

ИМИТАТОР javaNetSim

Основной задачей имитатора javaNetSim является имитация работы всех уровней стека протоколов TCP/IP. Для этого имитируется работа протоколов каждого из уровней, чем достигается полная имитация работы сети. Имитатор javaNetSim является объектно-ориентированным и написан на языке Java. Программы написанные на этом языке являются машиннонезависимыми, т.е. имитатор javaNetSim будет работать на любом компьютере, для которого есть виртуальная Java машина.

Архитектура имитатора javaNetSim

Архитектура имитатора javaNetSim выглядит следующим образом. В основе лежит класс Simulation (Имитация), который содержит объекты классов Link (Линия) и Node (Узел). Этот класс предназначен для объединения устройств и линий связи в единую сеть. Класс Link содержит ссылки на объекты класса Node, и предназначен для соединения двух узлов между собой. Класс Node содержит ссылки на объекты класса Link и является наиболее общей моделью сетевого устройства. Все реальные сетевые устройства являются производными от объекта класса Node и соответствуют модели стека протоколов TCP/IP:

- Hub (Концентратор) – DataLink Layer Device (Устройство физического уровня) – имеет пять портов, т.е. в нему возможно подключить до пяти линий связи;
- Router (Маршрутизатор) – Network Layer Device (Устройство сетевого уровня) – имеет два порта, а также стек протоколов TCP/IP (ProtocolStack);
- PC (Компьютер) – Applications Layer Device (Устройство уровня приложений) – имеет один порт, стек протоколов TCP/IP, а также возможность выполнять клиентскую или серверную часть какоголибо приложения.

Для взаимодействия с пользователем каждому сетевому устройству нужно графическое соответствие. Его обеспечивают следующие классы:

- GuiHub (Графический пользовательский интерфейс концентратора);
- GuiRouter (Графический пользовательский интерфейс маршрутизатора);
- GuiPC (Графический пользовательский интерфейс компьютера). Как сами

сетевые устройства, так и графический пользовательский интерфейс сетевых устройств должен быть единым. Этим объединением занимается класс SandBox (Рабочая область).

Главное меню программы

Меню File(файл) позволяет создавать, открывать и сохранять конфигурации сетей для их дальнейшего использования. Меню содержит пять пунктов:

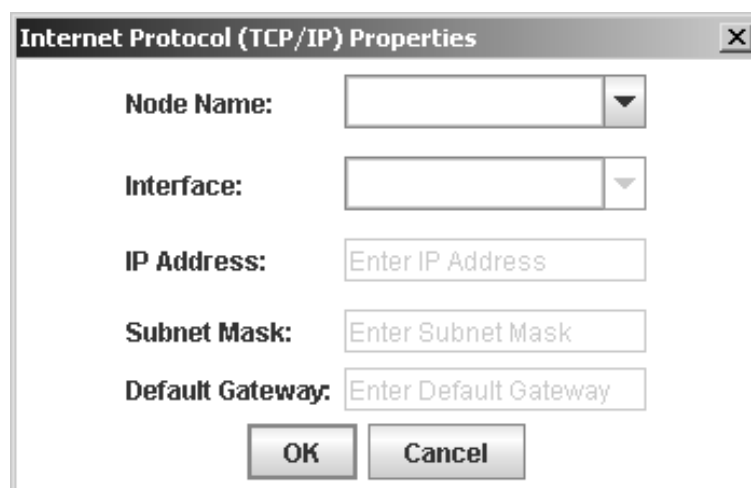
- New(Новый) – создать пустую конфигурацию;
- Open...(Открыть...) – открыть существующую конфигурацию;
- Save...(Сохранить...) – сохранить текущую конфигурацию;
- Save As...(Сохранить Как...) – сохранить текущую конфигурацию под новым именем;
- Exit(Выход) – выйти из имитатора javaNetSim.

Режим проектирования сети доступен из **меню Simulation(Имитация)**. Это меню позволяет создавать новые сетевые устройства (такие как: концентратор, маршрутизатор или компьютер) и изменять сетевые параметры уже существующих устройств. Меню содержит два пункта:

- подменю Add(Добавить) – позволяет создать компьютер(PC), маршрутизатор(Router) или концентратор(Hub);
- подменю Tools(Инструменты), в котором есть пункт Set TCP/IP Properties(Установить свойства TCP/IP) для изменения свойств TCP/IP.

В имитаторе javaNetSim задание IP-адреса узла, маски подсети и шлюза по умолчанию происходит через диалог "Internet Protocol (TCP/IP) Properties", вызов которого осуществляется через меню "Simulation > Tools

> Set TCP/IP Properties". В этом окне для выбранного устройства (Node Name) и интерфейса (Interface) можно задать IP-адрес (IP Address) и маску подсети (Subnet Mask) для интерфейса и шлюз по умолчанию (Default Gateway) для узла. Для компьютера доступен всего один интерфейс, для маршрутизатора – два.



Управление параметрами имитатора доступно из **меню Environment(Окружение)** и позволяет изменять режим отображения информации, а также очищать область вывода результатов. Меню содержит четыре пункта:

- Clear Console(Очистить консоль) – удаляет все записи из вкладки "консоль";
- Clear Node Information(Очистить информацию об устройствах) – удаляет все записи из вкладки "информация об устройствах";
- Show simulation messages for:(Показывать сообщения имитатора для:) – позволяет задать режим вывода на вкладку "консоль" сообщений только определенных уровней стека протоколов TCP/IP. Есть возможность выбрать следующие уровни: Link and DataLink Layers(Физический и канальный уровни), Network Layer(Сетевой уровень), Transport Layer(Транспортный уровень), Application Layer(Уровень приложений);
- Show headers:(Показывать заголовки) – задает режим вывода на вкладку "консоль" сообщений с названиями уровней и/или с типами пакетов.

С помощью меню "Environment > Show simulation messages for:" можно отключить сообщение от тех уровней стека протоколов TCP/IP в которых нет необходимости. Это уменьшит количество информации выводимой в "консоль" и облегчит поиск нужных данных.

Графический интерфейс имитатора javaNetSim.

Основное окно программы представляет собой инструмент взаимодействия пользователя с имитатором. С помощью этого инструмента пользователь может добавлять, удалять и соединять между собой сетевые устройства, а также работать с сетью на любом из четырех уровней стека протоколов TCP/IP.

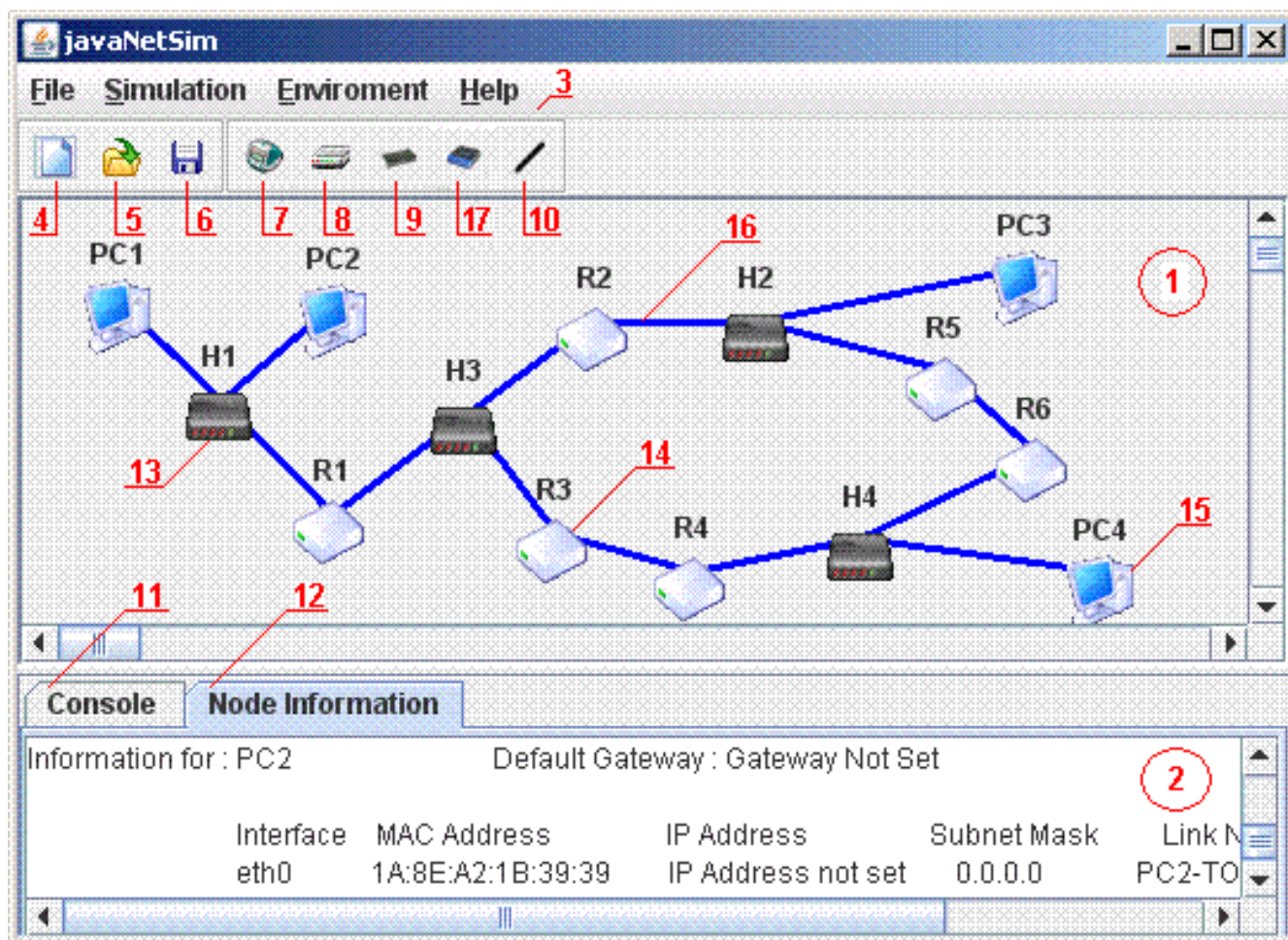


Рис. Основное окно программы javaNetSim.

Основное окно программы логически разделено на четыре части:

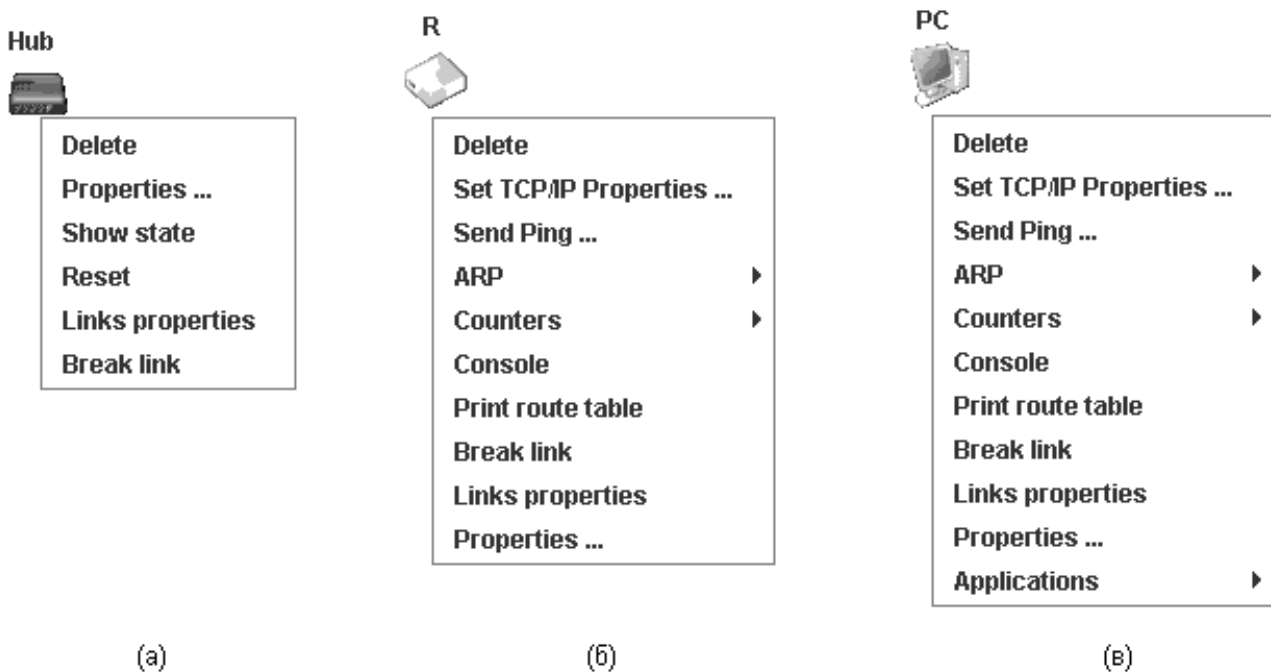
1. рабочая область, обозначенная на рисунке цифрой (1) – содержит сетевые устройства и линии связи между ними:
 - Концентратор на пять сетевых интерфейсов (13).
 - Маршрутизатор соединяющий две подсети (14).
 - Компьютер или конечный узел сети (15).
 - Линия связи между двумя сетевыми устройствами (16).
2. область вывода результатов (2) – содержит две вкладки:
 - вкладка "консоль" (11) – содержит журнал передачи пакетов по сети
 - вкладка "информация об устройствах" (12) – для каждого интерфейса всех сетевых устройств содержит IP-адрес, маску подсети и шлюз по умолчанию.
3. главное меню (3) – содержит основные действия по управлению имитатором;
4. линейка инструментов – содержит следующие кнопки:

- кнопка "создать пустую конфигурацию" (4);
- кнопка "открыть существующую конфигурацию" (5);
- кнопка "сохранить текущую конфигурацию" (6);
- кнопка "создать компьютер" (7);
- кнопка "создать маршрутизатор" (8);
- кнопка "создать концентратор" (9);
- кнопка "создать коммутатор" (17);
- кнопка "создать соединение" (10).

Контекстное меню javaNetSim.

Контекстное меню, вызываемое щелчком правой кнопкой мыши, отличается для устройств работающих на разных уровнях стека протоколов TCP/IP.

На рис. изображены контекстные меню для устройств: физического уровня - концентратор (а), сетевого уровня - коммутатор (б) и прикладного уровня - компьютер (в).



Контекстные меню устройств.

Link Properties (Свойства линий связи) вызывает диалог, который позволяет установить коэффициент пропускания для интерфейса, показывающий какой процент пакетов линия связи подключенная к этому интерфейсу будет пропускать.

Меню ARP позволяет управлять таблицей протокола ARP на выбранном устройстве содержит три подпункта:

- Add static entry to ARP table – вызывает два диалоговых окна: в первом вводится MAC адрес, а во втором IP адрес, после чего в ARP таблицу заносится статическая запись о связи IP и MAC адресов;
- Remove entry from ARP table – вызывает диалоговое окно, позволяющее ввести IP адрес, для которого будет удалена запись из ARP таблицы;
- Print ARP table – выводит на вкладку "консоль" ARP таблицу выбранного сетевого устройства.

Описание консольного режима (командной строки).

Устройства в javaNetSim конфигурируются в командной строке подобной Cisco IOS. Подсоединение к узлу осуществляется через Telnet или на IP-адрес любого из интерфейсов или через последовательный порт компьютера, связанный с консольным портом (Console).

Последний способ предпочтительнее, потому что процесс конфигурирования маршрутизатора может изменять параметры IP-интерфейсов, что приведет к потере соединения, установленного через Telnet. Кроме того, по соображениям безопасности доступ к реальному маршрутизатору через Telnet обычно запрещён.

В рамках данного курса конфигурация маршрутизаторов будет осуществляться посредством Console вызываемую через контекстное меню маршрутизатора.

Все команды и параметры могут быть сокращены (например, "**enable**" - "**en**", "**configure terminal**" - "**conf t**"); если сокращение окажется неоднозначным, маршрутизатор сообщит об этом, а по нажатию **табуляции** выдаст варианты, соответствующие введенному фрагменту.

В любом месте командной строки для получения помощи может быть использован **вопросительный знак**:

router#? /список всех команд данного контекста с комментариями/

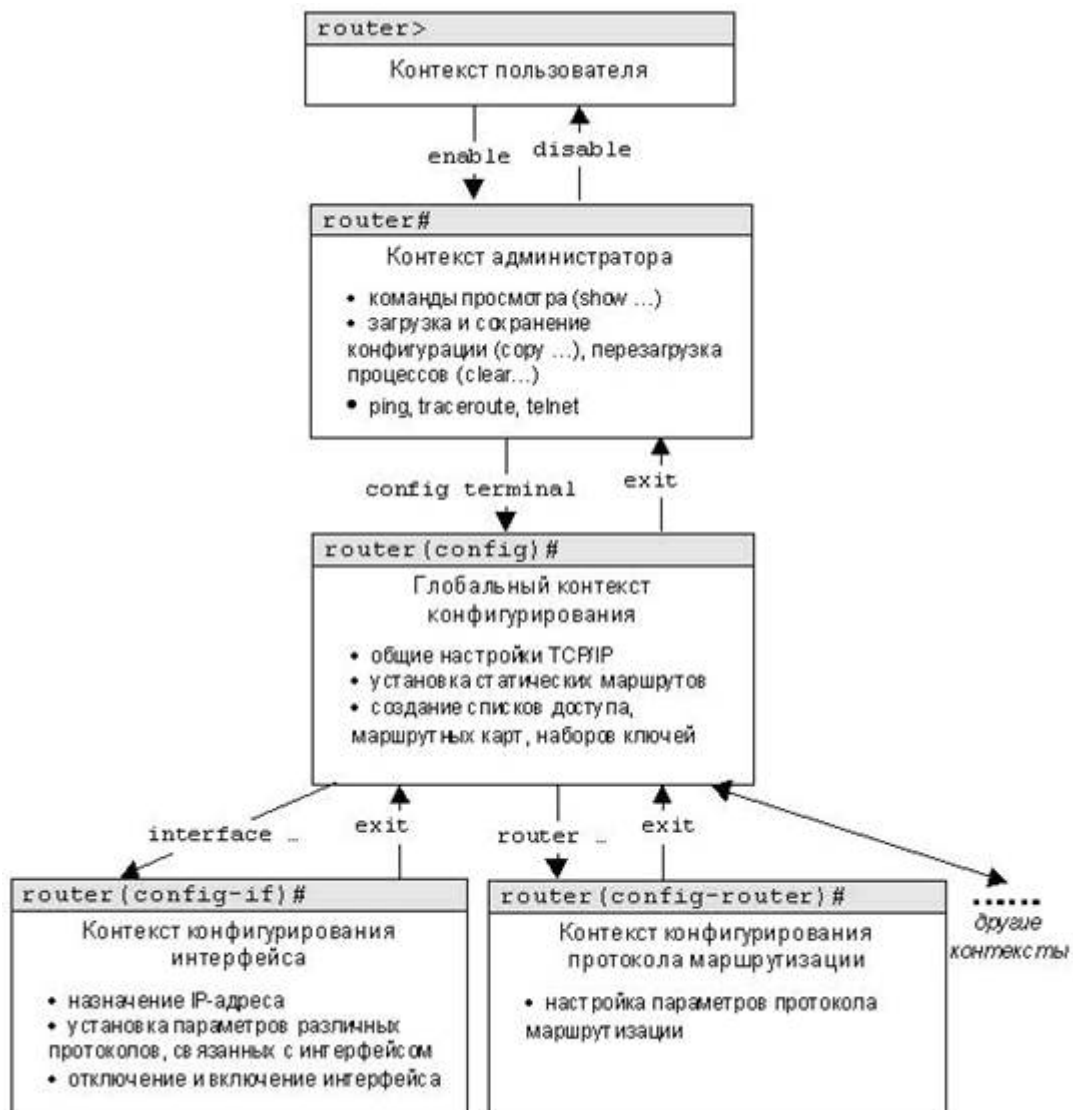
router#co? /список всех слов в этом контексте ввода, начинающихся на "co" - нет пробела перед "?"/

router#conf ? /список всех параметров, которые могут следовать за командой config - перед "?" есть пробел

При работе в командной строке существует несколько контекстов (режимов ввода команд).

Схема контекстов javaNetSim

Существует множество контекстов конфигурирования. Некоторые контексты конфигурирования находятся внутри других контекстов конфигурирования. Наиболее часто используемые контексты показаны на рисунке ниже.



Упрощенная схема контекстов javaNetSim

Вид приглашения командной строки в контекстах конфигурирования, которые будут встречаться наиболее часто:

```
router(config)# /глобальный/  
router(config-if)# /интерфейса/ router(config-router)# /  
динамической маршрутизации/ router(config-line)# /  
терминальной линии
```

Выход из глобального контекста конфигурирования в контекст администратора, а также выход из любого подконтекста конфигурирования в контекст верхнего уровня производится командой **exit** или **Ctrl-Z**. Кроме того, команда **end**, поданная в любом из контекстов конфигурирования немедленно завершает процесс конфигурирования и возвращает оператора в контекст администратора.

Контекст пользователя.

Контекст открывается при подсоединении к маршрутизатору; обычно при подключении через сеть требуется пароль, а при подключении через консольный порт пароль не нужен. В этот же контекст командная строка автоматически переходит при продолжительном отсутствии ввода в контексте администратора. В контексте пользователя доступны только простые команды (некоторые базовые операции для мониторинга), не влияющие на конфигурацию маршрутизатора. Вид приглашения командной строки:

```
router>
```

Вместо слова `router` выводится имя маршрутизатора, если оно установлено.

Контекст администратора.

Контекст "еxес" открывается командой **enable**, поданной в контексте пользователя; при этом обычно требуется пароль администратора. В контексте администратора доступны команды, позволяющие получить полную информацию о конфигурации маршрутизатора и его состоянии, команды перехода в режим конфигурирования, команды сохранения и загрузки конфигурации. Вид приглашения командной строки:

```
router#
```

Обратный переход в контекст пользователя производится по команде **disable** или по истечении установленного времени неактивности.

Завершение сеанса работы - команда **exit**.

В эмуляторе javaNetSim сразу настроен контекст администратора.

Глобальный контекст конфигурирования.

Открывается командой **config terminal** ("конфигурировать через терминал"), поданной в контексте администратора. Глобальный контекст конфигурирования содержит как непосредственно команды конфигурирования маршрутизатора, так и команды перехода в контексты конфигурирования подсистем маршрутизатора, например:

Контекст конфигурирования интерфейса.

Открывается командой **interface имя_интерфейса** (например `interface serial0`), поданной в глобальном контексте конфигурирования;

Контекст конфигурирования процесса динамической маршрутизации.

Открывается командой **router протокол номер_процесса** (например, `router ospf 1`, поданной в глобальном контексте конфигурирования.

Любая команда конфигурации вступает в действие немедленно после ввода, а не после возврата в контекст администратора.

Работа в javaNetSim с протоколами уровня приложений.

В имитаторе javaNetSim имеется возможность работы со следующими протоколами уровня приложений стека протоколов TCP/IP:

- Echo (UDP и TCP реализации),
- SNMP,
- TELNET.

Работа с протоколом Echo в javaNetSim.

Имитатор javaNetSim позволяет использовать протоколы UDP или TCP в качестве транспортных протоколов для протокола Echo.

Для установки echo-сервера в режим прослушивания порта надо выбрать пункт:

- "Applications" -> "Start udp echo server to listen" - для Echo-UDP
- "Applications" -> "Start tcp echo server to listen" - для Echo-TCP.

После этого в появившемся диалоговом окне следует ввести номер порта, на котором выбранное приложение будет ожидать сообщения. Теперь с любого другого узла можно отсылать сообщения на этот узел и получать ответы.

Для того, чтобы послать эхо-запрос, необходимо в контекстном меню выбрать

- "Applications" -> "Send data via udp echo client" - для Echo-UDP
- "Applications" -> "Send data via tcp echo client" - для Echo-TCP

и ввести четыре параметра:

1. IP-адрес компьютера, на котором запущен echo-сервер;
2. номер порта на котором echo-сервер ожидает сообщения;
3. сообщение – любой текст;
4. количество посылаемых сообщений.

Протокол Echo обладает простой структурой, поэтому при помощи telnet-клиента можно подключиться к Echo-TCP-серверу. В таком режиме нажатие любой клавиши на клавиатуре будет сопровождаться выводом ее на экран терминала.

Работа с протоколом SNMP в javaNetSim.

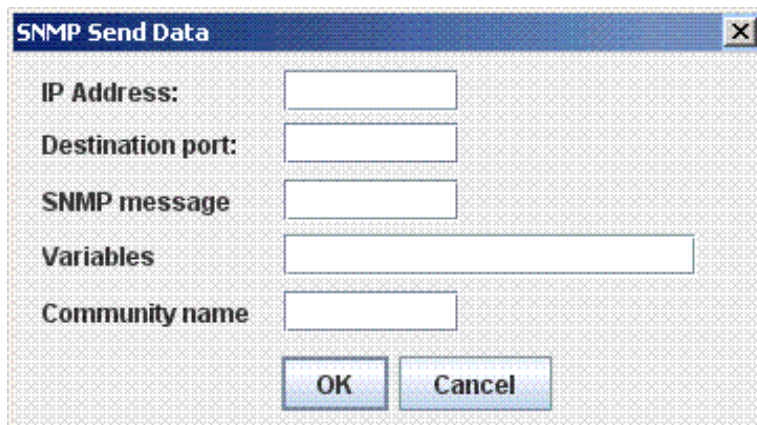
Для запуска SNMP агента: Application -> Start SNMP Agent и задать два параметра:

- порт, на котором SNMP агент будет ожидать пакеты;
- имя группы доступа для SNMP агента.

Чтобы послать запрос SNMP агенту: Application -> Send SNMP message и заполнить поля.

1. IP Address – IP адрес компьютера на котором установлен SNMP агент.
2. Destination Port – порт на котором SNMP агент ожидает пакеты.
3. SNMP message – SNMP запрос, может принимать значения: get, getnext, set.
4. Variables – SNMP переменные описываемые деревом MIB.

5. Community name – имя группы доступа, которое должно совпадать с именем группы доступа установленным при создании агента.

A screenshot of a Windows-style dialog box titled "SNMP Send Data". It contains five input fields: "IP Address:", "Destination port:", "SNMP message", "Variables", and "Community name". At the bottom right are "OK" and "Cancel" buttons.

Создание SNMP запроса.

Поле Variables имеет специальный формат, различный для запросов get(getnext) и set. Если SNMP запрос является get или getnext запросом, то строка переменных должна выглядеть следующим образом:

`<переменная>[;<переменная>]`

Например: `ip.address_eth0;device.hostname.`

А если SNMP запрос является set запросом, то в строке переменных к каждой переменной добавляется значение:

`<переменная>=<значение> [<переменная>=<значение>]`

Например: `ip.address_eth0="192.168.10.3"`

Результаты запроса будут выведены на вкладку "Console" javaNetSim. Например:

```
PC2 SNMP Protocol Data Application Received getResponse:
'IP.Address_Eth0=172.168.0.2' , 'Device.Hostname=PC1'
```

Список SNMP переменных, поддерживаемых имитатором javaNetSim.

Переменные, которые имеют режим доступа "только для чтения".

- Counter.InputIP – количество пришедших IP пакетов;
- Counter.OutputIP – количество отправленных IP пакетов;
- Counter.ARP – количество обработанных ARP пакетов;
- Counter.InputTCP – количество пришедших TCP пакетов;
- Counter.OutputTCP – количество отправленных TCP пакетов;
- Counter.ReceiveDuplicatedTCP – количество дублирующихся пакетов TCP полученных устройством;
- Counter.SendDuplicatedTCP – количество дублирующихся пакетов TCP отправленных устройством;
- Counter.SendAckTCP – количество посланных ACK пакетов;
- Counter.InputUDP – количество пришедших UDP пакетов;

- Counter.OutputUDP – количество отправленных UDP пакетов;
- Device.AllInterfaces – список всех возможных интерфейсов устройства;
- Device.AvailableInterfaces – список всех доступных интерфейсов устройства;
- Device.Hostname – имя устройства;
- Device.MACaddress_Eth0 – MAC адрес устройства на интерфейсе Ethernet0;
- IP.AllInterfaces – список всех возможных интерфейсов устройства работающих по протоколу IP;
- IP.ARPTTable – ARP таблица для устройства;
- SNMP.revision – версия модификации SNMP;
- SNMP.version – версия SNMP.
- Некоторые SNMP переменные имеют режим доступа "чтение и запись".
- IP.DefaultGateway – шлюз по умолчанию;
- IP.Address_Eth0 – IP адрес интерфейса Ethernet0;
- IP.SubnetMask_Eth0 – маска интерфейса Ethernet0;
- SNMP.CommunityName – имя группы доступа для SNMP агента.

Режим доступа определяет действия, которые можно производить с переменной. Если переменная имеет режим доступа только чтение, то попытка записать новое значение завершиться с ошибкой.

Работа с протоколом TELNET в javaNetSim.

Для запуска TELNET сервера необходимо выбрать пункт контекстного меню "Application" -> "Start telnet server to listen" и задать два параметра:

- порт, на котором TELNET-сервер будет ожидать пакеты;
- пароль для доступа к TELNET-серверу.

Для соединения с TELNET сервером необходимо выбрать пункт контекстного меню "Application" -> "Telnet client" и задать два параметра:

- IP адрес TELNET-сервера;
- порт, на котором TELNET-сервер ожидает пакеты.

После этого откроется окно терминала и если соединение прошло успешно появится приглашение ввести имя пользователя: login. После введения имени появится приглашение ввести пароль: password. После введения пароля, имя пользователя и пароль проверяются и, если они корректны, будет выведено приглашение в виде:

<имя компьютера> #

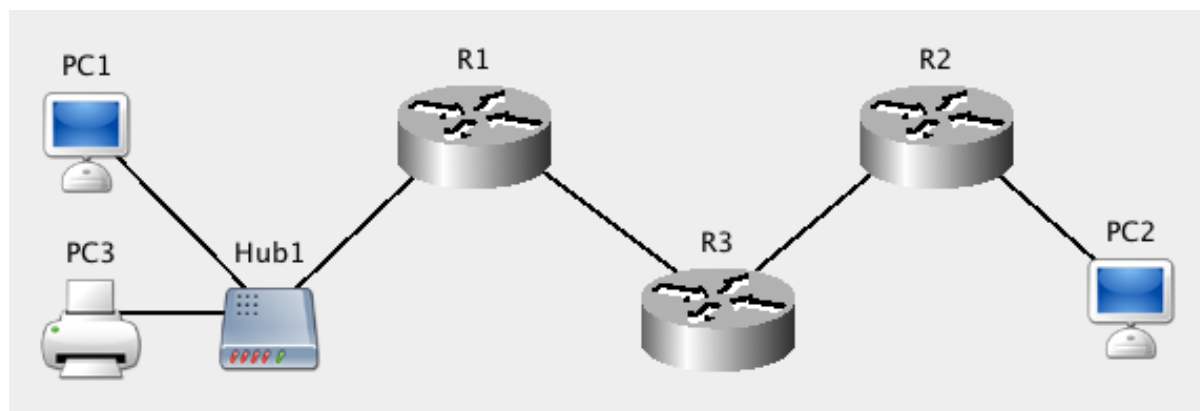
В javaNetSim для TELNET-сервера используется имя пользователя root и пароль, установленный при создании TELNET-сервера. В telnet доступны следующие команды:

- route – просмотр и редактирование сетевых маршрутов;
- arp – просмотр и редактирование ARP таблиц;
- snmp – запуск и остановка SNMP агента;
- counters – просмотр доступных сетевых счетчиков;
- passwd – изменение пароля на доступ к TELNET серверу;
- quit – закрыть TELNET сеанс;
- ? или help – посмотреть список доступных команд.

Пример выполнения лабораторной работы.

Исходные данные.

Исходная конфигурация сети:



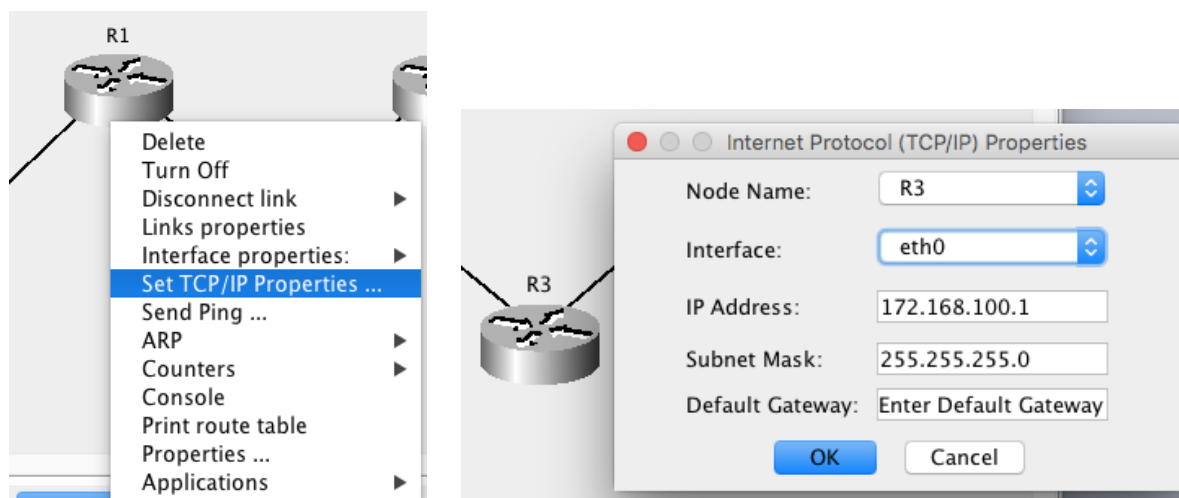
Структура исследуемой сетевой архитектуры (var0.jnst).

1. Файл со схемой сети: var0.jnst
2. Компьютер PC1 имеет IP-адрес 172.168.0.2
3. Компьютер PC2 имеет IP адрес 10.0.0.2
4. Сеть Hub1: 172.168.0.0/24
5. Сеть R1-R3: 172.168.100.0/24
6. Сеть R3-R2: 192.168.0.0/24

Порядок выполнения задания.

Начальная настройка узлов сети.

Сначала обеспечим корректную доставку пакетов в локальных сегментах сети. Воспользуемся в javaNetSim контекстным меню узла и диалогом Set TCP/IP Properties.



Для активных сетевых интерфейсов на всех узлах сети (на хостах, на конечных (тупиковых) маршрутизаторах, на промежуточных маршрутизаторах) установим корректные значения IP адресов (из заданного диапазона IP-сети), масок подсети и шлюзов по умолчанию.

Как видим, сети Hub1 (172.168.0.0) и R1-R3 (172.168.100.0) при использовании стандартной маски подсети для класса В были бы эквивалентными, поэтому будем использовать маску подсети, отличную от стандартной - маску 255.255.255.0.

Принтеру PC3 назначим любой свободный IP-адрес подсети Hub1, например 172.168.0.3 с маской 255.255.255.0.

Рассуждая подобным образом можно на основании имеющихся исходных данных корректно настроить сетевые параметры интерфейсов для всех узлов в сети.

Результаты начальной конфигурации интерфейсов показаны в таблице.

	IP адрес	Маска подсети	Шлюз по умолчанию:
PC1-eth0:	172.168.0.2	255.255.255.0	172.168.0.1
PC3-eth0:	172.168.0.3	255.255.255.0	172.168.0.1
R1-eth0:	172.168.0.1	255.255.255.0	172.168.100.1
R1-eth1:	172.168.100.2	255.255.255.0	172.168.100.1
R3-eth0:	172.168.100.1	255.255.255.0	-
R3-eth1:	192.168.0.2	255.255.255.0	-
R2-eth0:	192.168.0.1	255.255.255.0	192.168.0.2
R2-eth1:	10.0.0.1	255.0.0.0	192.168.0.2
PC2-eth0:	10.0.0.2	255.0.0.0	10.0.0.1

Такая конфигурация уже обеспечит доставку пакетов от PC1 к PC3, R1 и R3, но не доставку к R2 и PC2.

При попытке отправки пакета от PC1 на PC2, пакет сначала попадет на R1 (т.к. он является шлюзом по умолчанию для PC1), а оттуда на R3. Но на R3 пакет отбрасывается, т.к. маршрут для сети 10.0.0.0/8 на R3 пока отсутствует, а шлюза по умолчанию для R3 нет.

Таблицу маршрутизации можно посмотреть или через контекстное меню **Print route table** или в консоли командой **show ip route**.

Для R3 TM пока выглядит следующим образом:

```
R3# show ip route
Codes: C - connected, S - static, R - RIP,
B - BGP, O - OSPF, * - candidate default
C 192.168.0.2/255.255.255.0 is directly connected, eth1
C 172.168.100.1/255.255.255.0 is directly connected, eth0
R3# exit
```

Команды удобно дописывать табом, а помощь получать через «?» и help. Команды управления разбиты на домены, например, для роутинга:

- show ip route – показать записи в TM;
- configure terminal / ip route add – добавить запись в TM;
- configure terminal / no route add – удалить запись из TM.

Настройка промежуточных маршрутизаторов.

Чтобы обеспечить корректную доставку пакетов между хостами сети настроим таблицы маршрутизации на промежуточных маршрутизаторах. У нас такой маршрутизатор только один – R3.

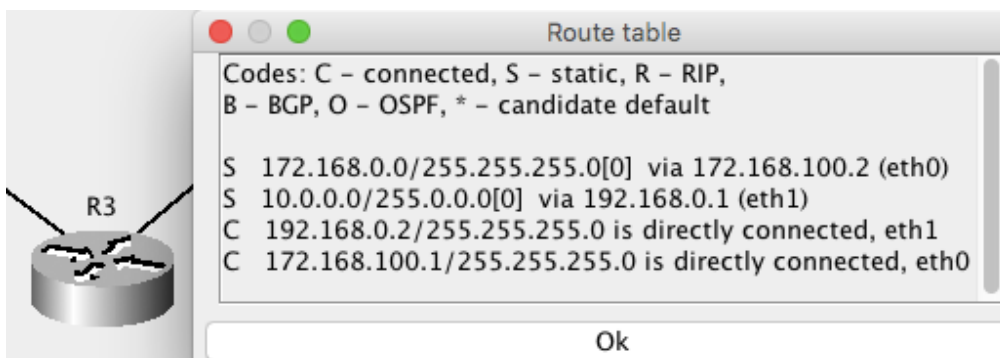
Для того чтобы пакеты между PC1 и PC2 прошли через R3 необходимо на R3 добавить маршруты для сетей 172.168.0.0/24 и 10.0.0.0/8.

```
R3# configure terminal
R3(config)# ip route add 172.168.0.0 255.255.255.0 172.168.100.2 eth0
Route added.
R3(config)# ip route add 10.0.0.0 eth0 255.0.0.0 192.168.0.1 eth1
Route added.
R3(config)# end
R3# exit
```

Чтобы проверить корректность добавления маршрута необходимо вновь вывести таблицу маршрутизации или в консоли маршрутизатора R3:

```
R3# show ip route
S 172.168.0.0/255.255.255.0[0] via 172.168.100.2 (eth0)
S 10.0.0.0/255.0.0.0[0] via 192.168.0.1 (eth1)
C 192.168.0.2/255.255.255.0 is directly connected, eth1
C 172.168.100.1/255.255.255.0 is directly connected, eth0
R3# exit
```

или в контекстном меню javaNetSim для нужного маршрутизатора **Print route table**:



Теперь любой пакет, попавший на R3 и имеющий в качестве подсети назначения 172.168.0.0/255.255.255.0 будет направлен на маршрутизатор R1.

Аналогичным образом можно настроить таблицы маршрутизации на остальных промежуточных маршрутизаторах сети.

В данной сети таких маршрутизаторов больше нет.

Проверка настроек сети.

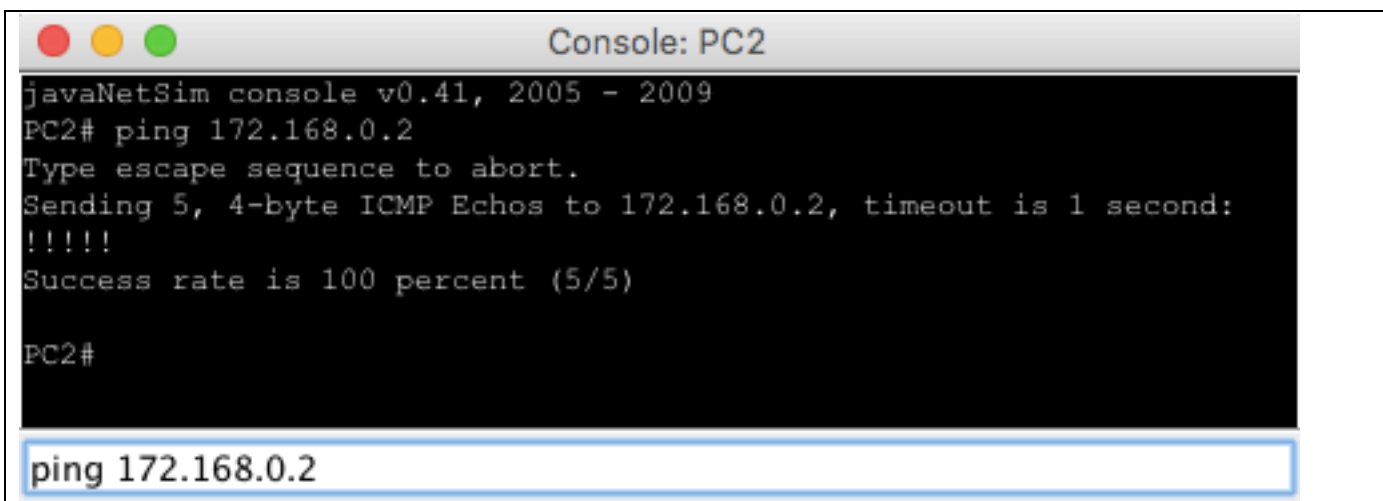
Для проверки корректности настроек выполним эхо-запросы с PC1 на PC2 и наоборот.

Отправим через контекстное меню PC1 эхо-запрос на PC2 и проанализируем результаты в главной Console javaNetSim:

```
PC1 Created Echo Request packet to 10.0.0.2
PC1 Sending packet from ProtocolStack (to 172.168.0.1).
...
PC2 ProtocolStack received packet from local Interface.
PC2 Confirmed Packet is for this Network Layer Device.
PC2 Created Echo Reply packet to 172.168.0.2
PC2 Sending packet from ProtocolStack (to 10.0.0.1).
...
PC1 ProtocolStack received packet from local Interface.
PC1 Confirmed Packet is for this Network Layer Device.
PC1 Echo reply packet received from 10.0.0.2
```

Как видим, на эхо-запрос пришёл ответ, следовательно таблица маршрутизации настроена верно.

Проверку можно выполнять и в консоли соответствующего узла командой ping, например:

A screenshot of a terminal window titled "Console: PC2". The window has a standard macOS-style title bar with red, yellow, and green buttons. The terminal text shows the execution of a ping command from PC2 to 172.168.0.2. The output indicates that 5 ICMP Echoes were sent successfully with a 100% success rate. At the bottom of the window, there is a text input field containing the command "ping 172.168.0.2".

```
javaNetSim console v0.41, 2005 - 2009
PC2# ping 172.168.0.2
Type escape sequence to abort.
Sending 5, 4-byte ICMP Echos to 172.168.0.2, timeout is 1 second:
!!!!
Success rate is 100 percent (5/5)

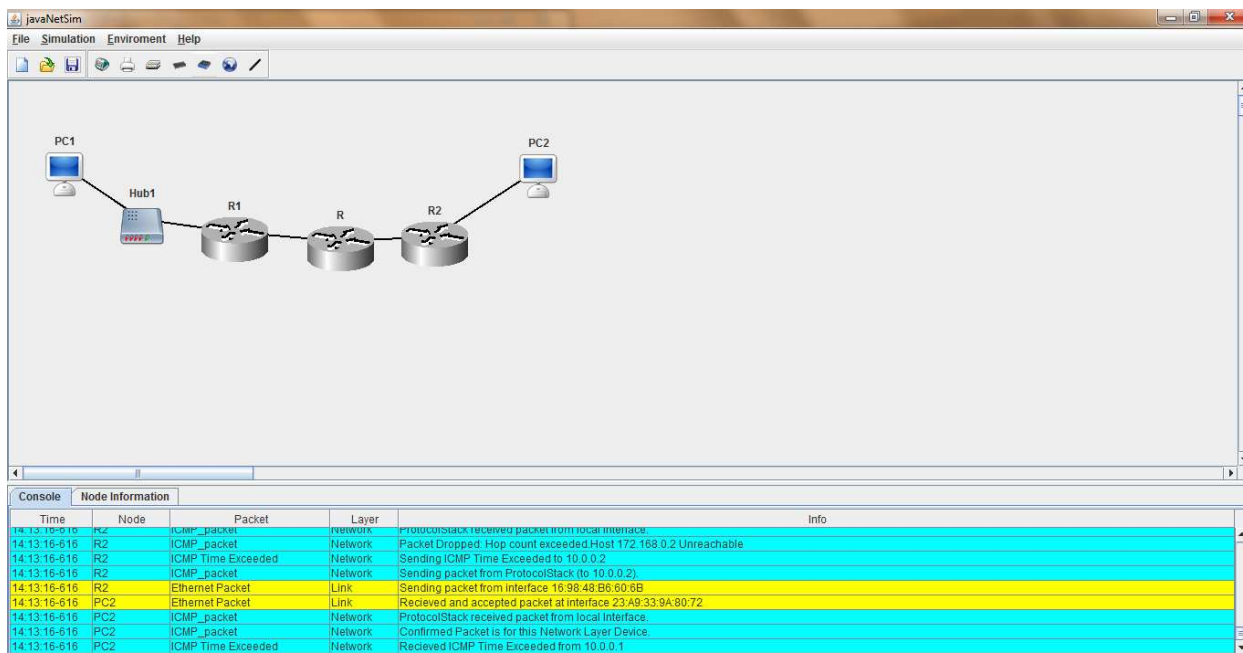
PC2#

ping 172.168.0.2
```

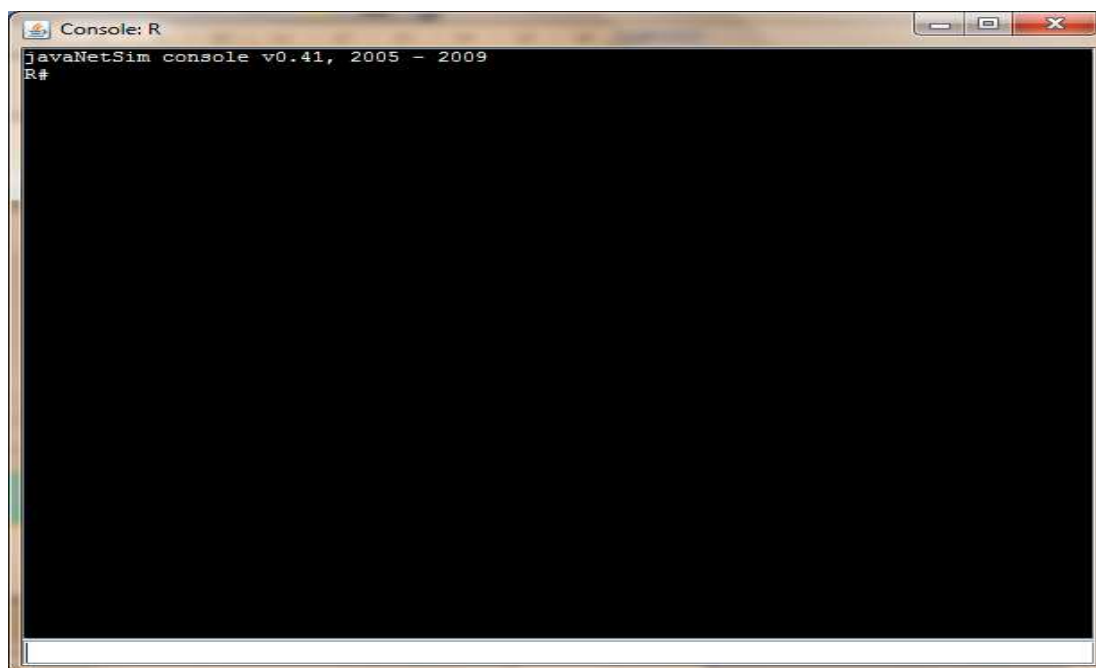
Аналогичным образом можно проверить связь между остальными узлами в сети. В данной сети маршрутизаторов больше нет.

Пример настройки маршрутизатора

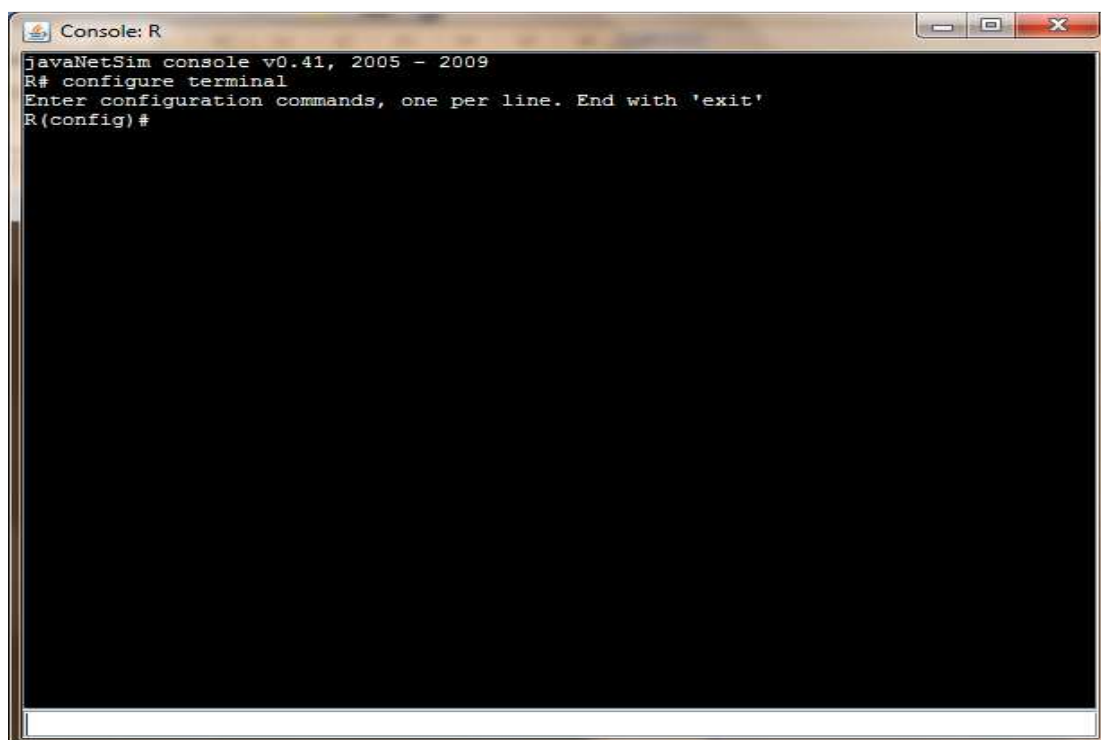
Схема сети приведена на рисунке



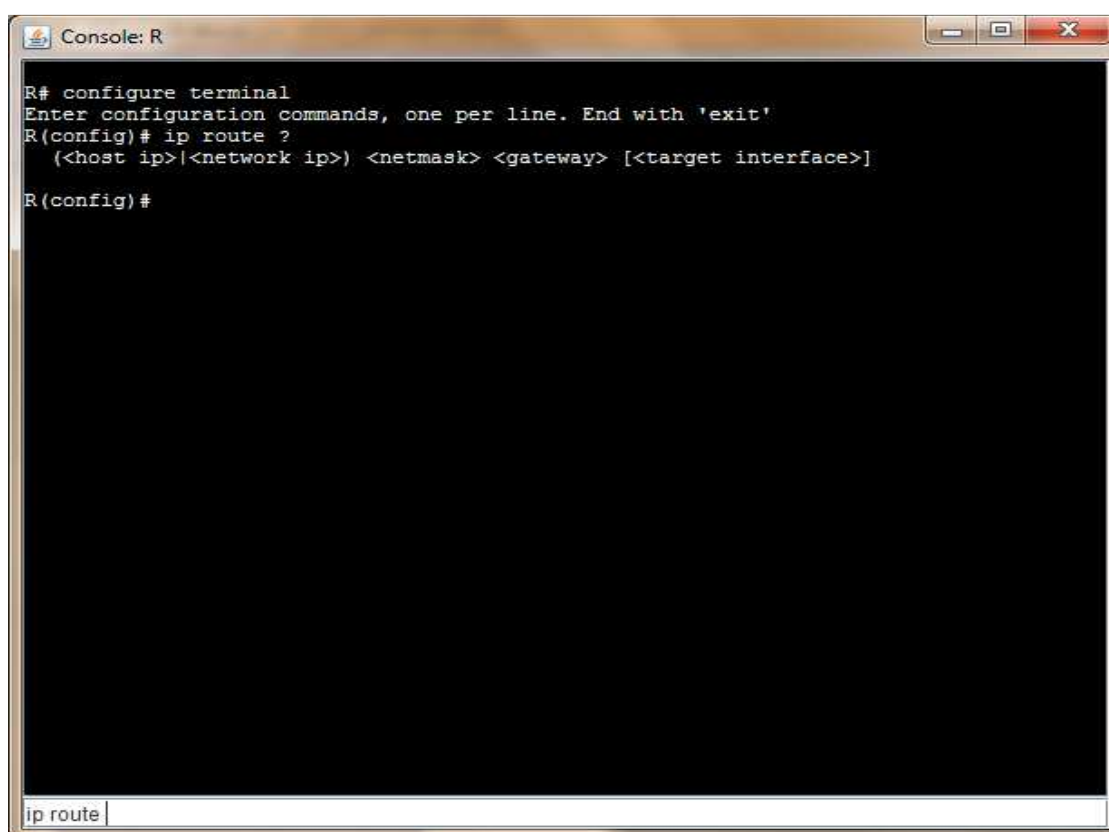
Кликнув правой кнопкой мыши по изображению маршрутизатора, выбрать пункт меню «Console», после чего откроется диалоговое окно:



В нижней строке необходимо набрать команду «configure terminal», в результате чего будет запущен режим конфигурирования роутера:

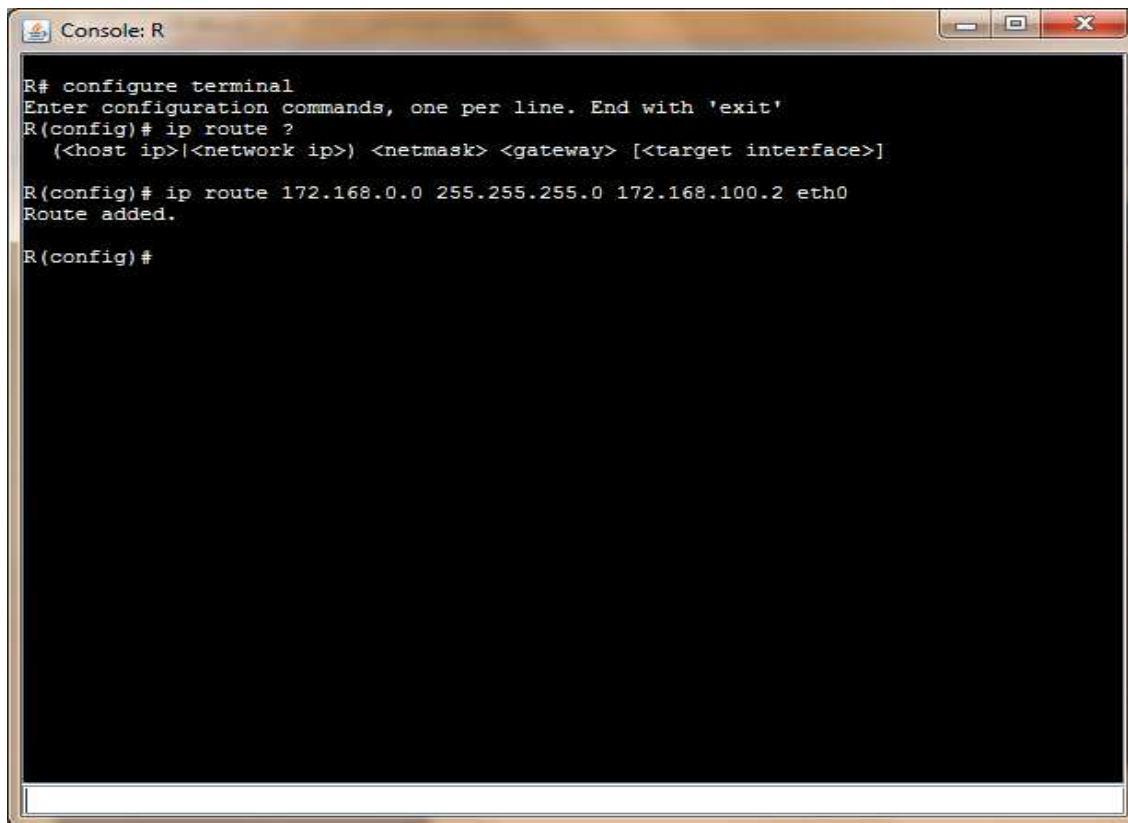


Для добавления маршрута используется команда «ip route»:



Для добавления маршрута с команду надо добавить строку:

172.168.0.0 255.255.255.0 172.168.100.2 eth0

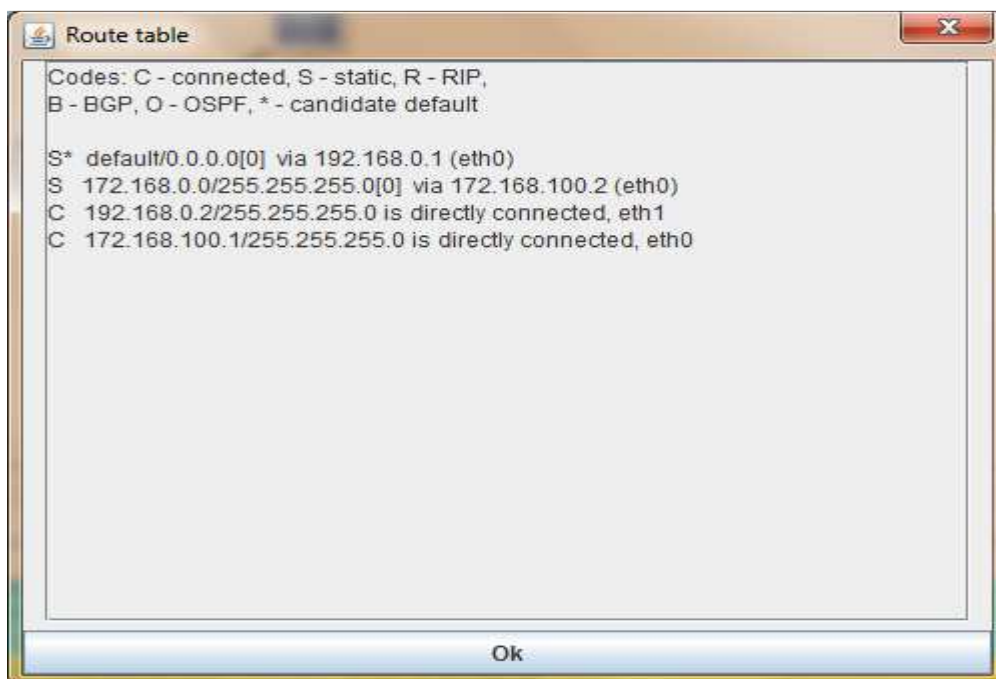


```
R# configure terminal
Enter configuration commands, one per line. End with 'exit'
R(config)# ip route ?
(<host ip>|<network ip>) <netmask> <gateway> [<target interface>]

R(config)# ip route 172.168.0.0 255.255.255.0 172.168.100.2 eth0
Route added.

R(config)#
```

Вид таблицы маршрутизации после добавления нового маршрута:



```
Codes: C - connected, S - static, R - RIP,
B - BGP, O - OSPF, * - candidate default

S* default/0.0.0.0[0] via 192.168.0.1 (eth0)
S 172.168.0.0/255.255.255.0[0] via 172.168.100.2 (eth0)
C 192.168.0.2/255.255.255.0 is directly connected, eth1
C 172.168.100.1/255.255.255.0 is directly connected, eth0
```

Ok

Результат выполнения утилиты «Ping»с адресом 172.168.0.2

The screenshot shows the javaNetSim interface with a network topology. PC1 is connected to Hub1, which is connected to R1. R1 is connected to R, which is connected to R2. R2 is connected to PC2. The console log shows the following sequence of events:

Time	Node	Packet	Layer	Info
14.03.52-530	R	ICMP_packet	Network	ProtocolStack received packet from local Interface
14.03.52-530	R	ICMP_packet	Network	Packet Received: Network Layer Device is Routable forwarding packet
14.03.52-530	R	ICMP_packet	Network	Forwarding packet from ProtocolStack(to 192.168.0.1)
14.03.52-530	R	Ethernet Packet	Link	Sending packet from interface A4:92:A2:BE:2E:62
14.03.52-530	R2	Ethernet Packet	Link	Received and accepted packet at interface 74:96:C4:94:86:C3
14.03.52-530	R2	ICMP_packet	Network	ProtocolStack received packet from local Interface
14.03.52-530	R2	ICMP_packet	Network	Packet Received: Network Layer Device is Routable forwarding packet
14.03.52-540	R2	ICMP_packet	Network	Forwarding packet from ProtocolStack(to 10.0.0.2)
14.03.52-540	R2	Ethernet Packet	Link	Sending packet from interface AE:5AB3:BC:2A:51
14.03.52-540	PC2	Ethernet Packet	Link	Received and accepted packet at interface A4:6B:84:9F:76:4E
14.03.52-540	PC2	ICMP_packet	Network	ProtocolStack received packet from local Interface
14.03.52-540	PC2	ICMP_packet	Network	Confirmed Packet is for this Network Layer Device
14.03.52-540	PC2	Echo Reply Packet	Network	Echo reply packet received from 172.168.0.2

Результат выполнения утилиты «Ping»с адресом 10.0.0.2

The screenshot shows the javaNetSim interface with the same network topology as above. The console log shows the following sequence of events:

Time	Node	Packet	Layer	Info
13.49.00-280	R	ICMP_packet	Network	ProtocolStack received packet from local Interface
13.49.00-280	R	ICMP_packet	Network	Packet Received: Network Layer Device is Routable forwarding packet
13.49.00-280	R	ICMP_packet	Network	Forwarding packet from ProtocolStack(to 172.168.0.2)
13.49.00-280	R	Ethernet Packet	Link	Sending packet from interface AA:30:98:8E:C3:83
13.49.00-280	R1	Ethernet Packet	Link	Received and accepted packet at interface A9:9F:44:30:6B:9A
13.49.00-280	R1	ICMP_packet	Network	ProtocolStack received packet from local Interface
13.49.00-280	R1	ICMP_packet	Network	Packet Received: Network Layer Device is Routable forwarding packet
13.49.00-280	R1	ICMP_packet	Network	Forwarding packet from ProtocolStack(to 172.168.0.2)
13.49.00-280	R1	Ethernet Packet	Link	Sending packet from interface 26:1C:40:55:5E:C6
13.49.00-280	PC1	Ethernet Packet	Link	Received and accepted packet at interface A9:9D:39:91:2F:91
13.49.00-280	PC1	ICMP_packet	Network	ProtocolStack received packet from local Interface
13.49.00-280	PC1	ICMP_packet	Network	Confirmed Packet is for this Network Layer Device
13.49.00-280	PC1	Echo Reply Packet	Network	Echo reply packet received from 10.0.0.2

ГЛОССАРИЙ

API Application Programming Interface. Набор функций, используемых прикладным программистом, оформленных в виде библиотеки.

ARP Address Resolution Protocol. Протокол разрешения трансляции MAC-адресов на IP-адреса.

Ethernet Пакетная технология вычислительных сетей.

ICMP Internet Control Message Protocol. Протокол передачи команд и сообщений об ошибках.

IP Internet Protocol. Основной протокол стека TCP/IP, функционирующий на уровне межсетевого взаимодействия.

IP-сокеты Номер порта в совокупности с номером сети и номером конечного узла.

javaNetSim Программный эмулятор работы сети, позволяющий работать с сетью на всех четырех уровнях модели стека протоколов TCP/IP.

MIB (Management Information Base) База данных информации для управления сетью, в которой определен стандартный набор информации, дающий возможность SNMP менеджеру собирать информацию о сетевом устройстве.

NIC Network Information Center. Информационный центр Internet, занимающийся распределением доменных имен первого уровня.

OSI Open System Interconnection. Стандарт открытого взаимодействия систем.

RARP Reverse Address Resolution Protocol. Протокол обратного разрешения MAC-адресов.

RFC Request For Comments. Формат документов для публикации стандартов по технологиям Internet.

SNMP (Simple Network Management Protocol) Простой протокол управления сетью, предназначенный для наблюдения и управления сетевыми устройствами работающими на уровне приложений.

SNMP агент Программное обеспечение сетевых устройств, которое на каждый запрос SNMP менеджера формирует отклик, содержащий результат обработки запроса.

SNMP менеджер Приложение управляющее сетью, посредством посылки запросов SNMP агенту.

TCP Transmission Control Protocol. Протокол стека TCP/IP, осуществляющий функции контроля за передачей данных. Функционирует на основном уровне.

TCP-Сегмент Часть информации, поступающей по протоколу TCP в рамках соединения от протоколов более высокого уровня, сформированная для передачи на сетевой уровень.

TELNET Сетевой протокол типа "клиент-сервер" для обеспечения незащищенного удаленного доступа к сетевому устройству с помощью командного интерпретатора.

TFTP Trivial File Transfer Protocol. Простой протокол передачи файлов.

TTL Time-To-Live. Время жизни IP-пакета.

UDP User Datagram Protocol. Протокол передачи пакетов пользователя.

X.25 Стандарт, описывающий канальный, сетевой и физический уровни OSI.

Демультимплексирование Процедура распределения протоколом поступающих от сетевого уровня пакетов между набором высокоуровневых служб, идентифицированных номерами портов.

Коммутатор (switch) Устройство, предназначенное для соединения нескольких сегментов компьютерной сети.

Концентратор (hub) Устройство для объединения нескольких устройств Ethernet в общий сегмент.

Маршрут Последовательность промежуточных узлов, которые проходит IP-пакет в процессе маршрутизации при движении от отправителя к месту назначения.

Маршрутизация Процесс определения маршрута следования пакетов данных в вычислительных сетях.

Маршрутизация динамическая Метод маршрутизации осуществляемый с помощью протоколов маршрутизации.

Маршрутизация статическая Метод маршрутизации осуществляемый на основе таблиц маршрутизации.

Маршрутизатор Сетевое устройство, используемое в компьютерных сетях передачи данных, которое, на основании информации о топологии сети (таблицы маршрутизации) и определенных правил, принимает решения о пересылке пакетов сетевого уровня их получателю.

Мультиплексирование Процедура обслуживания протоколом запросов, поступающих от нескольких различных прикладных служб.

Объект MIB Порция информации, существующая в базе данных MIB.

Порт Очередь, организуемая операционной системой, к точке входа прикладного процесса.

Протокол Совокупность синтаксических правил, определяющая взаимодействие двух узлов вычислительной сети.

Сетевой уровень (internetworking layer) Уровень модели TCP/IP, предназначенный для образования единой транспортной системы, объединяющей несколько сетей, причем эти сети могут использовать совершенно различные принципы передачи сообщений между конечными узлами и обладать произвольной топологией.

Сообщение (message) Единица данных при взаимодействии клиента и сервера посредством протокола прикладного уровня.

Таблица маршрутизации Специальная информационная структура, используемая для определения маршрута следования пакета по адресу его сети назначения.

Уровень приложений Набор сетевых служб, которые предоставляет система сетевым пользовательским приложениям. Уровень приложений обеспечивает набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам, преобразование структур данных для пересылки их по сети и поддержку сеансов связи.

Управление сетью Процесс управления отказами, контроля конфигураций, мониторинга производительности, обеспечения защиты и учета деятельности в сети передачи данных.