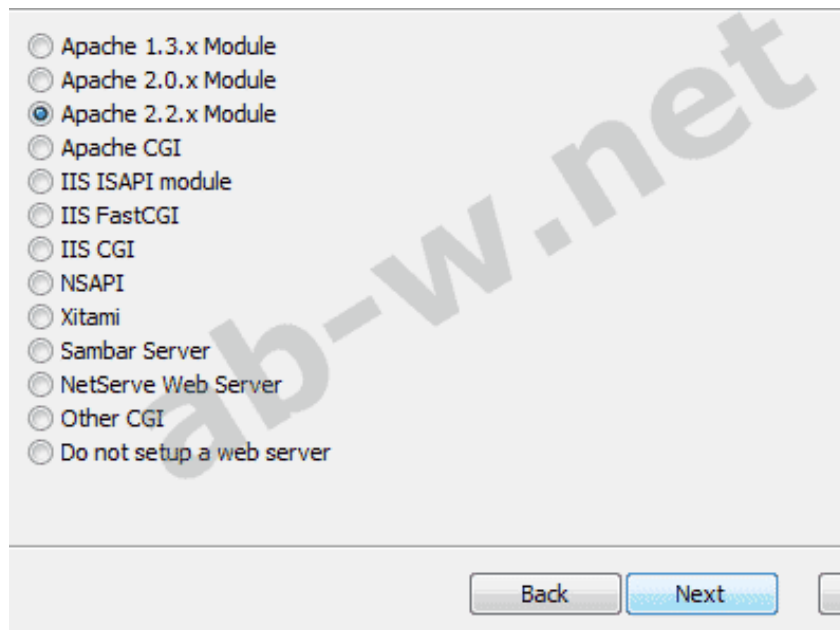


Рисунок 6.13.

7) Жмем **Next**.

8) Выбираем конфигурационную директорию установленного сервера Apache (рисунок 6.14).

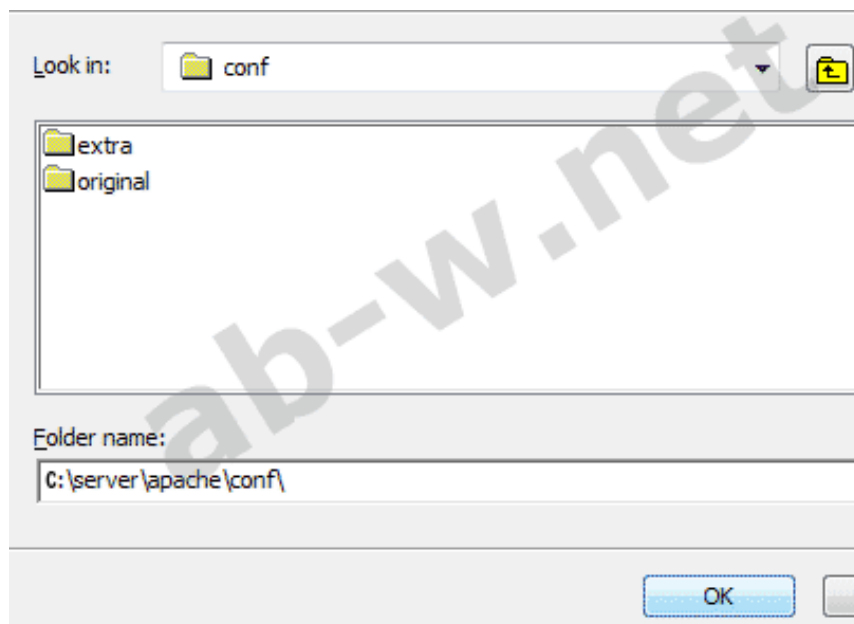


Рисунок 6.14.

9) **OK, Next**.

10) Далее в Extencions выбираем Entire feature will be installed.. (рисунок 6.15).

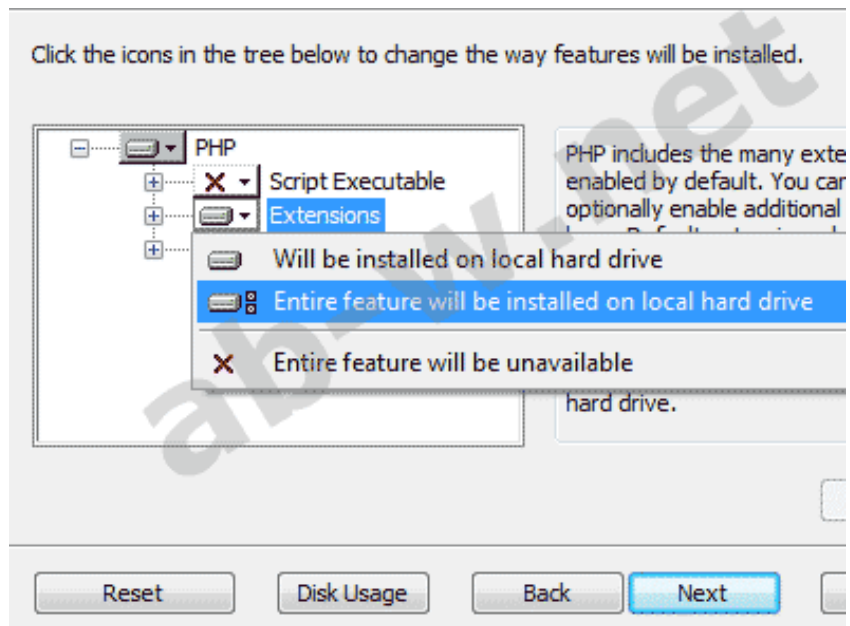


Рисунок 6.15.

11) Жмем **Next**.

12) Инсталлируем (рисунок 6.16).

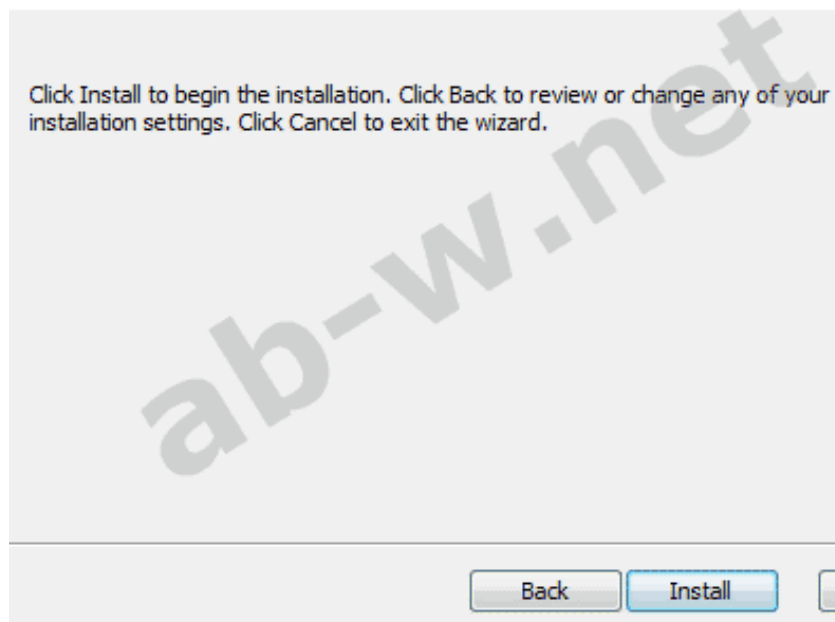


Рисунок 6.16.

13) Жмем **Finish** (рисунок 6.17).

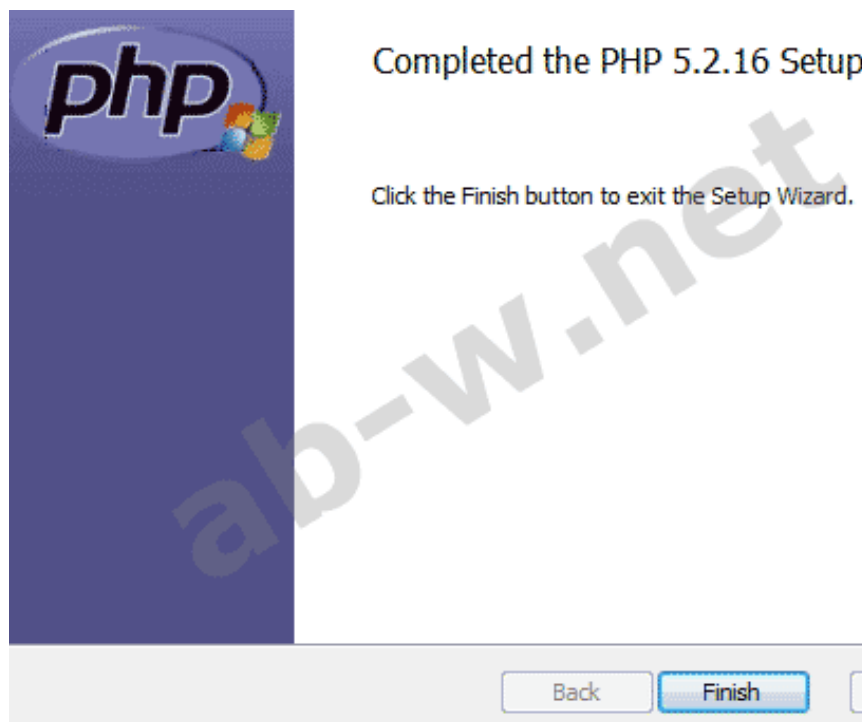


Рисунок 6.17.

3.3. Установка локального сервера баз данных MySQL

1) Запускаем инсталлятор MySQL (рисунок 6.18).

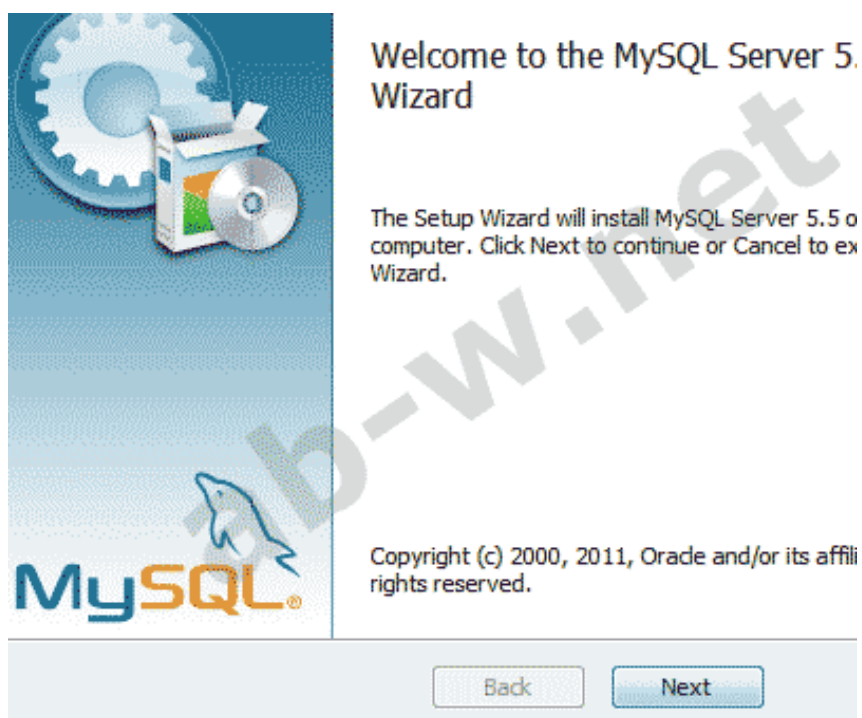


Рисунок 6.18.

2) Жмем **Next**.

3) Принимаем лицензионное соглашение, жмем **Next** (рисунок 6.19).

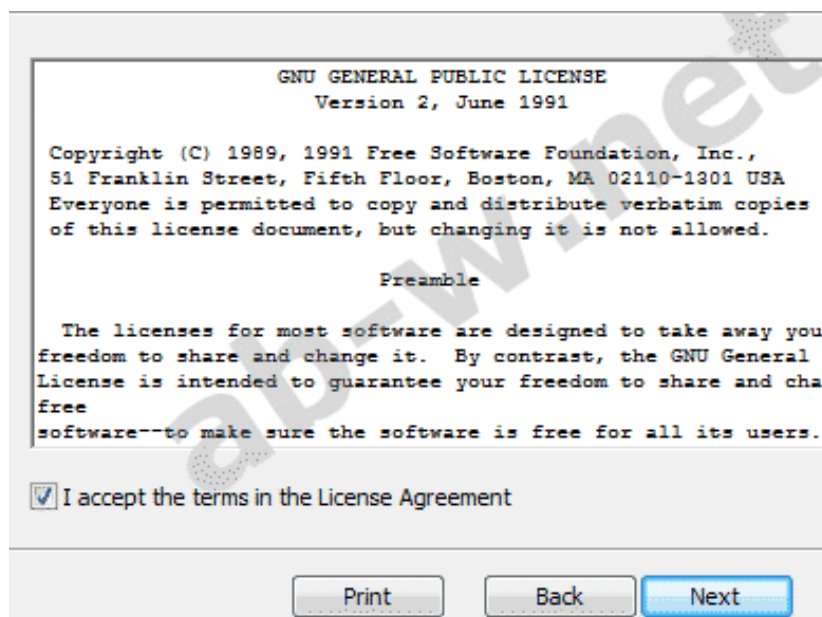


Рисунок 6.19.

4) В следующем окне выбираем **Custom** (рисунок 6.20).

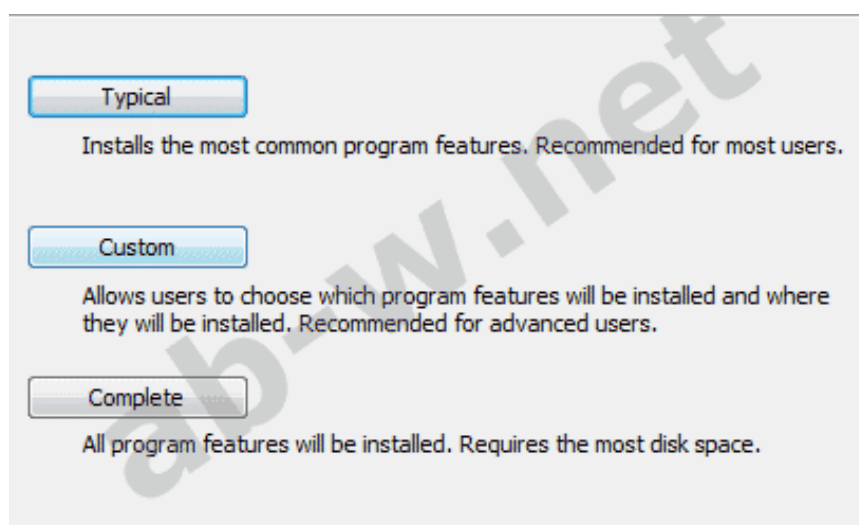


Рисунок 6.20.

5) Далее **Browse...**, чтобы изменить путь установки (рисунок 6.21).

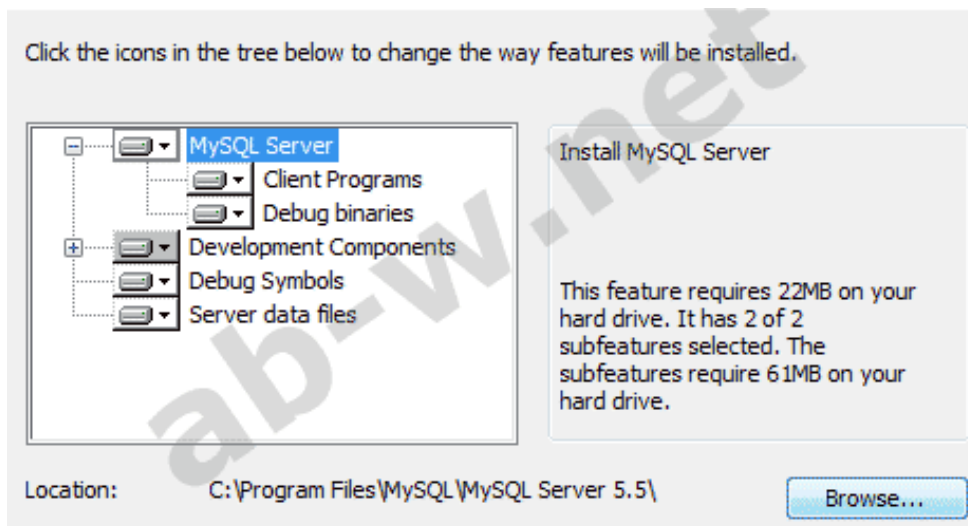


Рисунок 6.21.

6) Указываем папку (рисунок 6.22).

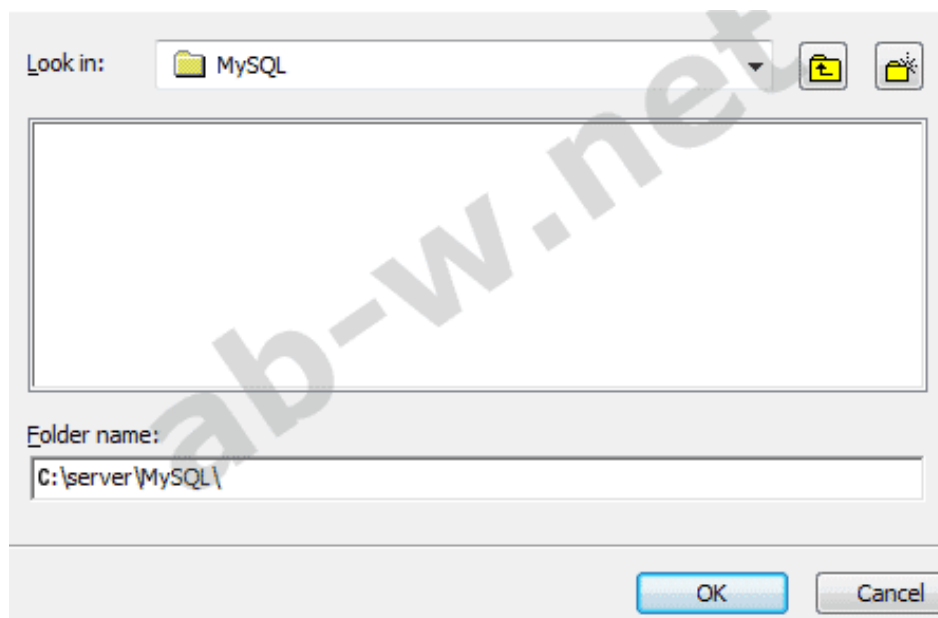


Рисунок 6.22.

7) Жмем **ОК**, затем **Next**.

8) Инсталлируем.

9) **Next** (рисунок 6.23).

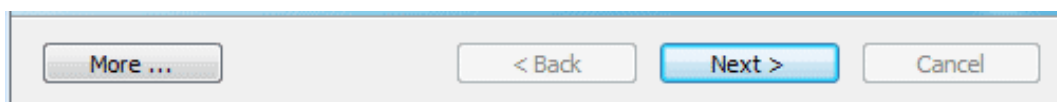



Рисунок 6.23.

10) В следующем окне жмем снова **Next**, в конце **Finish** и начинаем конфигурирование (рисунок 6.24).



Рисунок 6.24.

11) Если процесс конфигурирования не начался автоматически, запустите файл  MySQLInstanceConfig, который находится в папке bin, по адресу C:\server\MySQL\bin\.

12) Далее жмем **Next** (рисунок 6.25).

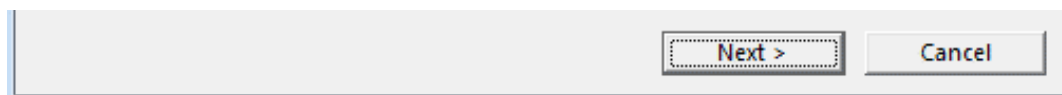


Рисунок 6.25.

12) Выбираем Standart Configuration, жмем **Next** (рисунок 6.26).

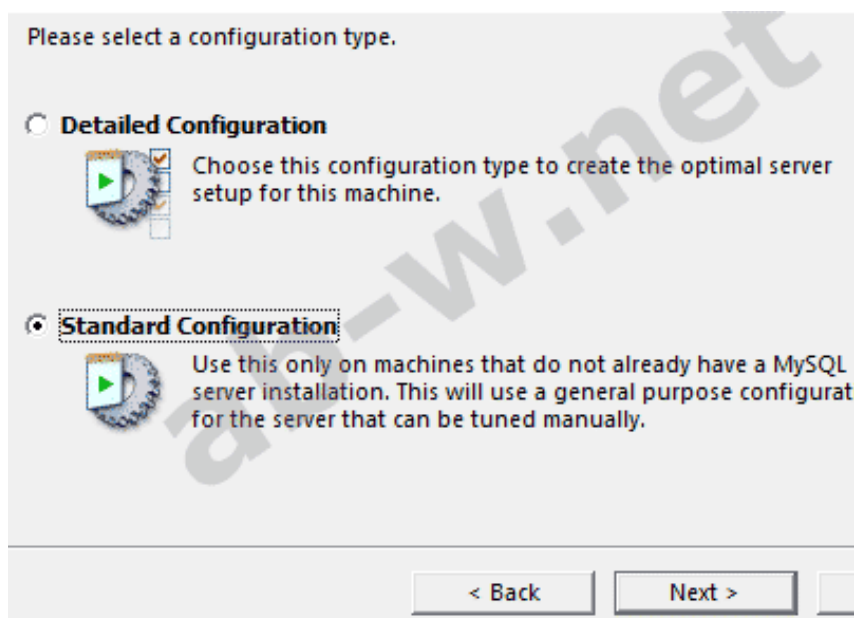


Рисунок 6.26.

13) Снова **Next** (рисунок 6.27).

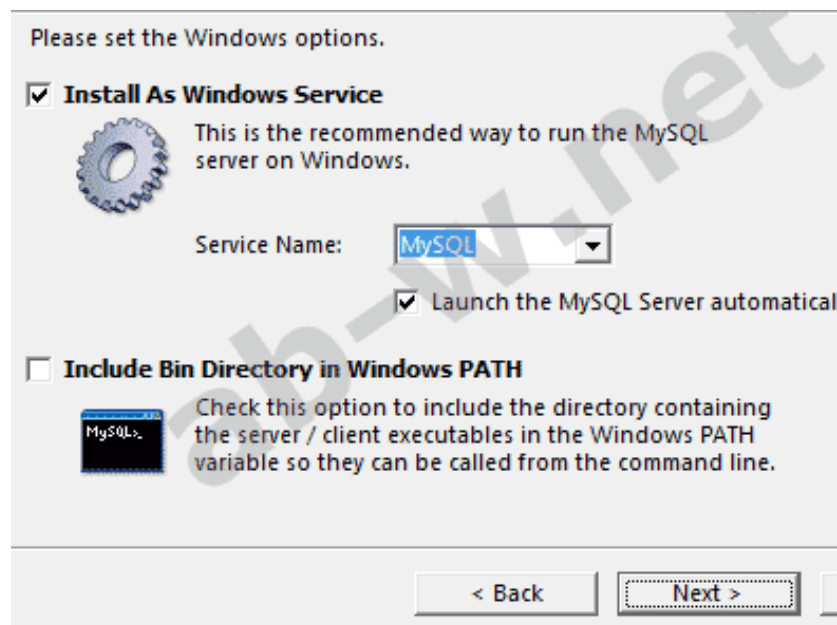


Рисунок 6.27.

14) В следующем окне указываем корневой пароль, который нужно запомнить (root password), например, 55555 (рисунок 6.28).



Рисунок 6.28.

15) Жмем **Next**, затем **Execute**.

16) Если все сделано верно, программа выдаст соответствующий отчет (рисунок 6.29).

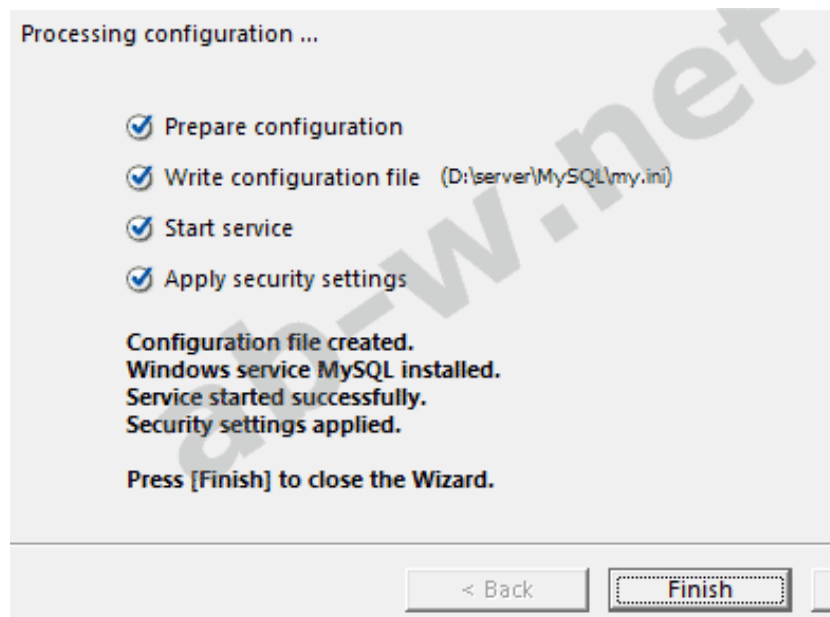


Рисунок 6.29.

17) Жмем **Finish**.

3.4. Настройка файла *httpd.conf*

Так как мы будем использовать сервер Apache в связке с PHP, нам необходимо настроить файл конфигурации *httpd.conf*, который находится на вашем локальном компьютере в директории *C:\server\apache\conf*.

Действуйте в следующем порядке:

- 1) Вызовите панель управления сервером двойным кликом по его значку в трее, нажмите **Stop**.
- 2) Двойным щелчком мыши откройте файл *httpd.conf*.
- 3) Найдите строчку *ServerRoot*, впереди нее не должно быть символа решетки #. Впишите адрес папки с установленным сервером Apache, результатом должна быть строка следующего содержания: *ServerRoot "C:/server/apache"* (это строка может быть изначально).
- 4) Найдите строчку *DocumentRoot*
"C:/server/apache/htdocs" и замените ее на *DocumentRoot "C:/server/www"* – она укажет серверу путь к папке, где будут храниться страницы сайта и тому подобное.
- 5) Найдите строчку *<Directory*
"C:/server/apache/htdocs"> и замените ее на *<Directory "C:/server/www">*.

- 6) Строчку `DirectoryIndex index.html` замените на `DirectoryIndex index.php index.php3 index.html index.htm`.
- 7) Строчку `ScriptAlias /cgi-bin/` "`C:/server/apache/cgi-bin/`" замените на `ScriptAlias /cgi-bin/ "C:/server/www/cgi-bin/"`.
- 8) `<Directory "C:/server/apache/cgi-bin">` замените на `<Directory "C:/server/www/cgi-bin">`.
- 9) В контейнере `<IfModule mime_module>` ниже строчки `AddType application/x-gzip .gz .tgz` добавьте две другие: `AddType application/x-httpd-php .php` и `AddType application/x-httpd-php .php3`.
- 10) Сохраните измененный файл там же.

3.5. *Настройка PHP интерпретатора*

- 1) Откройте двойным щелчком файл `php.ini` из папки `C:\server\php\`.
- 2) В разделе "Paths and Directories" должна присутствовать строка `extension_dir = "C:\server\php\ext"`.
- 3) В разделе "Dynamic Extensions" найдите; Be sure to appropriately set the `extension_dir` directive., внизу этой строчки поместите:
`extension=php_gd2.dll`
`extension=php_mbstring.dll`
`extension=php_mysql.dll`
`extension=php_mysqli.dll`
`extension=php_pdo.dll`
`extension=php_pdo_mysql.dll`
`extension=php_sqlite.dll`
- 4) Где-то в конце файла найдите строчку; `End:` и удалите все, что за ней следует.
- 5) Сохраните измененный файл там же.

Завершив редактирование файлов, создайте папку `www` в папке `server` и папку `cgi-bin` в папке `www`. Перезапустите компьютер. В результате в трее, в статусе сервера появился зеленый треугольник вместо

красного квадрата, если нет и выведено окно с предупреждением об ошибке, значит изменение файла `httpd.conf` было неточным и вам придется внимательно все проверить, наклоны слэшей, кавычки и так далее.

`www` — корневая директория (`root directory`) локального сервера, именно здесь следует сохранять все документы (папки, страницы, изображения, скрипты и другие файлы) вашего сайта.

3.6. Тестируем PHP интерпретатор

1) Создайте новый файл в редакторе (Notepad2 или Notepad+) и напишите код:

```
<?php
echo phpinfo();
?>
```

2) Сохраните файл в директории `C:\server\www\` как `index.php`. В браузере наберите `http://127.0.0.1` или `http://localhost/` ⇒ должна быть показана таблица.

Мы имеем локальный сервер в связке с интерпретатором.

3.7. Тестируем соединение с сервером баз данных MySQL

1) Небольшой скрипт:

```
<?php
$dblocation = "127.0.0.1";
$dbname = "test";
$dbuser = "root";
$dbpasswd = ""; /* Укажите пароль, который вы вводили
при
установке MySQL */
$dbcnx = @mysql_connect($dblocation, $dbuser, $dbpasswd);
if (!$dbcnx)
{
echo "Не доступен сервер MySQL";
exit();
}
```

```

}
if (!@mysql_select_db($dbname,$dbcnx))
{
echo "Не доступна база данных";
exit();
}
$ver = mysql_query("SELECT VERSION()");
if(!$ver)
{
echo "Ошибка в запросе";
exit();
}
echo mysql_result($ver, 0);
?>

```

2) Сохраняем скрипт в папке C:\server\www\ как mysql.php и набираем в браузере <http://localhost/mysql.php> ⇒ будет показан серийный номер MySQL сервера:

5.5.12

3.8. Установка *phpMyAdmin*

1) Скачайте бесплатно дистрибутив менеджера по управлению базами данных с официального сайта php-myadmin.ru, последнюю стабильную версию, файлом с расширением all-languages.zip.

2) Распакуйте архив в папку phpmyadmin, и поместите её в папку www.

3) Скачайте [config.inc.php](#), извлеките из архива и скопируйте его в папку phpmyadmin.

4) Откройте config.inc.php, найдите строчку \$cfg['Servers'][\$i]['password'] = 'pass'; ⇒ вместо pass укажите корневой пароль (root), который вы использовали при установке сервера MySQL ⇒ сохраните файл.

5) В браузере наберите <http://localhost/phpmyadmin/> ⇒ будет показана титульная страница менеджера (рисунок 6.30).



Рисунок 6.30.

б) Установив вышеперечисленные компоненты, мы получили полноценный локальный web-сервер и всё необходимое для организации дальнейшей работы.

3.9. Создание веб-страниц

Для начала нужно ознакомиться с основными тегами, которые применяются при создании любого веб-сайта. Их нужно уметь видеть и понимать предназначение, тогда не будет проблем с изменениями и дополнениями на странице. С остальными тегами вы можете ознакомиться с помощью интернета.

- ❖ **<html>** - этот тег стоит в самом начале любой страницы, и объявляет кодировку и саму страницу.
- ❖ **</html>** - этим тегом обязательно заканчиваются все страницы, точнее объявляют конец страницы.
- ❖ **<head> </head>** - между этими тегами находится невидимая часть сайта, документация.
- ❖ **<title><.title>** - между этими тегами пишется название вашего сайта.
- ❖ **<body> </body>** - между этими тегами пишется весь контент сайта - наполнение страницы.
- ❖ **<table> <tr> <td>** - это таблица, одна из самых важных конструкций при построении сайта.
- ❖ **<h> ... </h> <p>...</p> ...** - теги для написания заголовков, параграфов, списков для вставки текста на страницу.
- ❖ **<div></div>** - блочный элемент, на котором можно построить весь сайт, изучить свойства тега div нужно обязательно.

- ❖ `` - так вставляется изображение.
- ❖ ` Страница 1 ` - так делается ссылка на страницу.

Для начала создадим простую HTML страницу и назовем ее - index.html. Она будет главной и название должно быть только таким, все остальные страницы можно называть как угодно, но обязательно с расширением `_.html` и обычно создают папку для хранения изображений.

Должно получиться, как показано на рисунке 6.31.

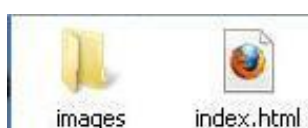


Рисунок 6.31.

Состоит страница минимум из трех частей:

хедер (шапка сайта) - в котором находятся все кодировки, стили, название страницы, основные логотипы, ключевые слова.

контент - здесь все наполнение, другими словами все, что нужно показать в браузере, это тексты, изображения и т.д.

футер (подвал) - здесь обычно счетчики, копирайты, возможно меню, реклама, или что-то другое.

`<!DOCTYPE_ _>` обязательная часть

`<html>` объявление языка написания страницы

`<head>`

`<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />` - кодировки сайта

`<title>`Здесь название страницы, отображаемое в верхнем левом углу браузера`</title>`

`</head>`

`<body>`

--- здесь располагают ХЕДЕР и меню сайта

----- дальше контентная часть

это уже веб-страница, которую можно сохранить в файл с названием – **ya.html** и посмотреть в браузере.

--- внизу обычно ФУТЕР

</body>

</html> - объявление окончания данной страницы

3.10. Простой пример веб-страницы

```
<html>
```

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
<title>Первая страница.</title>
```

```
<style>
```

```
/* здесь описаны стили оформления ссылок*/
```

```
a:link {
    color: #0033FF;
    text-decoration: none;
}
```

```
a:hover {
    color: #666633;
    text-decoration: underline;
}
```

```
/* здесь описаны стили оформления контента*/
```

```
#kontent {
    text-align: center;
}
```

```
</style>
```

```
</head>
```

```
<body id="kontent">
```

```
<h1 align="center">Моя первая страница!</h1>
```

```
<p align="center">Так можно создавать свою первую страницу.</p>
```

```
<p align="center">Для начала приведен простой пример, по ссылке  
можно посмотреть пример,
```

```
<br> который создан из таблиц.</p>
```

```
<p align="center"><a href="http://kapon.com.ua/sign_table.php" title="пример  
страницы">
```

```
пример страницы</a> построенной на таблицах.</p>
```

```
</body>
```

```
</html>
```


3.11. Запуск сайта в локальной сети

- 1) Для этого необходимо первым делом настроить локальную сеть между двумя компьютерами (А и В).
- 2) На компьютере А должен быть установлен Apache.
- 3) Прописать VirtualHost в Apache.
- 4) в <VirtualHost> прописывать нужно только IP предварительно указав директиву NameVirtualHost
например:
NameVirtualHost 192.168.1.10:80
<VirtualHost 192.168.1.10:80>
ServerName myhost // - А вот здесь надо указывать URL
DocumentRoot /var/www/default/html // - путь
</VirtualHost>
 - a. на компьютере А:
 - b. NameVirtualHost 192.168.1.33:80
 - c. <VirtualHost 192.168.1.33:80>
 - d. и в hosts IP 127.0.0.1 заменить на 192.168.1.33.
 - e. на компе В создать аналогичный hosts
- 5) Настроить файл hosts (C:\Windows\System32\drivers\etc\hosts) на стороне компьютера В, т.е. внести запись к **ПРИМЕРУ:**
 - a. **192.168.1.33 site1**
192.168.1.33 site2
 - b. , где 192.168.1.33 - IP компа А.
- 6) Проверить.

4. Контрольные вопросы

- 1) Как проводится настройка и установка Apache?
- 2) Что произойдет с веб-сервером, если в конфигурации допущена ошибка? (для самостоятельного изучения)
- 3) Что такое httpd.conf?
- 4) Как создать простую веб-страницу?
- 5) Что нужно сделать, чтобы запустить сайт в локальной сети?
- 6) Назначение файла Hosts?

5. Список рекомендуемой литературы