Федеральное агенство по образованию РФ

Санкт-Петербургский государственный электротехнический университет "ЛЭТИ"

ИССЛЕДОВАНИЕ УРОВНЕЙ ОРГАНИЗАЦИИ IP-СЕТЕЙ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Санкт-Петербург
Издательство СПбГЭТУ "ЛЭТИ"
2006

УДК 004.4:004.7
ББК ?????.??
А 47

Исследование уровней организации IP-сетей. Лабораторный практикум / Алекперов И.А. Большев А.К. Карпов К.Э. Кринкин К.В. Яновский В.В. СПб.: Изд-во СПбГЭТУ "ЛЭТИ", 2006. 80 с.
А 47
ISBN 5-????-????-?

Содержит теоретические сведения о структуре IP-сетей. Рассмотрены основные уровни организации стека протоколов TCP/IP: сетевой, транспортный, прикладной. Изложены принципы адресации в IP-сетях, механизмы статической маршрутизации. Уделено внимание протоколу ICMP и его приложениям. Представлен сравнительный анализ производительности протоколов транспортного уровня: TCP и UDP. На примере SNMP и TELNET показаны общие принципы функционирования сервисов прикладного уровня.

Содержит практические задания по моделированию и анализу IP-сетей на разных уровнях в среде сетевого имитатора javaNetSim, разработанного на кафедре МОЭВМ, а также задания по программированию простейших приложений на основе Windows ICMP API.

Предназначен для студентов специальностей 220400 "Программное обеспечение вычислительной техники и автоматизированных систем", по направлениям 510200 "Прикладная математика и информатика" и 552800 "Информатика и вычислительная техника", изучающих дисциплины "Современные компьютерные технологии", "Сетевые технологии", "Сети ЭВМ и телекоммуникации".

УДК 004.4:004.7
ББК ?????.??

Рецензенты: кафедра цифровой вычислительной техники и информатики Санкт-Петербургского государственного университета телекоммуникаций; д-р техн. наук, проф. Н. В. Егоров (СПбГУ).

Утверждено
редакционно-издательским советом университета
в качестве учебного пособия

ISBN 5-?????-????-?

ВВЕДЕНИЕ

Компьютерные сети и коммуникационные технологии являются одной из ключевых областей, знания в которой необходимы специалистам, занимающимся разработкой современных информационных систем и комплексов. Для того, чтобы лучше понимать организацию вычислительных сетей удобно использовать модель глобальной сети, часто называемую TCP/IP моделью. Каждый из уровней этой модели имеет собственные протоколы. Актуальность этой модели связана прежде всего с распространением и развитием технологии Internet, базирующейся на стеке TCP/IP, а также с большим соответствием процесса межсетевого взаимодействия.

Предлагаемый лабораторный практикум нацелен на изложение основных принципов организации IP- сетей на сетевом, транспортном и прикладном уровнях. Он ориентирован на студентов специальностей 220400 "Программное обеспечение вычислительной техники и автоматизированных систем", по направлениям 510200 "Прикладная математика и информатика" и 552800 "Информатика и вычислительная техника", изучающих дисциплины "Современные компьютерные технологии", "Сетевые технологии", "Сети ЭВМ и телекоммуникации".

Структурно, лабораторный практикум разделен на три части. Первая часть (главы 1–5) является основной. Она посвящена отдельным аспектам уровней стека протоколов TCP/IP. Во второй части (глава 2) дается описание имитатора сетевых технологий, разработанного на кафедре МОЭВМ специально для проведения лабораторных работ. Третья часть практикума – справочная и включает себя глоссарий и список литературы, рекомендуемой для более глубокого освоения предмета.

Каждая глава основной части практикума содержит теоретический материал, необходимый для освоения предмета, задания для самостоятельного выполнения в лаборатории, а также список вопросов для проверки полученных знаний. Лабораторные работы в главах 2, 3 и 5, 6 разработаны студентами 4 курса, изучавшими дисциплину "Сетевые технологии".

Сетевой уровень модели TCP/IP представлен главами 2–4. В них рассматриваются принципы адресации, в том числе отображение IP-адресов в адреса физического уровня. Освещаются вопросы статической маршрутизации и управления маршрутами, а также рассматривается протокол передачи служебных сообщений ICMP. Практические занятия охватывают моделирование алгоритмов преобразования адресов и маршрутизации, а также основы

программирования протокола ICMP в ОС Windows.

Транспортный уровень стека TCP/IP рассматривается в главе 5. Теоретический материал включает описание алгоритмов управления пакетной и поточной передачей данных. Приведен сравнительный анализ производительности протоколов UDP и TCP. В качестве практических заданий студентам предлагается на основе имитационного моделирования для заданных конфигураций сетей обосновать применение того или иного транспортного протокола.

Прикладной уровень (глава 6) рассматривается на примере протоколов SNMP и TELNET. Студенты в рамках практических занятий узнают принципы функционирования прикладных сервисов и знакомятся с их возможностями.

Для того, чтобы наглядно показать связи между уровнями модели авторами был разработан эмулятор javaNetSim, обеспечивающий иллюстрацию связей уровней между собой. В качестве исходного варианта использована разработка Канберрского технического университета JFirewallSim, распространяемая по лицензии BSD. Функциональность исходного варианта расширяется за счет введения средств иллюстрации уровней приложений, транспортного и сетевого. На сетевом уровне введены функции безклассовой адресации статической маршрутизации и работы с протоколом ARP. Необходимость доработок исходного варианта вызвана отсутствием доступных эмуляторов, отвечающих поставленным требованиям: минимальным временным затратам на обучение, комплексным требованиям к аппаратным платформам, инвариантностью к операционным системам. Малые объемы требуемой памяти, хорошие временные показатели, многоплатформенность, наглядный графический интерфейс, доступность javaNetSim обеспечивают ему преимущества по сравнению с другими эмуляторами.

1. СЕТЕВОЙ УРОВЕНЬ: IP-АДРЕСАЦИЯ

Сетевой уровень (межсетевой уровень) модели TCP/IP служит 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 [8]. На этом уровне термин сеть означает совокупность компьютеров, соединенных между собой в соотвествии с некой топологией и использующих физический уровень модели TCP/IP для передачи данных. Единицей данных сетевого уровня является пакет. На этом уровне определяются два вида протоколов – сетевые протоколы и протоколы маршрутизации. Первые реализуют продвижение пакетов

через сеть. Это такие протоколы как IP, ICMP, ARP и другие. Вторые – предоставляют способы обмена информацией о маршрутах. Кроме того, сетевой уровень TCP/IP, с помощью средств IP-адресации, решает важную задачу идентификации узла-получателя пакета [5].

1.1. Типы сетевых адресов

Каждый компьютер в сети TCP/IP имеет адреса двух типов:

- локальный (физический) адрес узла, определяемый технологией, с помощью которой построена отдельная сеть, в которую входит данный узел. Для узлов, входящих в локальные сети – это MAC-адрес сетевого адаптера или порта маршрутизатора. Эти адреса назначаются производителями оборудования и являются уникальными адресами, так как управляются централизовано. Для всех существующих технологий локальных сетей MAC-адрес имеет формат 6 байтов: старшие 3 байта – идентификатор фирмы производителя, а младшие 3 байта назначаются уникальным образом самим производителем. Для узлов, входящих в глобальные сети, такие как X.25 или frame relay, физический адрес назначается администратором глобальной сети. Пример физического адреса: 44-BC-89-A2-FE-00;
- IP-адрес, состоящий из 4 байт. Этот адрес используется на сетевом уровне. Он назначается администратором во время конфигурирования компьютеров и маршрутизаторов. IP-адрес состоит из двух частей: номера сети и номера узла. Номер сети определяет конкретную физическую сеть. Номер узла определяет конкретную рабочую станцию, сервер и пр., включенную в сеть. *Подсеть* – это физический сегмент TCP/IP сети, в котором используется IP-адреса с общим номером сети [1]. Пример IP-адреса: 172.168.10.15.

Номер узла в протоколе IP назначается независимо от физического адреса узла. Деление IP-адреса на поле номера сети и номера узла – гибкое, и граница между этими полями может устанавливаться произвольно. Узел может входить в несколько IP-сетей. В этом случае узел должен иметь несколько IP-адресов, по числу сетевых связей. Таким образом IP-адрес характеризует не отдельный компьютер или маршрутизатор, а один сетевой интерфейс (физический или виртуальный). IP-адрес имеет длину 4 байта и обычно записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме, и разделенных точками, например:
10.10.1.4 – традиционная десятичная форма представления адреса,

00001010 00001010 00000001 00000100 – двоичная форма представления этого же адреса.

1.2. Структура IP-адреса

Какая часть IP-адреса относится к номеру сети, а какая к номеру узла, определяется двумя способами: с помощью классов (классовая адресация) или с помощью масок подсети (бесклассовая адресация).

В классовой адресации номер сети и номер узла определяются по принадлежности IP-адреса одному из классов адресов: A,B,C,D или E. Класс определяется значениями первых битов адреса:

- если адрес начинается с 0, то сеть относят к классу A, и номер сети занимает один байт, остальные 3 байта интерпретируются как номер узла в сети. Сети класса A имеют номера в диапазоне от 1 до 126. (Номер 0 не используется, а номер 127 зарезервирован для специальных целей);
- если первые два бита адреса равны 10, то сеть относится к классу B. В сетях класса B под адрес сети и под адрес узла отводится по 16 битов, то есть по 2 байта;
- если адрес начинается с последовательности 110, то это сеть класса C. Под адрес сети отводится 24 бита, а под адрес узла – 8 битов;
- если адрес начинается с последовательности 1110, то он является адресом класса D и обозначает особый, групповой адрес. Если в пакете в качестве адреса назначения указан адрес класса D, то такой пакет должны получить все узлы, которым присвоен данный адрес;
- если адрес начинается с последовательности 11110, то это адрес класса E, он зарезервирован для будущих применений.

В бесклассовой адресации номер сети к которому принадлежит узел с заданным IP-адресом определяется другим способом: вместе с IP-адресом нам предоставляется *маска подсети*. В терминологии сетей TCP/IP маской подсети или маской сети называется битовая маска, определяющая, какая часть IP-адреса узла сети относится к адресу сети, а какая к адресу самого узла в этой сети. Например, узел с IP-адресом 192.168.0.1 и маской подсети 255.255.255.0 находится в сети 192.168.0.0. Чтобы получить адрес сети, зная IP-адрес и маску подсети, необходимо применить к ним операцию поразрядной конъюнкции. Например:

```
IP-адрес:       00001010 00001010 00000001 00000100 (10.10.1.4)
Маска подсети:  11111111 00000000 00000000 00000000 (255.0.0.0)
Адрес сети:     00001010 00000000 00000000 00000000 (10.0.0.0)
```

Для стандартных классов сетей маски имеют следующие значения:

- 255.0.0.0 – маска для сети класса A;
- 255.255.0.0 – маска для сети класса B;
- 255.255.255.0 – маска для сети класса C.

1.3. Отображение физических адресов на логические

Отображение физических адресов на IP-адреса происходит с помощью протокола *ARP*. Функционирование ARP происходит различным образом, в зависимости от того, какой протокол канального уровня работает в данной сети. В локальных сетях протокол ARP использует широковещательные кадры протокола канального уровня для поиска в сети узла с заданным IP-адресом. Узел, которому нужно выполнить отображение IP-адреса на локальный адрес, формирует ARP запрос, вкладывает его в кадр протокола канального уровня, указывая в нем известный IP-адрес, и осуществляет его широковещательную рассылку по сети. Все узлы локальной сети получают ARP запрос и сравнивают указанный там IP-адрес с собственным. В случае их совпадения узел формирует ARP-ответ, в котором указывает свои IP- и локальный- адреса. ARP-запросы и ответы используют один и тот же формат пакета. Так как локальные адреса могут в различных типах сетей иметь различную длину, то формат пакета протокола ARP зависит от типа сети. Для того, чтобы не перегружать сеть запросами, ARP использует таблицу отображения (так называемый ARP-кэш). Эта таблица содержит три поля – IP-адрес, соответствующий ему MAC-адрес и тип. Тип может быть статическим или динамическим. Запись в таблице имеет динамический тип, если она внесена в таблицу путем широковещательного запроса ARP. Такие записи имеют время устаревания (обычно 180 или 360 секунд), после истечения которого они удаляются из таблицы. Запись в таблице будет иметь статический тип, если она добавлена вручную (например, с помощью команды arp в ОС Windows или Unix). Статическая запись имеет неограниченное время устаревания.

1.4. Маршрутизация по умолчанию

Для объединения подсетей в единую сеть в простейшем случае используется маршрутизация по умолчанию. Она организуется посредством шлюзов. Шлюзом будем называть узел внутри подсети, который предоставляет доступ в другую подсеть [5]. Чаще всего в виде шлюза выступает маршрутизатор. Схема такой маршрутизации выглядит следующим образом: задан адрес *шлюза по умолчанию*. При

попытке отправки пакета в сеть, узел проверяет совпадение подсети назначения пакета с подсетью узла. Если подсети разные, то пакет отправляется на шлюз. В простейшем случае, шлюз сравнивает сеть IP-адреса назначения с номерами сетей на своих интерфейсах и в случае совпадения, направляет пакет узлу назначения через этот сетевой интерфейс. В противном случае он отправляет пакет узлу, указанному в качестве шлюза по умолчанию на самом шлюзе. Если такового нет, то пакет теряется.

1.5. Протокол ICMP

Для проверки соединений и корректного функционирования сети обычно используется *протокол ICMP* . ICMP – Internet Control Message Protocol, протокол управляющих сообщений интернета. ICMP – протокол сетевого уровня и работает поверх протокола IP. Он предназначен для обмена информацией об ошибках между маршрутизаторами (шлюзами) сети и узлом-источником пакета [8].С помощью специальных пакетов этот протокол сообщает о невозмодности доставки пакета, превышении времени жизни, аномальных значениях параметров, изменении маршрта пересылки, состоянии системы и т. п. В простейшем случае, для проверки работоспособности сети используются два сообщения ICMP: эхо-запрос (Echo request) и эхо-ответ (Echo reply). Когда на узел приходит сообщение ICMP типа эхо-запрос, он отправляет сообщение эхо-ответ на тот узел, с которого пришел запрос. Пример реализации такого обмена представлен в утилите ping, входящей в состав почти любой сетевой ОС. Подробнее протокол ICMP будет рассмотрен в главе 3.

1.6. Лабораторная работа 1

Цель: *изучение и практическое освоение основ адресации, разрешения физических адресов и простейшей маршрутизации в IP-сетях.*

1.6.1. Порядок выполнения работы

1. Исправить структуру сети (если это необходимо), обеспечив корректную доставку кадров на **физическом** уровне.
2. Задать ip-адреса, маски подсети и шлюзы по-умолчанию для всех узлов сети, чтобы обеспечить корректную доставку эхо-запроса от K1 к K2 и эхо-ответа обратно. Обосновать свои установки.
3. Выполнить эхо-запрос с K1 на K2. Посмотреть вывод программы.

4. Добавить статическую запись ARP для К3 на К1 (или для ближайшего к К1 маршрутизатора, находящегося между К3 и К1). Подождать устаревания ARP-таблиц и выполнить эхо запрос с К1 на К3. Объяснить результат.
5. Выполнить эхо-запрос на IP-адрес 200.100.0.1 с К1. Объяснить вывод программы.
6. Выполнить эхо запросы с К1 и К2 на все узлы сети. Убедиться, что эхо-ответы приходят.

В отчет необходимо включить схему сети, настройки протокола TCP/IP для все узлов сети и результаты вывода программы полученные при выполнении при эхо-запросов.

1.6.2. Варианты заданий

Вариант 1. Файл со схемой сети: lab1_var1.jfst. Сеть между маршрутизаторами R1,R2 и Boss_R: 117.168.0.0. Компьютер Boss имеет IP-адрес 64.2.0.1. Компьютер Hacker имеет IP-адрес 117.168.0.5.
Обозначения в задании: К1 – Boss, К2 – Hacker, К3 – OFFICE2 pc1.

Вариант 2. Файл со схемой сети: lab1_var2.jfst. Сеть между маршрутизаторами OFF_R и R2: 136.15.0.0. Компьютер BIG BOSS имеет IP-адрес 136.15.32.1. Компьютер M_CH_S имеет IP-адрес 10.10.0.2. Сеть между маршрутизаторами R2 и M_CH_S_Router: 192.178.0.0.
Обозначения в задании: К1 – BIG BOSS, К2 – M_CH_S, К3 – OFFICE1_pc4.

Вариант 3. Файл со схемой сети: lab1_var3.jfst. Сеть между маршрутизаторами R1,R2 и Boss_R: 172.198.0.0. Компьютер Boss имеет IP-адрес 10.2.0.1. Компьютер Hacker имеет IP-адрес 172.198.99.252.
Обозначения в задании: К1 – Boss, К2 – Hacker, К3 – OFFICE2_pc1.

Вариант 4. Файл со схемой сети: lab1_var4.jfst. Сеть между маршрутизаторами OFF_R и R2: 204.188.0.0. Компьютер BIG BOSS имеет IP-адрес 204.188.0.1. Компьютер M_CH_S имеет IP-адрес 10.0.0.2. Сеть между маршрутизаторами R2 и M_CH_S_Router: 192.178.0.0.
Обозначения в задании: К1 – BIG BOSS, К2 – M_CH_S, К3 – OFFICE1_pc4.

Вариант 5. Файл со схемой сети: lab1_var5.jfst. Сеть между маршрутизаторами RServers, RManagers и RBosses: 10.0.0.0. Компьютер MegaBoss имеет IP-адрес 172.16.0.5. Компьютер Manager2 имеет IP-адрес 172.16.1.12. Компьютер FileServer имеет IP-адрес 172.16.10.10.
Обозначения в задании: К1 – MegaBoss, К2 – Manager2, К3 – FileServer.

Вариант 6. Файл со схемой сети: lab1_var6.jfst. Сеть между маршрутизаторами RServers, RManagers и RBosses: 192.168.0.0.

Компьютер MicroBoss имеет IP-адрес 10.0.1.5. Компьютер Manager3 имеет IP-адрес 10.0.2.5. Компьютер PrintServer имеет IP-адрес 10.0.64.1.
Обозначения в задании: K1 – Manager3, K2 – PrintServer, K3 – Micro-Boss.

Вариант 7. Файл со схемой сети: lab1_var7.jfst. Сеть между маршрутизаторами R1 и ADSL: 172.168.0.0. Компьютер Station1 имеет IP-адрес 172.168.1.2. Компьютер Remote1 имеет IP-адрес 10.0.0.110. Сеть между маршрутизаторами ADSL и ADSL2: 192.168.0.0.
Обозначения в задании: K1 – Station1, K2 – Remote1, K3 – Station2.

Вариант 8. Файл со схемой сети: lab1_var8.jfst. Сеть между маршрутизаторами R1 и ADSL: 192.168.0.0. Компьютер Station1 имеет IP-адрес 192.168.1.2. Компьютер Remote1 имеет IP-адрес 99.11.0.11. Сеть между маршрутизаторами ADSL и ADSL2: 172.168.0.0.
Обозначения в задании: K1 – Station1, K2 – Remote1, K3 – Station2.

Вариант 9. Файл со схемой сети: lab1_var9.jfst. Сеть между маршрутизаторами R1 и R2: 192.168.100.0. Компьютер PC1 имеет IP-адрес 129.64.128.1. Компьютер PC2 имеет IP-адрес 129.64.127.254. Компьютер PC4 имеет IP-адрес: 10.0.0.2. Длина маски подсети (количество значащих единиц) на PC1, PC2, PC3 должно быть минимально возможным (обеспечивая при этом корректную работу).
Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.

Вариант 10. Файл со схемой сети: lab1_var10.jfst. Сеть между маршрутизаторами R1 и R2: 192.168.0.0. Компьютер PC1 имеет IP-адрес 172.168.0.1. Компьютер PC2 имеет IP-адрес 172.168.0.65. Компьютер PC4 имеет IP-адрес: 1.0.0.2.
Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.

Вариант 11. Файл со схемой сети: lab1_var11.jfst. Сеть между маршрутизаторами R-C-M и R-S-C: 10.1.0.0. Сеть между маршрутизаторами R-C-M и R-M-S: 10.0.32.0. Сеть между маршрутизаторами R-M-S и R-S-C: 10.0.0.128. Компьютер Chief имеет IP-адрес 10.1.0.3. Компьютер Manager1 имеет IP-адрес 10.0.32.11. Компьютер Service имеет IP-адрес: 10.0.0.135.
Обозначения в задании: K1 – Chief, K2 – Manager1, K3 – Service.

Вариант 12. Файл со схемой сети: lab1_var12.jfst. Сеть между маршрутизаторами R-C-M и R-S-C: 172.168.128.0. Сеть между маршрутизаторами R-C-M и R-M-S: 172.168.1.0. Сеть между маршрутизаторами R-M-S и R-S-C: 172.168.0.64. Компьютер Chief имеет IP-адрес 172.168.128.5. Компьютер Manager3 имеет IP-адрес 172.168.1.13. Компьютер Service имеет IP-адрес: 172.168.0.76.
Обозначения в задании: K1 – Manager3, K2 – Service, K3 – Chief.

Вариант 13. Файл со схемой сети: lab1_var13.jfst. Сеть между маршрутизаторами R120, R230 и R232: 172.31.128.0. Сеть между маршрутизаторами R232 и R233: 10.10.0.0. Компьютер Remote1 имеет IP-адрес 172.31.127.0. Компьютер Remote2 имеет IP-адрес 172.31.200.1. Компьютер Remote3 имеет IP-адрес: 10.0.39.0.
Обозначения в задании: K1 – Remote1, K2 – Remote2, K3 – Remote3.
Вариант 14. Файл со схемой сети: lab1_var14.jfst. Сеть между маршрутизаторами R120, R230 и R232: 63.12.95.0. Сеть между маршрутизаторами R232 и R233: 63.12.225.0. Компьютер Remote1 имеет IP-адрес 168.20.88.0. Компьютер Remote2 имеет IP-адрес 63.12.95.1. Компьютер Remote3 имеет IP-адрес: 168.20.120.0.
Обозначения в задании: K1 – Remote2, K2 – Remote3, K3 – Remote1.

1.6.3. Пример выполнения лабораторной работы

Рассмотрим конфигурацию сети, приведенную на рисунке 1.1. Файл со схемой сети: lab1_sample.jfst. Сеть между маршрутизаторами R1 и R2: 172.168.100.0. Компьютер PC1 имеет IP-адрес 172.168.0.2. Компьютер PC2 имеет IP-адрес 10.0.0.2.

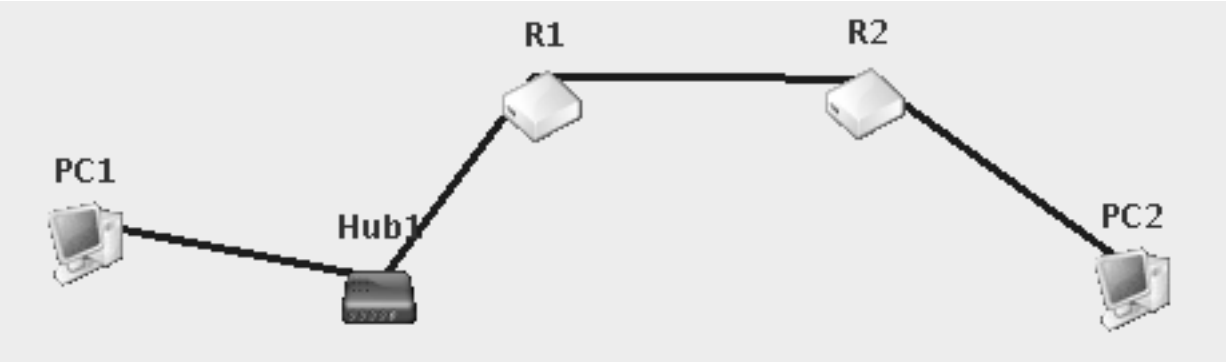


Рис. 1.1 Схема сети

Задание:

1. Задать маски подсети и шлюзы по умолчанию для PC1 и PC2, а также IP-адреса из заданного диапазона вместе с масками и шлюзами для R1 и R2 так, чтобы обеспечить корректную доставку эхо-запроса от PC1 к PC2 и эхо-ответа обратно. Обосновать свои установки.
2. Выполнить эхо-запрос с PC1 на PC2. Проанализировать вывод программы.
3. Выполнить эхо-запрос на IP-адрес 192.168.0.1 с PC1. Объяснить вывод программы.

Порядок выполнения будет следующим:

1. Зададим IP-адреса и маски подсети для маршрутизаторов R1 и R2. Как видно, сети 172.168.100.0 и 172.168.0.0 (если использовать стандартную маску подсети для класса B) эквивалентны. Будем использовать маску подсети, отличную от стандартной. Зададим для маршрутизатора R1 на интерфейсе eth0 адрес 172.168.0.1 и маску подсети 255.255.255.0. На интерфейсе eth1 установим адрес 172.168.100.1 и маску 255.255.255.0. Теперь необходимо сконфигурировать маршрутизатор R2. На его интерфейсе eth0 зададим IP 172.168.100.2 и маску 255.255.255.0. Установим шлюз по умолчанию в 172.168.100.1. На интерфейсе eth1 для R2 установим любой IP-адрес из диапазона сети PC2, например 10.0.0.1 и соответствующую ему маску: 255.0.0.0. Для корректной маршрутизации осталось задать только шлюз по умолчанию для R1. Он будет адресом маршрутизатора R2.
2. Теперь настроим конечные узлы. На PC1 зададим маску подсети соответствующую новому адресному пространству – 255.255.255.0. Так как пакеты от узла PC1 в другие сети должны проходить через маршрутизатор R1, зададим шлюз по умолчанию 172.168.0.1 (адрес R1). Аналогичные операции проведем на PC2 – установим маску подсети в 255.0.0.0, а шлюз по умолчанию в 10.0.0.1. Стоит заметить, что приведенная конфигурация не является единственно верной.
3. После отправки эхо-запроса с PC1 на PC2 в консоли будет выведен результат прохождения запроса и ответа на него по сети:
```
PC1  Created Echo Request packet to 10.0.0.2
PC1  Created ARP discovery packet to source MAC address.
     for IP 172.168.0.1
PC1  Sending broadcast packet from ProtocolStack.
...
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 видно, PC1 успешно получил эхо-ответ на свой запрос к PC2.
4. Выполним эхо-запрос для несуществующего узла с IP-адресом 192.168.0.1. Для этого выполним на PC1 последовательность действий, аналогичную предыдущему пункту, вместо адреса 10.0.0.2 используя адрес 192.168.0.2:
```
PC1  Created Echo Request packet to 192.168.0.1
PC1  Sending packet from ProtocolStack (to 172.168.0.1).
...
```

```
R1   ProtocolStack received packet from local Interface.
R1   Packet Received: Network Layer Device is
     Routable forwarding packet.
R1   Forwarding packet from ProtocolStack
     (to 172.168.100.1).
R2   ProtocolStack received packet from local Interface.
R2   Packet Dropped: Hop count exceeded.
Host 192.168.0.2 Unreachable
```

Как видно, пакет попал в "петлю" между двумя маршрутизаторами и находился там, пока у него не закончилось время жизни (TTL).

1.6.4. Контрольные вопросы

1. Что такое кэш ARP? Какие типы записей могут содержаться в кэше ARP?
2. Какому классу IP-адресов принадлежат адреса 10.11.0.1, 127.1.1.1?
3. Разделите адресное пространство 192.168.1.0 на 4 подсети при помощи масок.
4. Что такое концентратор? Объясните принцип работы концентратора. Чем концентратор отличается от повторителя?
5. Что такое шлюз?
6. Для чего предназначен протокол ICMP?

2. СЕТЕВОЙ УРОВЕНЬ: СТАТИЧЕСКАЯ МАРШРУТИЗАЦИЯ

Маршрутизация – важнейший процесс в IP-сетях. Чтобы некоторый узел мог найти в сети другой, должен быть определен механизм описания того, как достичь от произвольного узла к другому узлу. Такой механизм и называется маршрутизацией. Более строгое определение звучит так: *Маршрутизация* – процесс определения маршрута следования пакетов данных в компьютерных сетях. Маршрутизация выполняется специальными программными или аппаратными средствами – маршрутизаторами. *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. Маршрутизация, осуществляемая IP – это процесс поиска в таблице маршрутизации, определение интерфейса, куда будет послан пакет. Существует два типа маршрутизации:

- Статическая
- Динамическая

Статическая маршрутизация осуществляется на основе таблиц маршрутизации, задаваемых администратором. Динамическая маршрутизация осуществляется с помощью протоколов маршрутизации. Протокол маршрутизации это сетевой протокол, используемый маршрутизаторами для определения возможных маршрутов следования данных в составной компьютерной сети. Применение протокола маршрутизации позволяет избежать ручного ввода всех допустимых маршрутов, что, в свою очередь, снижает количество ошибок, обеспечивает согласованность действий всех маршрутизаторов в сети и облегчает труд администраторов.

2.1. Принцип статической маршрутизации

В этой главе будет рассмотрена статическая маршрутизация в IP-сетях. Частично, эта тема уже затрагивалась в главе 1, когда рассматривалась простейшая маршрутизация на основе шлюзов по умолчанию. Как правило, маршрутизатор имеет несколько интерфейсов и может одновременно находится в нескольких подсетях. Таблица маршрутизации содержит информацию, на основе которой маршрутизатор принимает решение о дальнейшей пересылке пакетов. Как только маршрутизатор (или на любой узел сети, поддерживающем сетевой уровень модели TCP/IP) получает IP-пакет, выполняется следующая последовательность действий [1]:

1. Маршрутизатор определяет, является ли узел-получатель пакета локальным (т.е. находится ли он в той же подсети, что маршрутизатор). Если получатель находится в той же подсети, то пакет направляется ему напрямую.
2. Если узел-получатель находится в другой подсети, то просматривается таблица маршрутизации на предмет пути к этому узлу. При просмотре, сеть узла-получателя сравнивается с записями о сетях в таблице маршрутизации и при обнаружении совпадения пакет направляется маршрутизатору, указанному в соответствующей записи.
3. Если маршрут в таблице маршрутизации не найден, то пакет отправляется шлюзу по умолчанию, указанному на маршрутизаторе.
4. Если запись о шлюзе по умолчанию отсутствует, то пакет уничтожается.

Для указания шлюза по умолчанию, в таблице существует специальная запись default. Запись default указывает, на какой узел должен быть направлен пакет, если совпадений его места назначения в таблице не

найдено.

2.2. Таблицы маршрутизации

Чтобы по адресу сети назначения можно было бы выбрать маршрут дальнейшей пересылки пакета, каждый узел, поддерживающий сетевой уровень анализирует специальную информационную структуру, называемую *таблицей маршрутизации* [7]. Простейшая таблица маршрутизации включает в себя информацию об узле(или сети) назначения, маске подсети для узла(сети) назначения, интерфейсе, через который следует направить пакет и шлюзе – удаленном узле, которому будет передан пакет. Рассмотрим, как может выглядеть таблица маршрутизации на примере. Пусть задана конфигурация сети показанная на рисунке 2.1.

Таблица для маршрутизатора R2 может выглядеть например так:

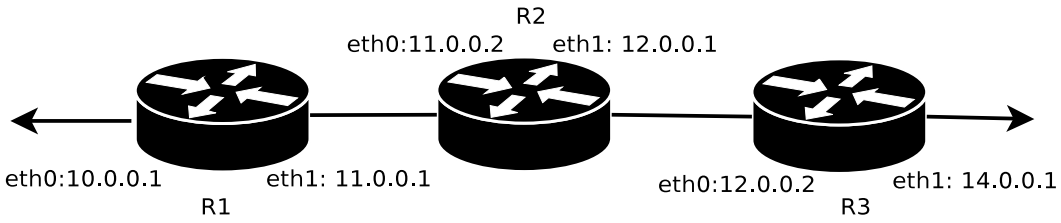


Рис. 2.1 Схема сети

Destination	Gateway	Genmask	Iface
default	11.0.0.1	255.0.0.0	eth0
14.0.0.0	12.0.0.2	255.0.0.0	eth1

В первой колонке таблице перечисляются номера подсетей, во второй – какому маршрутизатору следует перенаправить пакет для для отправки в заданную подсеть. Третья колонка задает маску подсети назначения, в четвертой указывается через какой интерфейс следует направить пакет.

2.3. Лабораторная работа 2

Цель: *изучить методы статической маршрутизации в IP-сетях. Научиться управлять таблицами маршрутизации на узлах сетевого уровня.*

2.3.1. Порядок выполнения работы

1. Для всех узлов сети установить IP-адреса, маски подсетей и шлюзы по-умолчанию, чтобы добиться успешного выполнения эхо-запроса ближайших соседей (находящихся в одной подсети).

2. Настроить таблицы маршрутизации на маршрутизаторах, чтобы добиться доставки пакетов от узла K1 к узлу K2 и обратно, от узла K2 к K3 и обратно, от узла K3 к K1 и обратно. Пакеты должны доходить до узлов кратчайшим путем
3. Настроить таблицы маршрутизации на узлах K1, K2 и K3 чтобы обеспечить кратчайшую доставку пакетов между этими узлами, если это невозможно было обеспечить в пункте 2.

В отчете привести конфигурацию TCP/IP для каждого из узлов, таблицы маршрутизации, результаты эхо-запросов между узлами K1, K2 и K3, а также обоснование правильности и оптимальности выбранных маршрутов.

2.3.2. Варианты заданий

Вариант 1. Файл со схемой сети: lab2_var1.jfst. Сеть между маршрутизаторами R1, R2 и R3: 192.168.3.0. Сеть между маршрутизаторами R3 и R4: 192.168.4.0. Сеть между маршрутизаторами R5 и R6: 192.168.5.0. Компьютер PC1 имеет IP-адрес 192.168.0.100. Компьютер PC3 имеет IP-адрес 192.168.1.100. Компьютер PC4 имеет IP-адрес: 192.168.2.100.

Обозначения в задании: K1 – PC1, K2 – PC3, K3 – PC4.

Вариант 2. Файл со схемой сети: lab2_var2.jfst. Сеть между маршрутизаторами R1, R2 и R3: 172.168.3.0. Сеть между маршрутизаторами R5 и R6: 172.168.4.0. Компьютер PC1 имеет IP-адрес 172.168.0.100. Компьютер PC3 имеет IP-адрес 172.168.1.100. Компьютер PC4 имеет IP-адрес: 172.168.2.100.

Обозначения в задании: K1 – PC1, K2 – PC3, K3 – PC4.

Вариант 3. Файл со схемой сети: lab2_var3.jfst. Сеть между маршрутизаторами R1, R2, R3 и R4: 192.168.0.96. Сеть между маршрутизаторами R4 и R5: 172.168.4.0. Маршрутизатор R6 имеет адрес 10.120.0.1 на первом интерфейсе и 10.159.0.1 на втором интерфейсе. Сеть между маршрутизаторами R3 и R8: 11.0.0.0. Компьютер PC1 имеет IP-адрес 192.168.0.4. Компьютер PC3 имеет IP-адрес 192.168.0.34. Компьютер PC4 имеет IP-адрес: 192.168.0.250.

Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.

Вариант 4. Файл со схемой сети: lab2_var4.jfst. Сеть между маршрутизаторами R1, R2, R3 и R4: 199.0.5.96. Сеть между маршрутизаторами R4 и R5: 172.168.4.0. Маршрутизатор R6 имеет адрес 11.120.0.1 на первом интерфейсе и 11.159.0.1 на втором интерфейсе. Сеть между маршрутизаторами R3 и R8: 12.0.0.0. Компьютер PC1 имеет IP-адрес 199.0.5.2. Компьютер PC3 имеет IP-адрес 199.0.5.52. Компьютер PC4 имеет IP-адрес: 199.0.5.250.

Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.

Вариант 5. Файл со схемой сети: lab2_var5.jfst. Сеть между узлами PC3 и R3, R4, R6: 204.188.45.128. Сеть между маршрутизаторами R1, R2, R3: 204.188.45.192. Компьютер PC1 имеет IP-адрес 204.188.45.1. Компьютер PC2 имеет IP-адрес 204.188.45.65. Компьютер PC3 имеет IP-адрес 204.188.45.129. Длина маски подсети должна быть минимально возможной.

Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.

Вариант 6. Файл со схемой сети: lab2_var6.jfst. Сеть между узлами R1, R2, R3: 192.115.120.0. Сеть между узлами R2, R3, R4: 192.115.112.0. Сеть между узлами R4, R5, R7: 192.115.108.0. Сеть между узлами R6 и R7: 192.115.96.0. Компьютер PC1 имеет IP-адрес 192.115.128.1. Компьютер PC2 имеет IP-адрес 192.115.100.1. Компьютер PC3 имеет IP-адрес 192.115.88.2. Длина маски подсети должна быть минимально возможной.

Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.

Вариант 7. Файл со схемой сети: lab2_var7.jfst. Сеть между узлами PC3 и R3, R4, R6: 204.188.45.128. Сеть между маршрутизаторами R1, R2, R3: 204.188.45.192. Компьютер PC1 имеет IP-адрес 204.188.45.1. Компьютер PC2 имеет IP-адрес 204.188.45.65. Компьютер PC3 имеет IP-адрес 204.188.45.129. Компьютер BOSS имеет IP-адрес 204.188.45.196.

Обозначения в задании: K1 – PC1, K2 – BOSS, K3 – PC3.

Вариант 8. Файл со схемой сети: lab2_var8.jfst. Сеть между узлами R1, R2, R3: 192.115.120.0. Сеть между узлами R4 и R7: 192.115.108.0. Сеть между узлами R6 и R7: 192.115.96.0. Компьютер PC1 имеет IP-адрес 192.115.128.1. Компьютер PC2 имеет IP-адрес 192.115.112.4. Компьютер PC3 имеет IP-адрес 192.115.88.2. Длина маски подсети должна быть минимально возможной.

Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.

Вариант 9. Файл со схемой сети: lab2_var9.jfst. Сеть между узлами R1, R2, R3: 10.0.120.0. Сеть между узлами R3 и R4: 192.168.0.0. Сеть между узлами R4 и R5: 192.168.1.0. Компьютер PC1 имеет IP-адрес 10.0.0.3. Компьютер PC2 имеет IP-адрес 10.0.0.10. Компьютер PC3 имеет IP-адрес 10.0.0.18.

Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.

Вариант 10. Файл со схемой сети: lab2_var10.jfst. Сеть между узлами R3 и R4: 192.168.0.0. Сеть между узлами R4 и R5: 192.168.1.0. Компьютер PC1 имеет IP-адрес 10.0.0.5. Компьютер PC2 имеет IP-адрес 10.0.0.130. Компьютер PC3 имеет IP-адрес 10.0.0.194.

Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.

Вариант 11. Файл со схемой сети: lab2_var11.jfst. Все маршрутизаторы и компьютеры имеют адреса из диапазона 192.168.0.1 – 192.168.0.254.
Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.
Вариант 12. Файл со схемой сети: lab2_var12.jfst. Все маршрутизаторы и компьютеры имеют адреса из диапазона 200.0.1.1 – 200.0.254.254.
Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.
Вариант 13. Файл со схемой сети: lab2_var13.jfst. Все маршрутизаторы и компьютеры имеют адреса из диапазона 172.1.1.1 – 172.254.254.254.
Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.
Вариант 14. Файл со схемой сети: lab2_var14.jfst.Все маршрутизаторы и компьютеры имеют адреса из диапазона 172.0.10.1 – 172.0.88.254.
Обозначения в задании: K1 – PC1, K2 – R5, K3 – PC3.

2.3.3. Пример выполнения лабораторной работы

Пусть дана конфигурация сети, показанная на рисунке 2.2. Файл со схемой сети: lab2_sample.jfst. Компьютер PC1 имеет IP адрес 172.168.0.2. Компьютер PC2 имеет IP адрес 10.0.0.2), Сеть между маршрутизаторами R1 и R: 172.168.100.0. Сеть между маршрутизаторами R2 и R: 192.168.0.0.

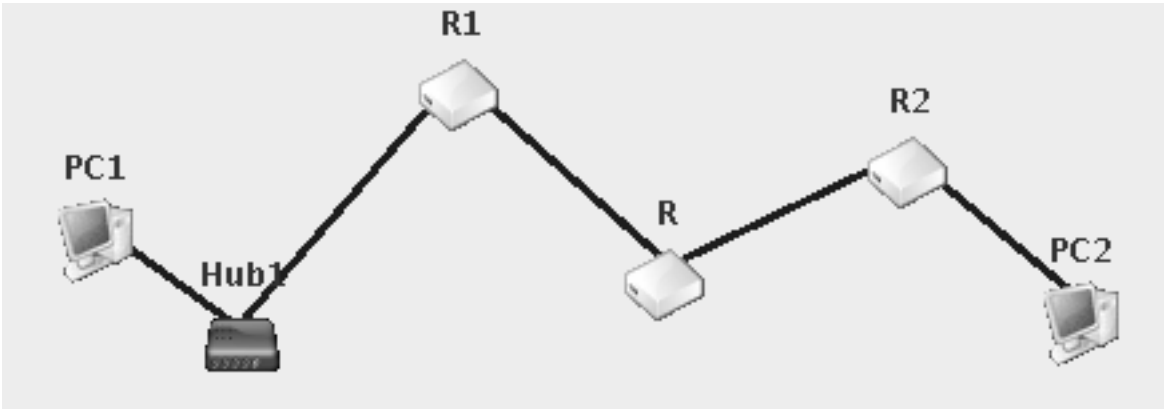


Рис. 2.2 Схема сети

Задание:

1. Задать маски подсети и шлюзы по-умолчанию для PC1 и PC2, а также IP-адреса из заданного диапазона вместе с масками и шлюзами для R1, R и R2 так, чтобы обеспечить корректную доставку пакетов от PC1 к PC2.
2. Настроить таблицу маршрутизации на R так, чтобы обеспечить корректную доставку пакетов от PC2 к PC1
3. Выполнить эхо-запрос с PC1 на PC2. Посмотреть вывод программы.

Порядок выполнения работы будет следующим:

1. Установим для всех узлов IP адреса, маски подсетей и шлюзы по умолчанию:
   PC1: IP адрес: 172.168.0.2, маска подсети: 255.255.255.0, шлюз по умолчанию: 172.168.0.1
   PC2: IP адрес: 10.0.0.2, маска подсети: 255.0.0.0, шлюз по умолчанию: 10.0.0.1
   R1: Интерфейс eth0: IP адрес: 172.168.0.1, маска подсети: 255.255.255.0, шлюз по умолчанию: 172.168.100.1
   R1: Интерфейс eth1: IP адрес: 172.168.100.2, маска подсети: 255.255.255.0, шлюз по умолчанию: 172.168.100.1
   R: Интерфейс eth0: IP адрес: 172.168.100.1, маска подсети: 255.255.255.0, шлюз по умолчанию: 192.168.0.1
   R: Интерфейс eth1: IP адрес: 192.168.0.2, маска подсети: 255.255.255.0, шлюз по умолчанию: 192.168.0.1
   R2: Интерфейс eth0: IP адрес: 192.168.0.1 маска подсети: 255.255.255.0, шлюз по умолчанию: 192.168.0.2
   R2: Интерфейс eth1: IP адрес: 10.0.0.1, маска подсети: 255.0.0.0, шлюз по умолчанию: 192.168.0.2
   Такая конфигурация обеспечит доставку пакетов от PC1 к PC2, но не обратно. Это объясняется тем, что при попытке отправки пакета с PC2 на PC1, пакет сначала попадет на R2 (т.к. он является шлюзом для PC2), а оттуда на R. Но с маршрутизатора R он снова будет отправлен на R2, т.к. для R шлюзом является R2. Таким образом, возникнет "петля". Устранить её можно с помощью настройки таблицы маршрутизации на R.
2. Таблицу маршрутизации на R будет выглядеть следующим образом:

```
R # route print
Destination     Gateway         Genmask         Type     Iface
default         192.168.0.1     255.255.255.0     0      eth0
```

   Для того чтобы пакеты от PC2 прошли через R на R1, необходимо добавить маршрут для сети 172.168.0.0/255.255.255.

```
R # route add 172.168.0.0 eth0 255.255.255.0 172.168.100.2
Route added.
```

   Теперь любой пакет, попавший на R и имеющий в качестве подсети назначения 172.168.0.0/255.255.255.0 будет направлен на маршрутизатор R1. Чтобы проверить корректность добавления маршрута необходимо вывести таблицу маршрутизации:

```
R # route print
IP routing table:
```

```
Destination     Gateway         Genmask         Type     Iface
default         192.168.0.1     255.255.255.0     0      eth0
172.168.0.0     172.168.100.2   255.255.255.0     0      eth0
```

С PC1 посылаем эхо-запрос PC2.

```
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 видно из вывода программы, на эхо-запрос пришел ответ, следовательно таблица маршрутизации настроена верно. Аналогичным образом можно настроить таблицы на остальных устройствах.

2.3.4. Контрольные вопросы

1. Что такое маршрутизация?
2. Для чего предназначен маршрутизатор?
3. Перечислите типы маршрутизации.
4. Что такое таблицы маршрутизации и для чего они нужны?
5. Какие типы записей могут быть в таблице маршрутизации?
6. Объясните механизм статической маршрутизации.

3. СЕТЕВОЙ УРОВЕНЬ: ICMP

Протокол передачи команд и сообщений об ошибках ICMP (Internet Control Message Protocol) предназначен для диагностики сетевых соединений и призван сообщать о сбоях и ошибках. Протокол ICMP активно используется протоколом TCP для контроля процесса передачи данных.

Протокол ICMP сообщает об ошибках в IP-дейтаграммах, но не дает информации об ошибках в самих ICMP-сообщениях. ICMP использует IP, а IP-протокол должен использовать ICMP. В случае IP-фрагментации сообщение об ошибке будет выдано только один раз на дейтаграмму, даже если ошибки были в нескольких фрагментах.

Обобщая изложенное, можно сделать вывод, что протокол ICMP осуществляет:

- передачу ECHO-запроса и ECHO-отклика;
- контроль времени жизни дейтаграмм в системе;
- реализацию переадресации пакета;
- выдачу сообщения о недостижимости адресата или о некорректности параметров;
- формирование и пересылку временных меток;
- выдачу запросов и откликов для адресных масок и другой информации.

ICMP-Сообщения никогда не выдаются:

- в ответ на сообщение ICMP об ошибке;
- при мультикастинговой или широковещательной адресации;
- для фрагмента дейтаграммы (кроме первого);
- для дейтаграмм, чей адрес отправителя является нулевым, широковещательным или мультикастинговым.

Эти правила призваны блокировать потоки дейтаграмм, создающих излишнюю загрузку сетевого трафика служебной информацией.

3.1. Типы сообщений ICMP

Список всевозможных типов сообщений протокола ICMP достаточно обширен; полный список этих типов приведен в [3]. Представленные там данные соответствуют действующей версии протокола IPv4; в разрабатываемой версии IPv6 (называемой еще IPng – от IP Next Generation) этот набор будет расширен.

Все ICMP-сообщения делятся на два типа:

- сообщения об ошибках;
- информационные.

Некоторые из типов сообщений ICMP будут рассмотрены более подробно.

3.1.1. Сообщение "ICMP_ECHO"

Одним из средств определения достижимости удаленного компьютера в сети является входящая в Windows NT утилита **ping**. В основе ее работы лежат ICMP-сообщения с кодами 8 (ECHO-запрос) и 0 (ECHO-отклик). Стандарт стека протоколов TCP/IP требует, чтобы любая его реализация полностью поддерживала протокол ICMP, – это означает, что любой компьютер, на котором установлен протокол IP, обязан корректно обрабатывать ICMP-запросы. Вид пакета для ICMP_ECHO представлен на рис. 3.1.

Поля *Идентификатор* и *Номер по порядку* служат для того, чтобы отправитель мог связать в пары запросы и отклики. Поле

Рис. 3.1

Тип определяет, является ли этот пакет запросом (8) или откликом (0). Поле *Контрольная сумма* представляет собой 16-разрядное дополнение по модулю 1 контрольной суммы всего ICMP-сообщения, начиная с поля *Тип*. Поле *Данные* служит для записи информации, возвращаемой отправителю. При выполнении процедуры PING ECHO-запрос с временной меткой в поле посылается адресату. Если адресат активен, он принимает IP-пакет, меняет адрес отправителя и адрес получателя местами и посылает его обратно отправителю. ЭВМ-Отправитель, восприняв этот отклик, может сравнить временную метку, записанную им в пакет, с текущим показанием внутренних часов и определить время распространения пакета – RTT (Round Trip Time). Размер поля *Данные* не регламентирован и определяется предельным размером IP-пакета.

Время распространения ICMP-запроса в общем случае не равно времени распространения отклика. Это связано не только с возможными изменениями в канале, но и с тем, что маршруты их движения могут быть различными.

3.1.2. Сообщение "Адресат недостижим"

Когда маршрутизатор не может доставить дейтаграмму по месту назначения, он посылает отправителю сообщение "Адресат недостижим" (destination unreachable). Формат такого сообщения показан на рис. 3.2.

Поле *Код* содержит целое число, соответствующее коду ошибки. Полный перечень кодов таких сообщений можно узнать из RFC792. Поле MTU характеризует максимальную длину пакетов на очередном шаге пересылки.

Рис. 3.2

Так как в сообщении содержится IP-заголовок и первые 64 бита дейтаграммы, которая вызвала возникновение ошибочной ситуации, легко понять, какой адрес оказался недостижим. Этот тип сообщения ICMP посылается и в случае, когда дейтаграмма имеет флаг DF = 1 ("не фрагментировать"), а фрагментация необходима. В поле *Код* в этом случае будет записано число 4.

Если буфер приема сообщений переполнен, ЭВМ посылает сообщение, содержащее в поле *Тип* значение 4 для каждого из не записанных в буфер сообщений.

3.1.3. Сообщение "Запрос временной метки"

В процессе трассировки маршрутов возникает проблема синхронизации работы часов в различных ЭВМ. Это требуется при синхронных измерениях, например при замере времени передачи пакета по сети. Для запроса временной метки другого узла сети используется сообщение "Запрос временной метки", которое вызывает отклик с форматом, представленным на рис. 3.3.

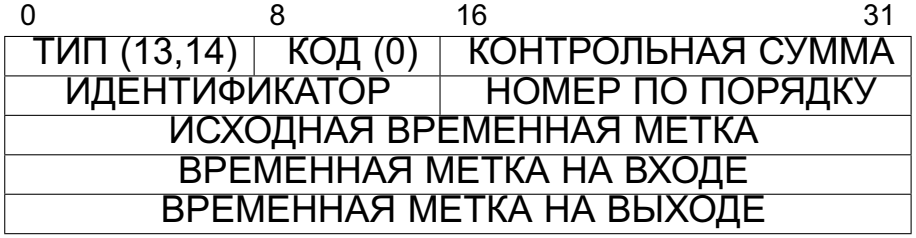


Рис. 3.3

Поле *Тип = 13* указывает на то, что это – запрос, а *Тип = 14* – отклик. Поля *Идентификатор* и *Номер по порядку* используются отправителем для связи запроса и отклика. Поле *Исходная временная метка* заполняется отправителем непосредственно перед отправкой пакета. Поле *Временная метка на входе* заполняется маршрутизатором при получении этого пакета, а поле *Временная метка на выходе* – непосредственно перед его отправкой.

Именно указанный формат используется в утилитах `ping` и `tracert`. Данные утилиты позволяют не только диагностировать, но и оптимизировать маршруты.

3.1.4. Сообщение "Запрос маски подсети"

Для получения маски подсети ЭВМ может послать сообщение "Запрос маски подсети" маршрутизатору и получить отклик, содержащий данную маску. ЭВМ может это сделать непосредственно, если ей известен адрес маршрутизатора, либо прибегнув к

широковещательному запросу. Формат пакета ICMP для запроса маски подсети представлен на рис. 3.4.

Рис. 3.4

Поле *Тип* здесь специфицирует модификацию сообщения: *Тип = 17* – это запрос, а *Тип = 18* – отклик. Поля *Идентификатор* и *Номер по порядку*, как обычно, обеспечивают взаимную привязку запроса и отклика, а поле *Адресная маска* содержит 32-разрядную маску подсети.

3.2. Определение маршрута IP-пакета при помощи ICMP

В Internet используется много различных типов пакетов, но один из основных – пакет IP (стандарт описан в RFC791): именно он вкладывается в кадр Ethernet и именно в него вкладываются пакеты UDP, TCP. Протокол IP предлагает ненадежную транспортную среду – ненадежную в том смысле, что гарантия благополучной доставки IP-дейтаграммы отсутствует.

Алгоритм доставки в рамках данного протокола предельно прост: при ошибке дейтаграмма уничтожается, а отправителю посылается соответствующее ICMP-сообщение или не посылается ничего. Обеспечение же надежности возлагается на более высокий уровень (UDP или TCP).

Маршрутизация IP определяет характер перемещения дейтаграмм через объединенные сети. В начале путешествия весь маршрут неизвестен. Вместо этого, на каждом переходе через маршрутизатор, вычисляется следующий пункт назначения сопоставлением адреса пункта назначения, содержащегося в дейтаграмме, с записью данных в маршрутной таблице текущего узла. Участие каждого узла в процессе маршрутизации дейтаграмм состоит из продвижения пакетов, базирующегося только на внутренней информации вне зависимости от того, насколько успешным будет процесс и достигнет или нет пакет конечного пункта назначения. Другими словами, IP не обеспечивает отправку сообщений о неисправностях в узел-источник, если возникли аномалии маршрутизации. Выполнение этой задачи предоставлено другому протоколу Internet, а именно протоколу управляющих сообщений (Internet Control Message Protocol – ICMP).

3.2.1. Поиск маршрута IP-пакета

Для определения маршрута прохождения пакета от источника к месту назначения существует несколько методик. Далее рассматривается одна из них. Для определения маршрута будет использоваться поле *TTL* в заголовке IP-пакета, и сообщение ICMP об истечении времени жизни этого пакета. Что же произойдет, когда, пройдя через очередной маршрутизатор, время жизни пакета TTL уменьшится на единицу и станет равным нулю? Пакет с *TTL = 0* не может быть передан далее по сети и подлежит уничтожению. Однако, уничтожая пакет, маршрутизатор должен сообщить отправителю посредством ICMP-протокола о недостижимости узла.

С учетом изложенного техника определения маршрута предельно проста: нужно посылать адресату IP-пакеты с временем *TTL = 1, 2, 3, . . .* и анализировать приходящие сообщения об ошибках до тех пор, пока не будет достигнут узел назначения.

Таким образом, записывая на каждом шаге адрес маршрутизатора, вернувшего сообщение ICMP об ошибке либо об истечении времени жизни пакета, можно определить маршрут пакета IP.

3.3. ICMP API в ОС Windows NT 4.0

Для использования протокола ICMP в программах необходима кропотливая работа по правильной инициализации полей заголовков пакетов, контролю за фрагментацией и сборкой, что требует значительных усилий со стороны программиста. Однако для более простого использования ICMP, когда для решения задачи не требуется иметь доступ к полям заголовка IP-пакета, в Windows NT 4.0 можно использовать специализированный API, помещенный в библиотеку icmp.dll, который позволяет работать с потоком ICMP-пакетов как с обычным файловым потоком.

Для определения маршрута IP-пакета по алгоритму, описанному ранее, потребуется несколько функций этого API. Их имена и назначение перечислены ниже:

- **IcmpCreateFile** открывает новый поток ICMP-сообщений, возвращает в качестве идентификатора хэндл нового потока для ICMP-сообщений. В дальнейшем для управления этим потоком нужно пользоваться данным идентификатором.
- **IcmpCloseHandle** закрывает открытый с помощью **IcmpCreateFile** ICMP-поток.
- **IcmpSendEcho** отправляет ICMP-пакет и получает(если нужно) ответное сообщение.

Наиболее интересной здесь является функция **IcmpSendEcho**, которая собственно и занимается посылкой и приемом сообщений. Поэтому следует рассмотреть ее прототип подробнее:

```
DWORD WINAPI IcmpSendEcho (HANDLE IcmpHandle,
    IPAddr DestinationAddress,
    LPVOID RequestData,
    WORD RequestSize,
    PIP_OPTION_INFORMATION RequestOptions,
    LPVOID ReplyBuffer,
    DWORD ReplySize,
    DWORD Timeout);
```

Первый параметр *IcmpHandle* – хэндл потока для ICMP-сообщений, полученный при помощи функции **IcmpCreateFile**. *DestinationAddress* – адрес узла назначения, которому направляются запросы. *RequestData*, *RequestSize* и *ReplyBuffer*, *ReplySize* – указатели на данные и размер этих данных для посылаемой и принимаемой информации соответственно. *RequestOptions* – указатель на структуру *IP_OPTION_INFORMATION*. Параметр *Timeout* – задает время ожидания ответа.

Структура *IP_OPTION_INFORMATION* описывается так:

```
typedef struct
{
 u_char Ttl;
 u_char Tos;
 u_char IPFlags;
 u_char OptSize;
 u_char FAR *Options;
} IP_OPTION_INFORMATION, *PIP_OPTION_INFORMATION;
```

Эта структура содержит дополнительные параметры посылаемых пакетов: время жизни, тип сервиса и флаги.

3.4. Практический пример

Для использования библиотеки ICMP API необходимо произвести ряд подготовительных действий. Сначала требуется загрузить DLL в адресное пространство текущего процесса при помощи функции **LoadLibrary**, а затем получить адреса всех необходимых функций при помощи **GetProcAddress**. Инициализацию ICMP API удобно оформить в виде следующей функции:

```
bool init_icmp_dll ()
```

```
{
 HINSTANCE icmp=LoadLibrary("icmp.dll");

 IcmpOpen=(HANDLE (WINAPI*) (VOID))
  GetProcAddress(icmp,"IcmpCreateFile");

 IcmpClose=(BOOL (WINAPI*)(HANDLE))
  GetProcAddress(icmp,"IcmpCloseHandle");

 IcmpSend=(DWORD (WINAPI*)
  (HANDLE,DWORD,LPVOID,WORD,PIPINFO,
   LPVOID,DWORD,DWORD))
  GetProcAddress(icmp,"IcmpSendEcho");

 if (IcmpOpen && IcmpClose && IcmpSend) return true;
 return  false;
}
```

Функция **init_icmp_dll** возвращает *true* в случае успешного подключения API и *false* – в противном случае.

Следующая функция реализует алгоритм трассировки пакета (т.е. механизм определения маршрута).

```
// he – указывает на заполненную структуру описания узла
void trace(hostent *he)
{
 IPINFO ipInf;        // информация о IP
 ICMPECHO icmpInf; // пакет ICMP
   // адрес узла назначения DWORD
 DWORD *addr=(DWORD*)(*he->h_addr_list);
 sockaddr_in dest; // полный адрес узла назначения
 in_addr ina;         // адрес ответившего узла
 char *name;          // указатель на имя узла
  // инициализация исходными значениями
 ina.S_un.S_addr=0;
 CopyMemory(&(dest.sin_addr),he->h_addr,he->h_length);
  // печать имени узла назначения
  printf ("\ndestination host %s",inet_ntoa(dest.sin_addr));
 HANDLE icmp=IcmpOpen();
 for( int  i=1;i<100;i++)
 {
   ipInf.Ttl=i ; //  установка текущего TTL
```

```
    ipInf .Tos =0; //  тип сервиса для ICMP
    //  отправление пакета и получение ответа на него
    int   f=IcmpSend(icmp,*addr,0,0,&ipInf,&icmpInf,
                          sizeof( struct  tagICMPECHO),10000);
   ina.S_un.S_addr=icmpInf.Source;
    //  если ответивший узел может вернуть свое имя, то
    //  выдача информации об узле маршрута в полном виде
   he=gethostbyaddr((char*)&ina,sizeof(ina),AF_INET);
    if (he)
   {
     name=he->h_name;
      printf ("\nReply from %-15s %s Time=%ldms TTL=%d",
                          inet_ntoa(ina ),
                          name,icmpInf.RTTime,icmpInf.ipInfo.Ttl);
      //  достигнут узел назначения, выход из цикла
      if (*addr==icmpInf.Source)
     {
        printf ("\nHost %s reached on %d hop(s).\n",he->h_name,i);
       break;
     }
   } else  //    печать адреса ответившего узла
   {
     printf ("\n * " );
   }
  }
}
```

3.5. Лабораторная работа 3

Разработать программу на основе ICMP API и найти маршрут до заданного преподавателем узла.

3.6. Контрольные вопросы

1. Для чего предназначен протокол ICMP?
2. Перечислите основные типы сообщений протокола ICMP и укажите их назначение.
3. В каких случаях сообщения ICMP не выдаются?
4. Опишите алгоритм нахождения маршрута пакета в IP-сетях на основе протокола ICMP.
5. Опишите принципы работы с различными API в ОС Windows NT.

6. Опишите состав и укажите назначение ICMP API в ОС Windows NT.

4. ТРАНСПОРТНЫЙ УРОВЕНЬ: СРАВНЕНИЕ ПРОТОКОЛОВ TCP И UDP

Поскольку на сетевом уровне не устанавливаются соединения, то нет никаких гарантий, что все пакеты будут доставлены к месту назначения в нужном порядке и без искажений информации, содержащейся внутри. Задачу обеспечения надежной информационной связи между двумя конечными узлами решает основной или транспортный уровень стека TCP/IP.

На этом уровне функционируют два протокола: протокол управления передачей (Transmission Control Protocol) и протокол дейтаграмм пользователя (User Datagram Protocol). Протокол TCP обеспечивает надежную передачу сообщений между удаленными прикладными процессами за счет образования логических соединений. Обмен данными возможен в дуплексном режиме.

TCP работает непосредственно над протоколом IP и использует для транспортировки своих блоков данных потенциально ненадежный протокол IP. Надежность передачи данных протоколом TCP достигается за счет того, что он основан на установлении логических соединений между взаимодействующими процессами. Поэтому ошибки протокола IP не будут влиять на правильное получение данных. Протокол IP используется протоколом TCP в качестве транспортного средства. Перед отправкой своих блоков данных протокол TCP помещает их в оболочку IP-пакета. При необходимости протокол IP осуществляет любую фрагментацию и сборку блоков данных TCP.

На рисунке 4.1 показано, как процессы, выполняющиеся на двух конечных узлах, устанавливают с помощью протокола TCP надежную связь через составную сеть, все узлы которой используют для передачи сообщений дейтаграммный протокол IP.

Протокол UDP позволяет передавать пакеты дейтаграммным способом, как и протокол уровня межсетевого взаимодействия IP. UDP выполняет только функции связующего звена между сетевым протоколом и пользовательскими процессами, а также многочисленными службами прикладного уровня.

4.1. Понятие портов

Протоколы TCP и UDP взаимодействуют через межуровневые интерфейсы с нижележащим протоколом IP и с вышележащими протоколами прикладного уровня или приложениями. Таким образом,

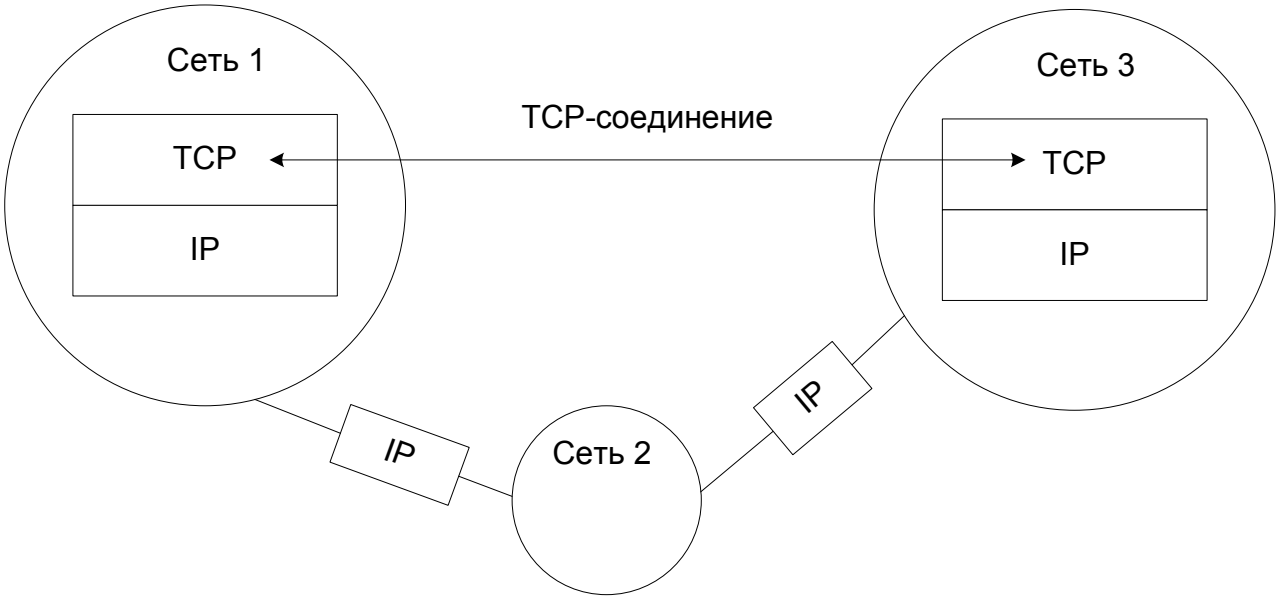


Рис. 4.1 логическое TCP–соединение

после того, как пакет средствами протокола IP доставлен в компьютер-получатель, данные необходимо направить конкретному процессу-получателю. На одном компьютере могут выполняться несколько процессов, более того, прикладной процесс может иметь несколько точек входа, выступающих в качестве адреса назначения для пакетов данных.

Пакеты, поступающие на транспортный уровень, организуются операционной системой в виде множества очередей к точкам входа различных прикладных процессов. В терминологии TCP/IP такие системные очереди называются *портами* [8]. Таким образом, адресом назначения, который используется протоколом TCP или UDP, является идентификатор (номер) порта прикладной службы. Номер порта в совокупности с номером сети и номером конечного узла однозначно определяет прикладной процесс в сети. Этот набор идентифицирующих параметров называется *сокет* (socket) [8]. Такое определение сокета можно применять не только в терминах TCP/IP, но и, к примеру, в терминах IPX.

Назначение номеров портов прикладным процессам осуществляется либо централизованно, если эти процессы представляют собой популярные общедоступные службы (FTP – 21, telnet – 23), либо локально для тех служб, которые еще не стали столь распространенными, чтобы закреплять за ними стандартные номера. Централизованное присвоение службам номеров портов выполняется организацией *Internet Assigned Numbers Authority (IANA)* [8]. Эти номера затем закрепляются и опубликовываются в стандартах Internet.

Протокол TCP (также как и UDP) ведет для каждого порта две очереди: очередь пакетов, поступающих в данный порт из сети, и очередь пакетов, отправляемых данным приложением через порт в сеть. Процедура обслуживания протоколом запросов, поступающих от нескольких различных прикладных служб называется *мультиплексированием* [8]. Обратная процедура распределения протоколом поступающих от сетевого уровня пакетов между набором высокоуровневых служб, идентифицированных номерами портов, называется *демультиплексированием* [8].

4.2. Протокол транспортного уровня TCP

Сегменты и потоки.

Единицей данных для протокола TCP является сегмент. Информация, поступающая по протоколу TCP в рамках соединения от протоколов более высокого уровня, рассматривается протоколом TCP как неструктурированный поток байтов.

Поступающие данные буферизуются. Для передачи на сетевой уровень из буфера вырезается некоторая часть данных, – это и есть *сегмент* [8]. Отличительной особенностью TCP является то, что он подтверждает получение не пакетов, а байтов потока.

Сегменты, посылаемые через соединение могут иметь разный размер. Оба участника соединения должны договариваться о максимальном размере посылаемого сегмента. Размер выбирается таким образом, чтобы при упаковке в IP-пакет сегмент помещался в него целиком без фрагментации, т.е. размер максимального сегмента не должен превосходить максимального размера поля данных IP-пакета.

Соединения.

Для организации надежной передачи данных предусматривается установление логического соединения между двумя прикладными процессами. Т.к. соединение устанавливается через ненадежную среду IP, то во избежание ошибочной инициализации соединений используется специальная многошаговая процедура подтверждения связи.

Соединение в протоколе TCP идентифицируется парой полных адресов обоих взаимодействующих процессов – *сокетов*. Каждый из взаимодействующих процессов может участвовать в нескольких соединениях.

Формально соединение – это набор параметров, характеризующий процедуру обмена данными между двумя процессами [8]. К таким параметрам относятся:

- Согласованные размеры сегментов.

- Объемы данных, которые разрешено передавать без подтверждения.
- Начальные и текущие номера передаваемых байтов.

В рамках соединения осуществляется обязательное подтверждение правильности приема для всех переданных сообщений и, при необходимости, выполняется повторная передача.

Установка связи по протоколу.

Этапы, из которых состоит процесс установки связи по протоколу таковы:

- Узел-отправитель запрашивает соединение, посылая сегмент с установленным флагом синхронизации (SYN).
- Узел-адресат подтверждает получение запроса, отправляя обратно сегмент с:
  - установленным флагом синхронизации;
  - порядковым номером начального байта сегмента, который он может послать;
  - подтверждением, включающим порядковый номер следующего сегмента, который он ожидает получить.
- Запрашивающий узел посылает обратно сегмент с подтверждением номера последовательности и номером своего подтверждения (ACK).

Этап соединения проиллюстрирован на рис. 4.2

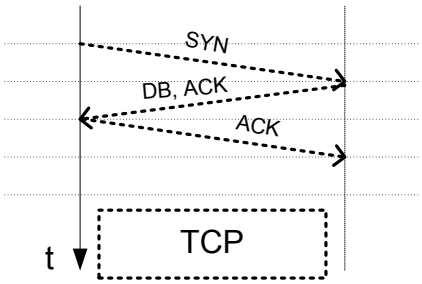


Рис. 4.2 этап соединения по протоколу TCP, SYN – пакет синхронизации, DB – блок данных, ACK – подтверждение

Из рисунка 4.2 видно, что соединение выполняется в три этапа, гарантирующих безошибочное установление связи.

Реализация скользящего окна в TCP.

Во время установленного соединения правильность передачи каждого сегмента должна подтверждаться квитанцией получателя. Квитирование – это один из традиционных методов обеспечения надежной связи. В протоколе TCP используется частный случай квитирования – алгоритм скользящего окна.

Идея состоит в том, что окно определено на множестве нумерованных байтов неструктурированного потока данных, поступающих с верхнего уровня и буферизуемых протоколом TCP. В качестве квитанции получатель сегмента отсылает ответное сообщение (сегмент), в которое помещается число на единицу большее, чем максимальный номер байта в полученном сегменте. Это число называют номером очереди [8].

Особенности реализации скользящего окна в протоколе TCP изображены на рис. 4.3. Если размер окна равен W, а последняя

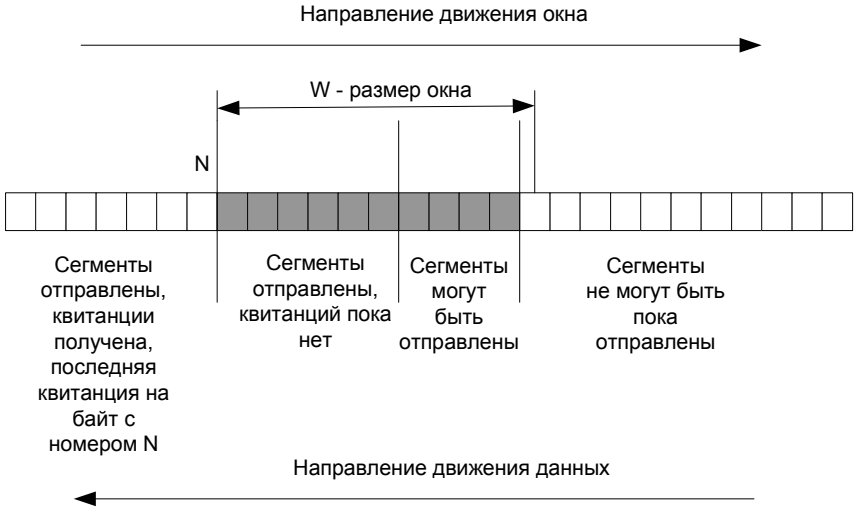


Рис. 4.3 скользящее окно в протоколе TCP

по времени квитанция содержала значение N, то отправитель может посылать новые сегменты до тех пор, пока в очередной сегмент не попадет байт с номером $N + W$. Этот сегмент выходит за рамки окна, и передача в таком случае должна быть приостановлена до прихода следующей квитанции.

Выбор тайм-аута очередной квитанции – важная задача, результат решения которой сильно влияет на производительность протокола TCP. Время не должно быть слишком коротким – для исключения повторных передач, снижающих полезную пропускную способность системы. И время не должно быть слишком большим во избежание длинных простоев, связанных с ожиданием несуществующей или потерявшейся квитанции.

Алгоритм определения тайм-аута. Для TCP он состоит в том, что при каждой передаче сегмента засекается время до прихода квитанции о его приеме (время оборота). Получаемые значения времен оборота усредняются весовыми коэффициентами, возрастающими от предыдущего значения к последующему, то есть усиливается влияние последних замеров. В качестве тайм-аута выбирается среднее время

оборота, умноженное на специальный коэффициент. На практике значение этого коэффициента должно превышать 2.

В сетях с большим разбросом времени оборота при выборе тайм-аута учитывается и дисперсия этой величины.

Поскольку каждый байт пронумерован, то каждый из них может быть опознан. Механизм опознавания является накопительным, поэтому опознавание номера X означает, что все байты с предыдущими номерами уже получены.

Этот механизм позволяет регистрировать появление дубликатов в условиях повторной передачи.

Нумерация байтов в пределах сегмента осуществляется так, чтобы первый байт данных сразу вслед за заголовком имел наименьший номер, а следующие за ним байты имели номера по возрастающей.

Размер окна, посылаемый получателем данных с каждым сегментом, определяет диапазон номеров очереди, которое отправитель окна (он же получатель данных) готов принять в настоящее время. Такой механизм связан с наличием в данный момент места в буфере данных.

Чем больше окно, тем большую порцию неподтвержденных данных можно послать в сеть. Но если пришло большее количество данных, чем может быть принято программой TCP, данные будут отброшены. Это приведет к излишним пересылкам информации и ненужному увеличению нагрузки на сеть и программу TCP. С другой стороны, указание малого размера окна может ограничить передачу данных скоростью, которая определяется временем путешествия по сети каждого сегмента в отдельности. Чтобы избежать применения малых окон, получателю данных предлагается откладывать изменение окна до тех пор, пока свободное место не составит 20-40 % от максимально возможного объема памяти для этого соединения. Отправитель также не должен спешить с посылкой данных, до того времени, пока окно не станет достаточно большим.

Разработчики протокола TCP предложили схему, согласно которой при установлении соединения заявляется большое окно, но впоследствии его размер существенно уменьшается.

Если сеть не справляется с нагрузкой, то возникают очереди в промежуточных узлах-маршрутизаторах и в конечных узлах-компьютерах.

При переполнении приемного буфера конечного узла "перегруженный" узел отправляет TCP-квитанцию, помещая в нее новый, уменьшенный размер окна. Если узел совсем отказывается от приема, то в квитанции устанавливается нулевой размер окна.

Однако, даже после этого приложение может послать сообщение на отказавшийся от приема порт, сопроводив его (сообщение) пометкой "срочно". В такой ситуации порт обязан принять сегмент, даже если для этого придется вытеснить их буфера уже находящиеся там данные.

После приема квитанции с нулевым размером окна протокол-отправитель время от времени делает контрольные попытки продолжить обмен данными. Если приемник уже готов принимать информацию, то в ответ он посылает квитанцию с указанием ненулевого размера окна.

Другим проявлением перегрузки сети является переполнение буферов в маршрутизаторах. В таких ситуациях они могут централизованно изменить размер окна, посылая управляющие сообщения некоторым конечным узлам, что позволяет им дифференцированно управлять интенсивностью потока данных в разных частях сети.

4.3. Протокол транспортного уровня UDP

Общее описание.

Протокол User Datagram Protocol (UDP) обеспечивает неориентированную на соединение службу доставки дейтаграмм по принципу "максимального усилия". Это означает, что получение всей дейтаграммы или правильной последовательности не гарантируется. Протокол UDP используется приложениями, не требующими подтверждения. Обычно такие приложения передают данные небольшого объема за один раз. К примеру, это: сервис имен Net-BIOS, сервис SNMP, сервис дейтаграмм NetBIOS.

Порты протокола UDP.

Для использования протокола UDP приложение должно знать IP-адрес и номер порта получателя. Порт действует как мультиплексная очередь сообщений, то есть он может получать несколько сообщение одновременно. Важно отметить, что порты UDP отличаются от портов TCP несмотря на использование одних и тех же значений номеров.

Описание работы UDP.

Порт UDP легче всего представить в виде очереди. В большинстве реализаций, когда прикладная программа "договаривается" с операционной системой об использовании данного порта, операционная система создает внутреннюю очередь, которая хранит приходящие сообщения. Часто приложение может указать или изменить размеры очереди. Когда узел получает дейтаграмму по UDP-протоколу, он проверяет, нет ли порта назначения с таким номером среди используемых портов. Если таких портов нет, UDP посылает ICMP-

сообщение об ошибке "порт недоступен" и уничтожает дейтаграмму. Если есть, UDP добавляет новую дейтаграмму в очередь порта, где прикладная программа может ее получить. Если очередь порта уже переполнена, то тогда UDP уничтожает новую дейтаграмму.

4.4. Сравнение производительности TCP и UDP

Последовательность действий (запросов и ответов) для протоколов TCP и UDP представлена на рис. 4.4. Как видно из рисунка, передача

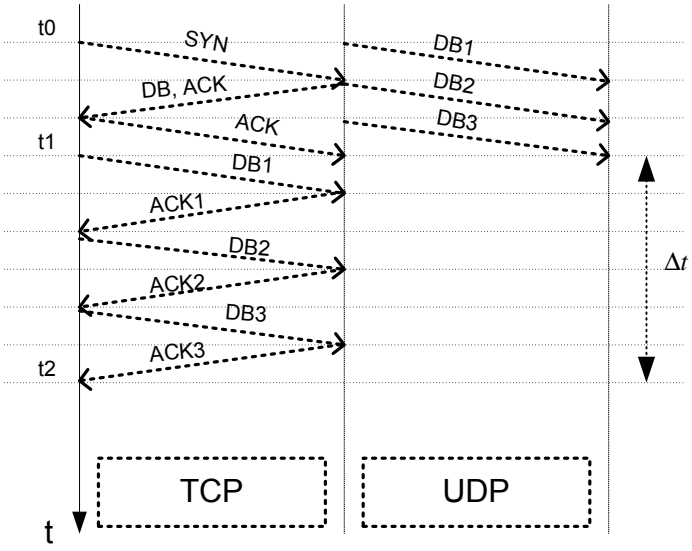


Рис. 4.4 последовательность запросов и ответов в TCP и UDP, SYN – пакет синхронизации, DB – блок данных, ACK – подтверждение

началась в момент времени t_0 по обоим протоколам. К моменту времени t_1 протокол UDP завершил передачу, в то время, как по протоколу TCP передача закончится только к моменту времени t_2. Легко получить разницу во времени $\Delta t = t_2 - t_1$. Таким образом, протокол UDP работает быстрее, чем TCP. Данные по протоколу UDP отсылаются получателю друг за другом и не требуется получение подтверждения об успешной доставке данных получателю. UDP не тратит время на установление связи в несколько этапов. Стоит отметить, что TCP путем механизма подтверждений гарантирует успешную доставку данных получателю, в то время, как данные посланные через UDP могут и не дойти до адресата.

4.5. Лабораторная работа №4

Цель: *провести анализ производительности протоколов TCP и UDP для заданной конфигурации сети, и на основании полученных результатов сделать заключение о том, какой протокол*

предпочтительнее использовать.

4.5.1. Порядок выполнения работы

1. В качестве схемы сети взять результат выполнения соответствующего варианта лабораторной работы №1. Установить коэффициенты прохождения пакетов согласно вашему варианту.
2. Протестировать отправку по UDP и по TCP 20 сообщений с K1 на K3.
3. Объяснить, анализируя вывод программы, какой протокол выгоднее использовать с точки зрения скорости доставки информации.
4. Протестировать отправку по UDP и по TCP 20 сообщений с K2 на K1.
5. Объяснить, анализируя вывод программы, какой протокол выгоднее использовать с точки зрения надежности доставки информации.
6. Подсчитать процент потерь пакетов. С учетом того, что должно теряться не более 7% пакетов. Объяснить, как привести сеть к требуемому лимиту по потерям.
7. Проанализировать время соединения, сделать вывод о том, какой протокол быстрее справился с поставленной задачей (необходимо учитывать требуемую надежность).
8. Определить состояние, при котором сеть начинает удовлетворять требованиям по потери пакетов. То есть подобрать такие значения коэффициентов пропускания, при которых будет теряться не более 7% пакетов. Разрешается использовать диапазон значений длины 10, то есть можно найти интервал значений коэффициентов пропускания длины 10, где на нижней границе сеть не удовлетворяет критерию потерь пакетов, а на верхней заданный критерий удовлетворяется.

Отчет должен содержать: анализ производительности протоколов TCP и UDP для заданной конфигурации сети при коэффициенте пропускания равном 100, расчет процента потерь пакетов и анализ производительности сети для обоих протоколов в условиях недоброкачественных линий передач для обоих протоколов, оценку удовлетворения сетью критерия по потере пакетов, анализ времени соединения. В отчете также необходимо привести вывод о том, какой протокол предпочтительнее использовать в данной конфигурации сети.

4.5.2. Варианты заданий

Вариант 1. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 1. Установите коэффициент прохождения пакетов между узлами Connector и Boss_R

в 82.
Обозначения в задании: K1 – Boss, K2 – Hacker, K3 – OFFICE2 pc1.
Вариант 2. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 2. Установите коэффициент прохождения пакетов между узлами H_OFFICE1 и OFF_R в 71.
Обозначения в задании: K1 – BIG BOSS, K2 – M_CH_S, K3 – OFFICE1 pc4.
Вариант 3. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 3. Установите коэффициент прохождения пакетов между узлами Connector и Hacker в 75.
Обозначения в задании: K1 – Boss, K2 – Hacker, K3 – OFFICE2 pc1.
Вариант 4. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 4. Установите коэффициент прохождения пакетов между узлами BOSS HUB и R2 в 85.
Обозначения в задании: K1 – BIG BOSS, K2 – M_CH_S, K3 – OFFICE1 pc4.
Вариант 5. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 5. Установите коэффициент прохождения пакетов между узлами HBosses и center в 80.
Обозначения в задании: K1 – MegaBoss, K2 – Manager2, K3 – FileServer.
Вариант 6. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 6. Установите коэффициент прохождения пакетов между узлами HManagers и center в 78.
Обозначения в задании: K1 – Manager3, K2 – PrintServer, K3 – MicroBoss.
Вариант 7. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 7. Установите коэффициент прохождения пакетов между узлами Hub2 и R1 в 85.
Обозначения в задании: K1 – Station1, K2 – Remote1, K3 – Station2.
Вариант 8. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 8. Установите коэффициент прохождения пакетов между узлами Hub2 и R1 в 55.
Обозначения в задании: K1 – Station1, K2 – Remote1, K3 – Station2.
Вариант 9. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 9. Установите

коэффициент прохождения пакетов между узлами Hub1 и R1 в 75.
Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.
Вариант 10. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 10. Установите коэффициент прохождения пакетов между узлами Hub1 и R1 в 65.
Обозначения в задании: K1 – PC1, K2 – PC2, K3 – PC3.
Вариант 11. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 11. Установите коэффициент прохождения пакетов между узлами R–C–M и HManagers в 75.
Обозначения в задании: K1 – Chief, K2 – Manager1, K3 – Service.
Вариант 12. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 12. Установите коэффициент прохождения пакетов между узлами R–M–S и HService в 80.
Обозначения в задании: K1 – Manager3, K2 – Service, K3 – Chief.
Вариант 13. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 13. Установите коэффициент прохождения пакетов между узлами H1 и R131 в 77. А также между узлами H2 и R120 в 90.
Обозначения в задании: K1 – Remote1, K2 – Remote2, K3 – Remote3.
Вариант 14. В качестве файла со схемой сети использовать результат выполнения лабораторной работы 1, вариант 14. Установите коэффициент прохождения пакетов между узлами H3 и Remote3 в 80. А также между узлами H2 и R120 в 85.
Обозначения в задании: K1 – Remote2, K2 – Remote3, K3 – Remote1.

4.5.3. Пример выполнения лабораторной работы

Файл со схемой сети: lab4_sample.jfst. Компьютер PC1 имеет IP-адрес 192.168.0.1. Компьютер PC2 имеет IP-адрес 192.168.0.2.
Задание:

1. Установить коэффициент пропускания всех линий равный 100.
2. Протестировать отправку по UDP и TCP 20 сообщений с PC2 на PC1.
3. Объяснить, анализируя вывод программы, какой протокол выгоднее использовать с точки зрения скорости доставки информации.
4. Установить коэффициент пропускания 65.
5. Протестировать отправку по UDP и TCP 20 сообщений с PC2 на PC1.

6. Объяснить, анализируя вывод программы, какой протокол выгоднее использовать с точки зрения надежности доставки информации.
7. Подсчитать процент потерь пакетов. С учетом того, что должно теряться не более 7% пакетов. Объяснить, как привести сеть к требуемой надежности.
8. Проанализировать время соединения, сделать вывод о том, какой протокол быстрее справился с поставленной задачей (необходимо учитывать требуемую надежность).
9. Определить состояние, при котором сеть начинает удовлетворять требованиям по потери пакетов. То есть подобрать такие значения коэффициентов пропускания, при которых будет теряться не более 7% пакетов. Разрешается использовать диапазон значений длины 10, то есть можно найти интервал значений коэффициентов пропускания длины 10, где на нижней границе сеть не удовлетворяет критерию потерь пакетов, а на верхней заданный критерий удовлетворяется.

Порядок выполнения работы будет следующим:

1. Убедимся, что коэффициент пропускания на всех линиях (в том числе между PC1 и PC2) равен 100.
2. Выберем PC1 и запустим на нем UDP-приложение (UPD-сервер), выбрав в качестве прослушиваемого порт 7.
   Программа выдаст следующее сообщение:
   ```
   pc1  Application is now listening on port 7.
   ```
3. Выберем PC2 и пошлем через UDP-приложение 20 сообщений со строкой "rsh"на PC1. Программа выдаст похожее на следующее сообщение (для первого из двадцати сообщений):
   ```
   pc2  Start sending echo message 'rsh' to 192.168.0.1:7
   pc2  Created UDP packet for 192.168.0.1:7.
   pc1  Created ARP Response packet to 192.168.0.2
   pc1  Sending packet from ProtocolStack (to 192.168.0.2).
   pc2  Sending packet from ProtocolStack (to 192.168.0.1).
   pc1  ProtocolStack received packet from local Interface.
   pc1  Confirmed Packet is for this Network Layer Device.
   pc1  UDP packet received from 192.168.0.2:3000 message:
        "rsh". UDP Port 7 has status "busy" from now.
   pc1  Application Recieving echo message 'rsh' from client.
   pc1  Application Sending echo message 'rsh' to client.
   pc1  Created UDP packet for 192.168.0.2:3000.
   pc1  Sending packet from ProtocolStack (to 192.168.0.2).
   pc2  ProtocolStack received packet from local Interface.
   ```

```
pc2  Confirmed Packet is for this Network Layer Device.
pc2  UDP packet received from 192.168.0.1:7 message: "rsh".
pc2  Recieving echo message 'rsh' from server.
pc1  Server closing connection. Now listening on 7.
pc1  Application is now listening on port 7.
```

4. Выберем меню статистики узла PC1 и проверим, сколько UDP дейтаграмм он получил и отправил. Будет выведен следующий результат: "Received UDP segments: 20", что означает, что получено 20 UDP дейтаграмм, и "Sent UDP segments: 20", что означает, что отправлено 20 UDP дейтаграмм. При заданных параметрах сети процент потерь равен 0%, что удовлетворяет требованиям.
5. Обнулим статистику узла PC1. Теперь установим коэффициент пропускания линии между двумя узлами в значение, равное 65 и снова пошлем с PC2 на PC1 20 UDP дейтаграмм.
6. Выберем меню статистики узла PC1 и проверим, сколько UDP дейтаграмм он получил и отправил. Будет, с большой вероятностью, выведен следующий результат: "Received UDP segments: 13", что означает, что получено 13 UDP дейтаграмм, и "Sent UDP segments: 13", что означает, что отправлено 13 UDP дейтаграмм.
7. Выберем меню статистики узла PC2 и проверим, сколько UDP дейтаграмм он получил и отправил за все время нашего опыта. Будет, с большой вероятностью, выведен следующий результат: "Received UDP segments: 26", что означает, что получено 26 UDP сегментов, и "Sent UDP segments: 40", что означает, что отправлено 40 UDP сегментов. При заданных параметрах сети процент потерь больше, чем 7%, что не удовлетворяет требованиям. Можно попробовать использовать протокол TCP.
8. Выберем PC1 и запустим на нем TCP-приложение (TCP-сервер), выбрав в качестве прослушиваемого порт 8.
   Программа выдаст следующее сообщение:
   ```
   pc1 Application is now listening on port 8.
   ```
9. Выберем PC2 и пошлем через TCP-приложение 20 сообщений со строкой "ppp"на PC1. Программа выдаст похожее на следующее сообщение (для первого из двадцати сообщений, дошедших до PC1):
   ```
   pc2  Connecting to host 192.168.0.1:8.
        Please wait...
   pc2  TCP SYN-packet for 192.168.0.1:8.
   pc1  ProtocolStack received packet from local Interface.
   pc1  Created ARP Response packet to 192.168.0.2
   ```

```
pc1  Sending packet from ProtocolStack (to 192.168.0.2).
pc2  ProtocolStack received packet from local Interface.
pc2  Sending packet from ProtocolStack (to 192.168.0.1).
pc1  ProtocolStack received packet from local Interface.
pc1  TCP SYN-packet received from 192.168.0.2:3000.
     TCP Port 8 has status "busy" from now.
pc1  Created TCP SYN-packet for 192.168.0.2:3000.
pc1  Sending packet from ProtocolStack (to 192.168.0.2).
pc2  TCP SYN-packet with ACK received from 192.168.0.1:8.
     TCP Port 3000 still has status "busy".
pc2  Created TCP acknowledgement packet for 192.168.0.1:8.
pc2  Sending packet from ProtocolStack (to 192.168.0.1).
pc1  TCP packet with establishing connection ACK received
     from 192.168.0.2:3000. Connection confirmed!
     New TCP connection established!
pc2  Start sending echo message 'ppp' to
     192.168.0.1:8
pc2  Created TCP data packet for 192.168.0.1:8.
pc2  Sending packet from ProtocolStack (to 192.168.0.1).
pc1  ProtocolStack received packet from local Interface.
pc1  Created TCP acknowledgement packet
     for 192.168.0.2:3000.
pc1  Sending packet from ProtocolStack
     (to 192.168.0.2).
pc2  ProtocolStack received packet from local Interface.
pc2  TCP packet with establishing connection ACK
     received from 192.168.0.1:8. Connection confirmed!
pc1  TCP packet with data received from 192.168.0.2:3000.
     Passing data to application program.
pc1  Recieving echo message 'ppp' from client.
pc1  Sending echo message 'ppp' to client.
```

10. Выберем меню статистики узла PC1 и проверим, сколько TCP сегментов он получил и отправил. Будет выведен следующий результат: "Received TCP segments: 45", "Sent TCP segments: 43", "Sent TCP ACKs: 23", что означает, что отправлено 23 подтверждения, также будет нулевая статистика по отосланным и принятым дубликатам. Выберем меню статистики узла PC2 и проверим, сколько TCP сегментов он получил и отправил. Будет выведен следующий результат: "Received TCP segments: 43", "Sent TCP segments: 45", "Sent TCP ACKs: 23", что означает, что

отправлено 23 подтверждения, также будет нулевая статистика по отосланным и принятым дубликатам. Из этого можно сделать вывод о том, что для хорошей линии передач излишне проводить загрузку канала подтверждениями о получении сегментов, которые занимают около 50% сегментов, задействованных в обмене информацией, однако, были доставлены все 20 сообщений, что удовлетворяет требованиям по процентам потерь.

11. Обнулим статистику узла PC1 и PC2. Теперь установим коэффициент пропускания 60 и снова пошлем с PC2 на PC1 20 TCP сегментов.
12. Выберем PC2 и пошлем через TCP-приложение 20 сообщений со строкой "ppp"на PC1.

    Программа, в нашем случае, выдаст следующее сообщение:

    ```
    pc2  Packet lost due to physical link problems!
    pc1  Server awaiting connection timeout!
         Now server is listening to port: 8.
    pc2  Connection timeout! Closing connection
         to host: 192.168.0.1:8.
    ```

    Это говорит о том, что на PC2 было закрыто подключение к PC1 и на PC1 было закрыто подключение к PC2, т.к. качество линий *в данном примере* не позволяет обмениваться информацией за установленные программой на соединение промежутки времени. При таких параметрах сеть не удовлетворяет требуемым условиям по потерям: не более 7%.

    Если проверить статистику PC2, то можно увидеть, что было отправлено 14 дубликатов, а получено 27 дубликатов. В то время, как было отправлено 10 сегментов, а получено 11.
13. Обнулим статистику узла PC1 и PC2. Теперь установим коэффициент пропускания 88 и снова пошлем с PC2 на PC1 5 TCP сегментов.
14. Выберем меню статистики узла PC2 и проверим, сколько TCP сегментов он получил и отправил за время нашего опыта. Будет, с большой вероятностью, выведен результат такой, что было отправлено 14 TCP сегментов, 8 дубликатов, 6 подтверждений, также было получено 10 сегментов.
15. В результате анализа полученных результатов можно сделать следующие выводы. В условиях качественного обеспечения передачи UDP протокол показал себя с хорошей стороны, так как все дейтаграммы дошли до адресатов. По времени было затрачено 32ms. Не тратилось время на установление соединения

и на подтверждения получения пакетов. При плохом качестве линий не все пакеты дошли до пунктов назначения. Оправданием использования UDP на плохих линиях может стать только то, что информация за время задержки или потери станет неактуальна, и ее можно не передавать (к примеру, видеоконференция через Интернет).

Результаты проведенной работы по протоколу TCP говорят о неэффективном использовании им (данным протоколом) качественных линий, так как дополнительное время тратится на подтверждение пакетов, а также на установление и разрыв связи. В условиях некачественной физической линии использование TCP явно предпочтительнее, так как "потерявшиеся" сегменты пересылаются и, в конечном счете, доходят до адресата. По времени передача по протоколу TCP заняла 344ms, что в 10.75 раза больше, чем время затраченное при передаче через UDP. Таким образом, применение протокола оправдано в случаях, требующих гарантированного получения адресатом всей посылаемой информации (к примеру, проверка электронной цифровой подписи).

Очевидно, что при использовании UDP сеть начинает удовлетворять семипроцентному критерию по потере пакетов при коэффициенте пропускания между узлами PC1 и PC2 не менее 93%. Если использовать TCP, то критерий по потере пакетов удовлетворяется при коэффициенте пропускания между узлами PC1 и PC2, принадлежащем интервалу от 60 до 65.

4.5.4. Контрольные вопросы

1. Какой из протоколов транспортного уровня обеспечивает надежную доставку данных? За счет какого механизма обеспечивается гарантия доставки?
2. Назовите ситуации, в которых применение протокола UDP является целесообразным.
3. Назовите ситуации, в которых применение протокола TCP является целесообразным.
4. Чем в TCP обеспечивается ускорение работы по передаче данных?
5. Сколькими пакетами обмениваются во время UDP-соединения клиент и сервер?
6. Сколькими пакетами обмениваются во время TCP-соединения клиент и сервер? Какие существуют пакеты TCP?

5. УРОВЕНЬ ПРИЛОЖЕНИЙ: ПРОТОКОЛЫ TELNET И SNMP

5.1. Уровень приложений стека протоколов TCP/IP

Уровень приложений модели стека протоколов TCP/IP выполняет следующие функции.

- Обеспечивает управление диалогом между устройствами: фиксирует какая из сторон является активной в настоящий момент, предоставляет средства синхронизации и занимается отделением данных одного приложения от данных другого приложения.

- Имеет дело с формой представления передаваемой по сети информацией, не меняя при этом ее содержания. За счет этого информация, передаваемая уровнем приложений одной системы, всегда понятна уровню приложений другой системы. С помощью средств данного уровня протоколы уровня приложений могут преодолеть синтаксические различия в представлении данных.
- Может выполнять шифрование и дешифрование данных.

- Предоставляет набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам, таким как файлы, принтеры или гипертекстовые Web-страницы, а также организуют свою совместную работу, например, с помощью протокола электронной почты [8].

Обобщая можно сказать, что прикладной уровень стека протоколов TCP/IP объединяет все службы, предоставляемые системой пользовательским приложениям. Прикладной уровень реализуется программными системами, построенными в архитектуре клиент-сервер, базирующимися на протоколах нижних уровней. И в отличие от протоколов остальных трех уровней стека протоколов TCP/IP, протоколы прикладного уровня описывают работу конкретного приложения и не способны к передаче данных по сети. Этот уровень постоянно расширяется т.к. наравне со старым, прошедшими многолетнюю эксплуатацию сетевыми протоколами типа TELNET, FTP, TFTP, DNS, SNMP создаются сравнительно новые такие, например, как протокол передачи гипертекстовой информации HTTP. Уровень приложений модели стека TCP/IP соответствует совокупности трех уровней модели OSI: сеансового уровня, уровня представлений и прикладного уровня. В этой главе рассмотрены два прикладных протокола: SNMP – простой протокол управления сетью и TELNET – протокол для создания

незащищенного соединения с серверным программным обеспечением.

5.1.1. Протокол SNMP

С любой сети функционирует большое количество узлов, маршрутизаторов и имеется широкий набор программных средств. Сеть сохраняет работоспособность благодаря жесткой протокольной регламентации, требующей разработки средств контроля и управления. Функции диагностики сети возложены на ICMP, а функции управления на SNMP (Simple Network Management Protocol – RFC1157). Чаще всего управляющая прикладная программа воздействует на сеть по цепочке: SNMP, UDP, IP, физическая сеть. *Управление сетью* – это процесс управления отказами, контроля конфигураций, мониторинга производительности, обеспечения защиты и учета деятельности в сети передачи данных [7]. Наиболее важным объектом управления обычно является маршрутизатор. Каждому управляемому объекту присваивается уникальный идентификатор.

Протокол SNMP использует UDP в качестве транспортного протокола и предназначен для использования сетевыми управляющими станциями. Он позволяет управляющим станциям собирать информацию о положении в сети. Протокол определяет формат данных, а их обработка и интерпретация остаются на усмотрение управляющих станций или менеджера сети.

Приложения управления сетью называемые *менеджерами*, общаются с программным обеспечением сетевых устройств, называемым *агентами*. SNMP – это протокол типа "запрос-отклик", то есть на каждый запрос, поступивший от менеджера, агент должен передать отклик. Под *запросом* будем понимать передачу информации от менеджера к агенту с целью получения параметров объекта управления. Под *откликом* будем понимать ответ агента, на запрос менеджера, содержащий требуемые параметры. Обмен данными между менеджером и агентом дает возможность менеджеру собирать стандартный набор информации, который определен в базе данных информации для управления сетью – MIB. Порция информации, существующая в базе данных, называется *объектом*.

Алгоритмы управления в сети обычно описывают в нотации ASN.1 (Abstract Syntax Notation). Все объекты в сети разделены на 10 групп и описаны в MIB: система, интерфейсы, обмены, трансляция адресов, IP, ICMP, TCP, UDP, EGP, SNMP. В группу "система" входит название и версия оборудования, операционной системы, сетевого программного обеспечения и пр. В группу "интерфейсы" входит число поддерживаемых интерфейсов, тип интерфейса, работающего под управлением IP (Eth-

ernet, LAPB и т.д.), размер дейтаграмм, скорость обмена, адрес интерфейса. IP-группа включает время жизни дейтаграмм, информацию о фрагментации, маски подсетей и т.д. В TCP-группу входит алгоритм повторной пересылки, максимальное число повторных пересылок и пр. [информация с сайта http://book.itep.ru/4/44/snm_4413.htm]

Протокол SNMP имеет достаточно простую структуру и включает в себя следующие команды:

- *Get-request* используется менеджером для получения от агента значения какого-либо параметра по его имени;
- *GetNext-request* используется для извлечения значения следующего объекта (без указания его имени) при последовательном просмотре таблицы объектов;
- *Set* используется менеджером для изменения значения какого-либо объекта. С помощью команды Set происходит собственно управление устройством. Агент должен понимать смысл значения объекта, который используется для управления устройством, и на основании этих значений выполнять реальное управляющее воздействие – отключить порт, установить IP адрес и т.п. Команда Set пригодна также для установки условия, при выполнении которого агент SNMP должен послать менеджеру соответствующее сообщение. Может быть определена реакция на такие события, как инициализация агента, рестарт агента, обрыв связи, восстановление связи, неверная аутентификация, потеря ближайшего маршрутизатора и др. Если происходит любое из этих событий, то агент инициализирует прерывание (*trap*). Если запросом *Set* устанавливаются значения сразу нескольких объектов, то в случае ошибки все объекты останутся без изменений;
- *Get-response* обеспечивает передачу ответа на команды *Get-request*, *GetNext-request* или *Set* от агента SNMP менеджеру. *Get-response* возвращает значения запрошенных объектов, только в случае успешного выполнения команд *Get*, *GetNext* или *Set*;
- *Trap* используется агентом для сообщения менеджеру о возникновении особой ситуации [8].

Схема иллюстрирующая обмен данными между SNMP менеджером и SNMP агентом представлена на рисунке 5.1. Прямоугольниками с числами обозначены порты на которых менеджер и/или агент ожидает дейтаграммы пользователя. Обычно SNMP агент использует 161 порт для ожидания запросов get, get-next или set, а SNMP менеджер – 162 порт для ожидания прерываний (trap). *Объектом управления* является любое сетевое устройство поддерживающее протокол SNMP,

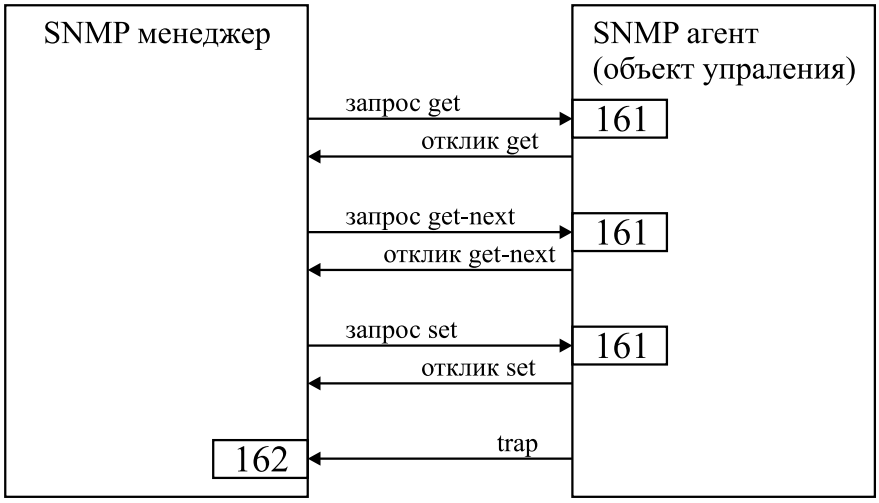


Рис. 5.1 Схема запросов/откликов SNMP.

на котором запущен SNMP агент.

В последнее время широкое распространение получила идеология распределенного протокольного интерфейса DPI (Distributed Protocol Interface). В этом случае для транспортировки SNMP-запросов используется не только UDP, но и TCP-протокол. Это дает возможность применять SNMP-протокол не только в локальных сетях. Форматы SNMP-DPI-запросов (версия 2.0) описаны в документе RFC1592. На рисунке 5.2 изображены форматы SNMP-DPI сообщений, вкладываемых в UDP-дейтаграммы для запросов *Get*, *GetNext* или *Set* На рисунке 5.2 прописными буквами латинского алфавита обозначены поля, которые содержат следующие данные:

A – длина сообщения, без первых двух байтов;

B – текущая версия протокола (для SNMPv2 это значение равно 2);

C – минимальная версия протокола совместимая используемой версией (для SNMPv2 это значение равно 2);

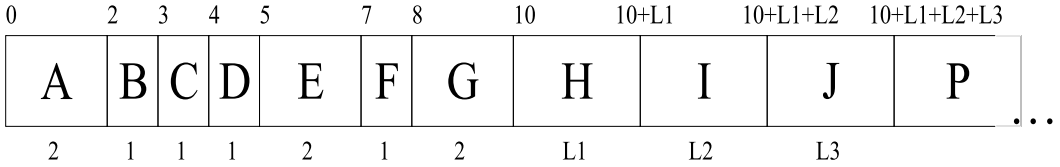
D – версия модификации основного протокола (в первой редакции протокола SNMPv2 это значение равно 0);

E – идентификатор сообщения, т.е. уникальное число, характеризующее отдельное сообщение посланный агенту и позволяющее связывать пары запрос-отклик (это необходимо при использовании UDP, т.к. возможна потеря пакетов);

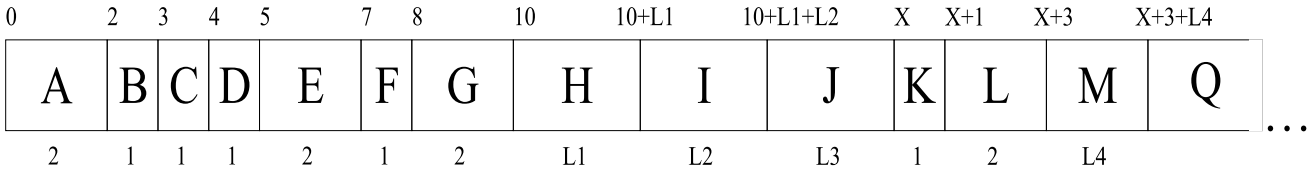
F – тип сообщения может принимать значения: SNMP_DPI_GET для get-запроса, SNMP_DPI_GETNEXT для getnext-запроса, SNMP_DPI_SET для set-запроса;

G – длина поля "имя группы доступа";

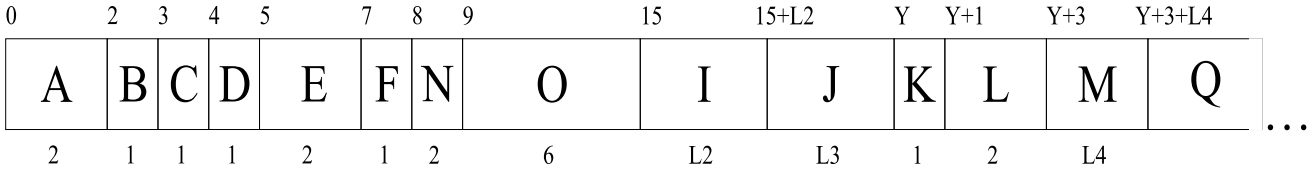
H – имя группы доступа – содержит последовательность символов, которая является пропуском при взаимодействии менеджера и объекта



(а) - формат пакета Get-Request и GetNext-Request



(б) - формат пакета Set-Request



(в) - формат пакета Get-Response

Рис. 5.2 Форматы сообщений протокола SNMP

управления (обычно это поле содержит 6-байтовую строку public);
I – идентификатор группы объекта управления – содержит последовательность чисел, определяющую путь по дереву MIB до объекта управления (последовательность должна заканчиваться значением *null*);
J – идентификатор объекта управления – лист MIB дерева, путь до которого определяет идентификатор группы объекта управления (идентификатор должен заканчиваться значением *null*);
K – тип SNMP переменной, например: SNMP_TYPE_Integer32, SNMP_TYPE_IpAddress, SNMP_TYPE_OCTET_STRING;
L – длина поля "значение SNMP переменной";
M – значение SNMP переменной соответствующего типа;
N – статус ошибки – определяет причину ошибки и может принимать следующие значения:

- noError(0) – нет ошибок;
- tooBig(1) – объект не может уложить отклик в одно сообщение;
- noSuchName(2) – в операции указана неизвестная переменная;
- badValue(3) – в команде set использована недопустимая величина или неправильный синтаксис;
- readOnly(4) – менеджер попытался изменить константу;

- genErr(5) – прочие ошибки;
- noAccess(6) – имя группы доступа содержащееся в сообщении и установленное на агенте не совпадают;
- wrongType(7) – использован неправильный тип SNMP переменной;
- wrongLength(8) – одно из полей длины не соответствует действительности;

O – индекс ошибки – порядковый номер SNMP переменной, которая привела к ошибке;
P – в случае необходимости возможна запись дополнительных SNMP переменных, т.е. пар: идентификатор группы(I), идентификатор объекта(J);
Q – в случае необходимости возможна запись дополнительных SNMP переменных со значениями, т.е. последовательностей состоящих из пяти полей: идентификатор группы(I), идентификатор объекта(J), тип SNMP переменной(K), длина поля "значение SNMP переменной"(L), значение SNMP переменной(M).
На рисунке числа и буквы над полями пакета обозначают смещение соответствующего поля от начала пакета в байтах. А числа и буквы под полями пакета обозначают длину соответствующего поля. Для краткости записи были введены следующие обозначения:
L1 – длина поля "имя группы доступа";
L2 – длина поля "идентификатор группы объекта управления";
L3 – длина поля "идентификатор объекта управления";
L4 – длина поля "значение SNMP переменной";
X = 10+L1+L2+L3;
Y = 15+L2+L3.
Видно, что сообщения SNMP имеют достаточно простую структуру. Это упрощает реализацию протокола, но ведет к тому, что имя группы доступа, предназначенное для ограничения доступа к SNMP агенту, ни как не шифруется и передается по сети в открытой форме.

5.1.2. Управляющая база данных MIB

Вся управляющая информация для контроля сетевых устройств (маршрутизаторы, коммутаторы и т.п.) концентрируется в базе данных MIB (Management Information Base, RFC1212, RFC1213). Каждая порция информации, существующая в базе данных, называется *объектом*. База данных информации для управления сетью содержит объекты, которые нужны менеджеру для управления сетью. MIB выглядит как дерево с раздельными пунктами данных в качестве выходов. Идентификатор объекта однозначно идентифицирует MIB-объект в дереве. Объект обозначается, как последовательность

чисел, разделенных точкой. Объекты организуются иерархически и их части могут принадлежать различным организациям. Верхний уровень идентификаторов MIB объектов установлен ISO/IEC. Объекты более низкого уровня выделяются специальными организациями. MIB дерево постоянно расширяется, как результат экспериментов частных разработок. Производители, например, могут определить свои личные ветви для включения образов своих продуктов. Такие деревья MIB не стандартизируются, а носят характер экспериментальных деревьев. Пример части такого дерева приведен на рисунке 5.3. Из этого рисунка

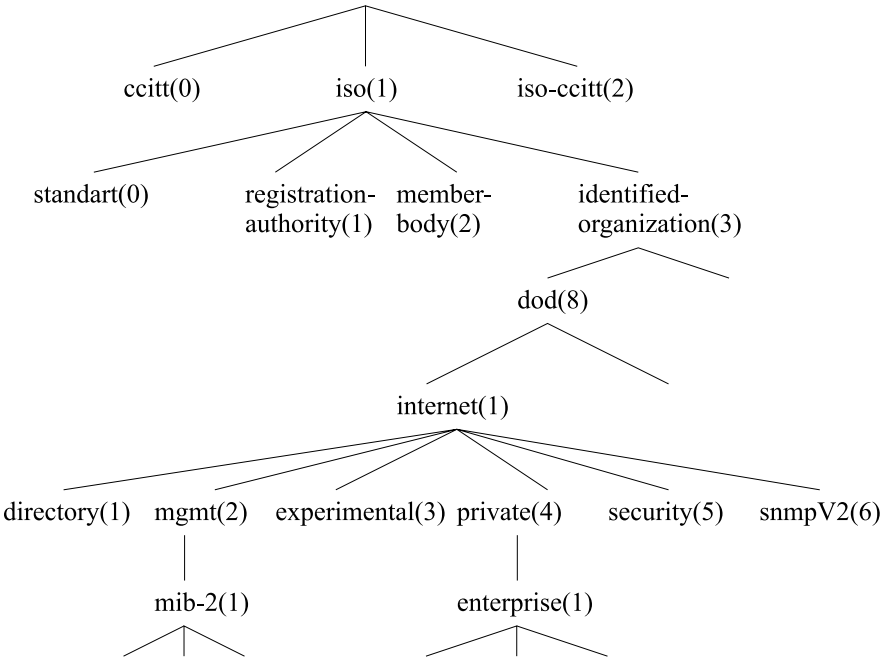


Рис. 5.3 Дерево MIB

видно, что например для узла snmpV2 идентификатор объекта будет: 1.3.8.1.6

5.1.3. Протокол TELNET

Для обеспечения удаленного доступа к сетевому устройству с помощью командного интерпретатора используется протокол TELNET (RFC854). Протокол TELNET – это сетевой протокол типа "клиент-сервер". TELNET обеспечивает незащищенное соединение, т.е. все данные передаются в открытой форме в том числе и пароли. TELNET использует TCP в качестве транспортного протокола. Общепринято, что TELNET-сервер ожидает соединения на 23 порту. TELNET позволяет пользователю установить TCP-соединение с сервером и затем передавать коды нажатия клавиш так, как если бы работа проводилась на консоли сервера. Для входа в командный режим обычно

нужна аутентификация – ввод имени пользователя и его пароля).
TELNET предлагает три услуги:

- определяет сетевой виртуальный терминал (NVT – network virtual terminal), который обеспечивает стандартный интерфейс доступа к удаленной системе;
- включает механизм, который позволяет клиенту и серверу согласовать опции обмена;
- обеспечивает соединение, при котором любая программа (например FTP) может выступать в качестве клиента.

Протокол TELNET позволяет обслуживающей машине рассматривать все удаленные терминалы как стандартные "сетевые виртуальные терминалы" строчного типа, работающие в кодах ASCII, а также обеспечивает возможность согласования более сложных функций (например, локальный или удаленный эхо-контроль, страничный режим, высота и ширина экрана и т. д.). На прикладном уровне над протоколом TELNET находится либо программа поддержки реального терминала, либо прикладной процесс в обслуживающей машине, к которому осуществляется доступ с терминала. Формат NTV достаточно прост. Для данных используются 7-битовые ASCII коды. А октеты из восьми бит зарезервированы для командных последовательностей [информация с сайта http://book.itep.ru/4/45/tlnt_453.htm].
В упрощенном варианте протокол TELNET работает следующим образом: Между клиентом и сервером устанавливается TCP соединение. Клиент посылает серверу символ перевода строки для того, чтобы сервер знал что это клиент хочет соединится по TELNET. В ответ сервер посылает приглашение ввода имени (например: login) и ждет ввода имени пользователя. После ввода сервер посылает приглашение ввода пароля (например: password) и ждет ввода пароля. Если введенные имя и пароль корректны то TELNET-сервер переходит в режим ввода. В этом режиме любой введенный текст пересылается удаленному сетевому устройству. Ввод может производиться посимвольно или построчно. При посимвольном режиме каждый введенный символ пересылается немедленно, при построчном режиме отклик на каждое нажатие клавиши производится локально, а пересылка выполняется лишь при нажатии клавиши <Enter>. В режиме ввода TELNET-сервер выдает какое-либо приглашение (например: telnet>), и ожидает ввода команд пользователя. При вводе команды quit сервер разрывает соединение.

5.2. Лабораторная работа 5

Цель: *на примере протоколов SNMP-DPI и TELNET ознакомиться с уровнем приложений стека протоколов TCP/IP.*

5.2.1. Порядок выполнения работы

1. На компьютере К1 запустить SNMP агента. Порт и имя группы доступа выбираются студентом;
2. С компьютера К2 отправить запрос(ы) *get*, и получить переменные П1, П2, П3. Сравнить полученные значения с реальными;
3. С компьютера К2 отправить запрос(ы) *getnext* для переменных П1, П2, П3. Объяснить полученные результаты;
4. На компьютере К1 с помощью диалога "Set TCP/IP Properties" изменить IP адрес, маску подсети и шлюз по умолчанию. С компьютера К2 с помощью запросов *set* вернуть К1 в исходное состояние. Проверить результаты посредством SNMP;
5. На компьютере К2 запустить TELNET сервер. Порт и пароль выбрать самостоятельно;
6. С компьютера К3 по протоколу TELNET подключиться к компьютеру К2. Удалить все значения из таблицы маршрутизации и ARP таблицы. Добавить в таблицу маршрутизации и ARP таблицу записи необходимые для корректной работы компьютера К2;
7. С помощью команды TELNET-сервера *snmp* запустить SNMP агента на К3. Проверить работоспособность snmp-сервера: с компьютера К2 попытаться получить значение SNMP переменной П2;

В отчет необходимо включить схему сети, все вводимые параметры (порт, имя группы доступа и др.), отправляемые запросы и получаемые ответы. Для протокола TELNET необходимо привести сообщения выводимые в TELNET-консоль.

5.2.2. Варианты заданий

Вариант 1. Файл со схемой сети lab1_var1.jfst.
Обозначения в задании: Компьютеры К1 – OFFICE2 pc1; К2 – Boss; К3 – Hacker. SNMP переменные П1 – Counter.InputIP; П2 – IP.AllInterfaces; П3 – IP.Address_Eth0.
Вариант 2. Файл со схемой сети lab1_var2.jfst.
Обозначения в задании: Компьютеры К1 – OFFICE1 pc4; К2 – BIG BOSS; К3 – M_CH_S. SNMP переменные П1 – Counter.OutputIP; П2 – IP.ARPTable; П3 – IP.SubnetMask_Eth0.
Вариант 3. Файл со схемой сети lab1_var3.jfst.
Обозначения в задании: Компьютеры К1 – OFFICE2 pc2; К2 – Hacker; К3 – Boss. SNMP переменные П1 – Counter.ARP; П2 – IP.DefaultGateway;

П3 – SNMP.CommunityName.

Вариант 4. Файл со схемой сети lab1_var4.jfst.

Обозначения в задании: Компьютеры К1 – BIG BOSS; К2 – OFFICE1 pc1; К3 – OFFICE1 pc3. SNMP переменные П1 – Counter.InputTCP; П2 – IP.Address_Eth0; П3 – SNMP.revision.

Вариант 5. Файл со схемой сети lab1_var5.jfst.

Обозначения в задании: Компьютеры К1 – FileServer; К2 – Manager1; К3 – MegaBoss. SNMP переменные П1 – Counter.OutputTCP; П2 – IP.SubnetMask_Eth0; П3 – IP.DefaultGateway.

Вариант 6. Файл со схемой сети lab1_var6.jfst.

Обозначения в задании: Компьютеры К1 – PrintServer; К2 – Manager3; К3 – MicroBoss. SNMP переменные П1 – Counter.ReceiveDuplicatedTCP; П2 – SNMP.CommunityName; П3 – IP.ARPTable.

Вариант 7. Файл со схемой сети lab1_var7.jfst.

Обозначения в задании: Компьютеры К1 – Station1; К2 – Station4; К3 – Remote1. SNMP переменные П1 – Counter.SendDuplicatedTCP; П2 – SNMP.revision; П3 – IP.AllInterfaces.

Вариант 8. Файл со схемой сети lab1_var8.jfst.

Обозначения в задании: Компьютеры К1 – Station3; К2 – Remote1; К3 – Station2. SNMP переменные П1 – Counter.SendAckTCP; П2 – Counter.InputIP; П3 – Device.MACaddress_Eth0.

Вариант 9. Файл со схемой сети lab1_var9.jfst.

Обозначения в задании: Компьютеры К1 – PC1; К2 – PC2; К3 – PC4. SNMP переменные П1 – Counter.InputUDP; П2 – Counter.OutputIP; П3 – Device.AvailableInterfaces.

Вариант 10. Файл со схемой сети lab1_var10.jfst.

Обозначения в задании: Компьютеры К1 – PC2; К2 – PC3; К3 – PC4. SNMP переменные П1 – Counter.OutputUDP; П2 – Counter.ARP; П3 – Device.AllInterfaces.

Вариант 11. Файл со схемой сети lab1_var11.jfst.

Обозначения в задании: Компьютеры К1 – Chief; К2 – Service; К3 – Manager1. SNMP переменные П1 – Device.AllInterfaces; П2 – Counter.InputTCP; П3 – Device.Hostname.

Вариант 12. Файл со схемой сети lab1_var12.jfst.

Обозначения в задании: Компьютеры К1 – Manager3; К2 – Service; К3 – Chief. SNMP переменные П1 – Device.AvailableInterfaces; П2 – Counter.OutputTCP; П3 – Counter.OutputUDP.

Вариант 13. Файл со схемой сети lab1_var13.jfst.

Обозначения в задании: Компьютеры К1 – Remote1; К2 – Remote2; К3 – Remote3. SNMP переменные П1 – Device.Hostname; П2 –

Counter.ReceiveDuplicatedTCP; П3 – Counter.InputUDP.
Вариант 14. Файл со схемой сети lab1_var14.jfst.
Обозначения в задании: Компьютеры К1 – Remote2; К2 – Remote3; К3 – Remote1. SNMP переменные П1 – Device.MACaddress_Eth0; П2 – Counter.SendDuplicatedTCP; П3 – Counter.SendAckTCP.

5.2.3. Пример выполнения лабораторной работы

Файл со схемой сети: javaNetSim/labs/lab1/lab1_sample.jfst.

Задание:

1. На компьютере PC1 запустить SNMP агента.
2. С компьютера PC2 отправить запрос(ы) get, и получить переменные *ip.address_eth0*, *device.hostname*.
3. На компьютере PC2 запустить TELNET-сервер.
4. С компьютера PC1 по протоколу TELNET подключиться к компьютеру PC2. Удалить все значения из кэша ARP. Добавить туда статическую запись для узла PC1.

Порядок выполнения работы будет следующим:

1. Запустим на PC1 SNMP агент с параметрами:
   - порт на котором SNMP агент будет ожидать пакеты: *161*;
   - имя группы доступа для SNMP агента: *defgroup*.
2. Выполним с PC2 запрос SNMP-агенту на PC1 со следующими параметрами:
   - IP адрес компьютера на котором установлен SNMP агент: *172.168.0.2*;
   - порт на котором SNMP агент ожидает пакеты: *161*;
   - SNMP запрос: *get*;
   - SNMP переменные: *ip.address_eth0;device.hostname*;
   - имя группы доступа: *defgroup*.

   Результаты запроса будут выведены в консоль:

   ```
   PC2  Received getResponse:
        'IP.Address_Eth0=172.168.0.2', 'Device.Hostname=PC1'
   ```
3. Запустим TELNET-сервер со следующими параметрами:
   - порт, на котором TELNET-сервер будет ожидать пакеты: *23*;
   - пароль для доступа к TELNET: *234*.
4. Запустим TELNET-клиент с параметрами:
   - IP адрес TELNET сервера: *10.0.0.2*;
   - порт, на котором TELNET-сервер ожидает пакеты: *23*.

   В ответ на приглашение к авторизации в системе необходимо ввести имя пользователя *root* и пароль *234*. После входа в систему будет

выведено приглашение командной строки:

```
pc1 #
```

Просмотрим записи в таблице маршрутизации:

```
pc1 # arp -a
Internet Address    Physical Address    Type
10.0.0.1            A2:2A:55:20:75:42   Dynamic
```

Как видно из вывода команды arp, в кэше находится лишь одна динамическая запись. Ее можно удалить следующим образом:

```
pc1 # arp -d 10.0.0.1
```

Для добавления статической записи в кэш ARP необходимо использовать ключ -s команды arp:

```
pc1 # arp -s 10.0.0.1 A2:2A:55:20:75:42
```

Таким образом была добавлена статическая запись для компьютера PC1. После завершения работы закрываем сеанс TELNET.

5.2.4. Контрольные вопросы

1. Для чего предназначен протокол SNMP?
2. Если на SNMP запрос пришел отклик с установленным флагом ошибки, то какие переменные будут содержаться в этом отклике? Если в set запросе часть переменных имеет корректные значения, а часть некорректные то какие переменные объекта управления изменятся?
3. Как обеспечивается защита в протоколе SNMP? Как вы думаете насколько безопасно применения протокола SNMP для управления реальной сетью? Что надо сделать для увеличения безопасности?
4. Для чего предназначен протокол TELNET?
5. Как работает протокол TELNET? Как обеспечивается безопасность при вводе пароля?

6. 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. В связи с этим имитатор javaNetSim удобен для выполнения лабораторных работ. Основные приемы работы с имитатором javaNetSim будут рассмотрены в данной главе.

Имитатор javaNetSim является объектно-ориентированным и написан на языке Java. Программы написанные на этом языке являются машинно-независимыми, т.е. имитатор javaNetSim будет работать на любом компьютере, для которого есть виртуальная Java машина. Хотя язык Java является интерпретируемым, это не оказывает существенного

влияния на быстродействие имитатора. Это объясняется тем, что имитатор разрабатывался для моделирования работы небольших сетей, обработка моделей которых не требует больших вычислительных ресурсов.

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 какого-либо 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). Рабочая область является частью основного окна программы, представленного на рисунке 6.1. Основное окно программы логически разделено на четыре части:

- рабочая область, обозначенная цифрой 1 – содержит сетевые устройства и линии связи между ними:
  - концентратор на пять сетевых интерфейсов – обозначен числом 13;

    - маршрутизатор соединяющий две подсети – обозначен числом 14;
    - компьютер (конечный узел сети) – обозначен числом 15;
    - Линия связи между двумя сетевыми устройствами – обозначен числом 16.
- область вывода результатов, обозначенная цифрой 2 – содержит две вкладки:
    - вкладка "консоль", обозначенная числом 11 – содержит журнал передачи пакетов по сети
    - вкладка "информация об устройствах", обозначенная числом 12 – для каждого интерфейса всех сетевых устройств содержит IP адрес, маску подсети и шлюз по умолчанию.
- главное меню, обозначенное цифрой 3 – содержит основные действия по управлению имитатором;
- линейка инструментов – содержит следующие кнопки:
    - кнопка "создать пустую конфигурацию" – обозначена цифрой 4;
    - кнопка "открыть существующую конфигурацию" – обозначена цифрой 5;
    - кнопка "сохранить текущую конфигурацию" – обозначена цифрой 6;
    - кнопка "создать компьютер" – обозначена цифрой 7;
    - кнопка "создать маршрутизатор" – обозначена цифрой 8;
    - кнопка "создать концентратор" – обозначена цифрой 9;
    - кнопка "создать соединение" – обозначена числом 10.

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.

6.1. 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.

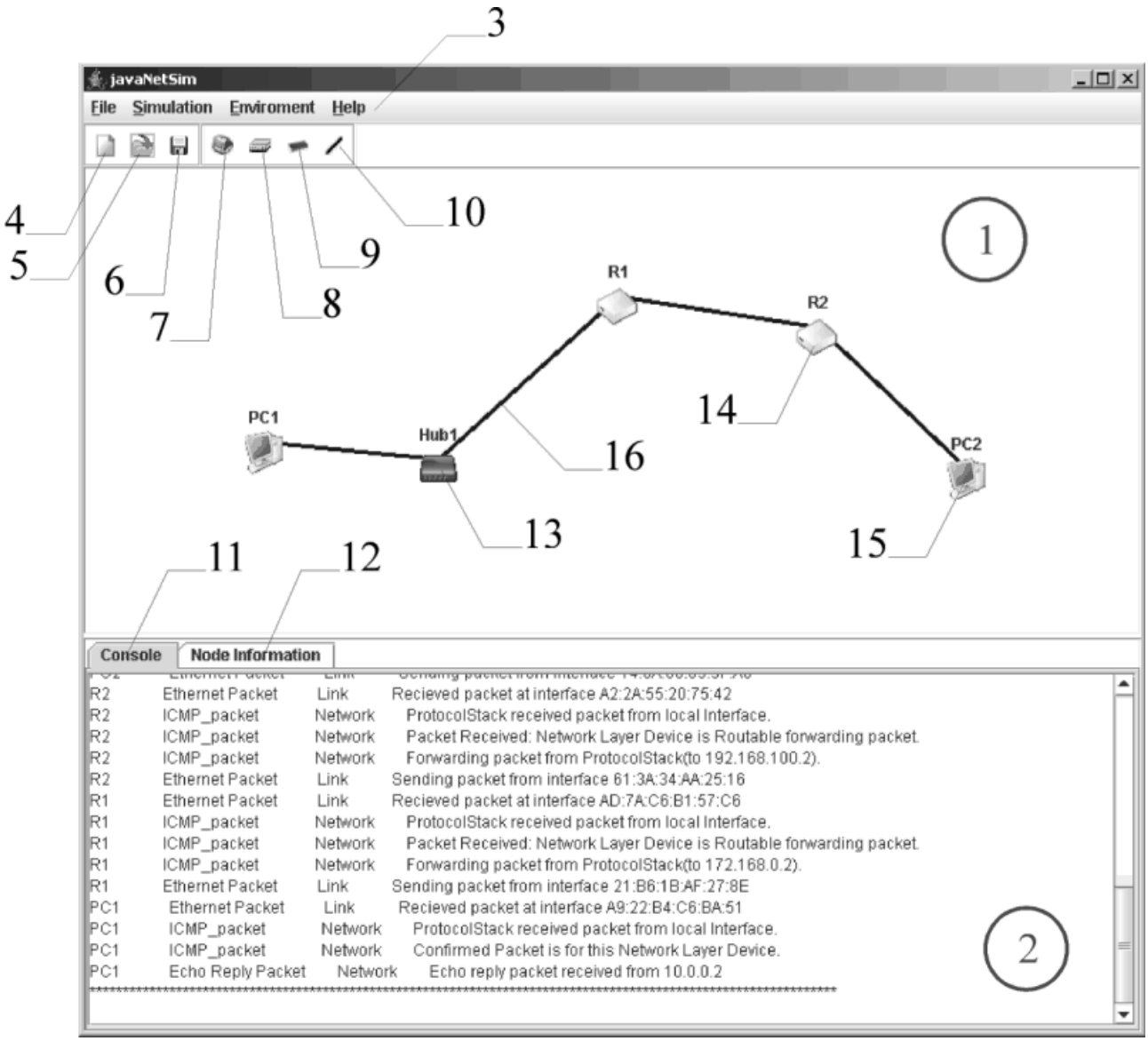


Рис. 6.1 Основное окно программы

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) позволяющий изменить свойства 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 (рисунок 6.2) 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.

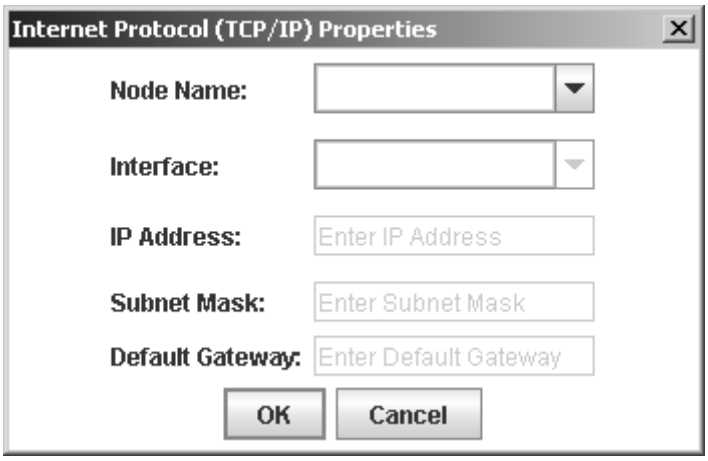


Рис. 6.2 Установка параметров TCP/IP

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) – позволяет задать 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.

6.2. Контекстное меню

Контекстное меню, вызываемое щелчком правой кнопкой мыши, отличается для устройств работающих на разных уровнях стека протоколов TCP/IP. На рисунке 6.3 изображены контекстные меню соответственно для устройств (а) – физического (концентратор), (б) – сетевого (маршрутизатор) и (в) – прикладного (компьютер) уровней.

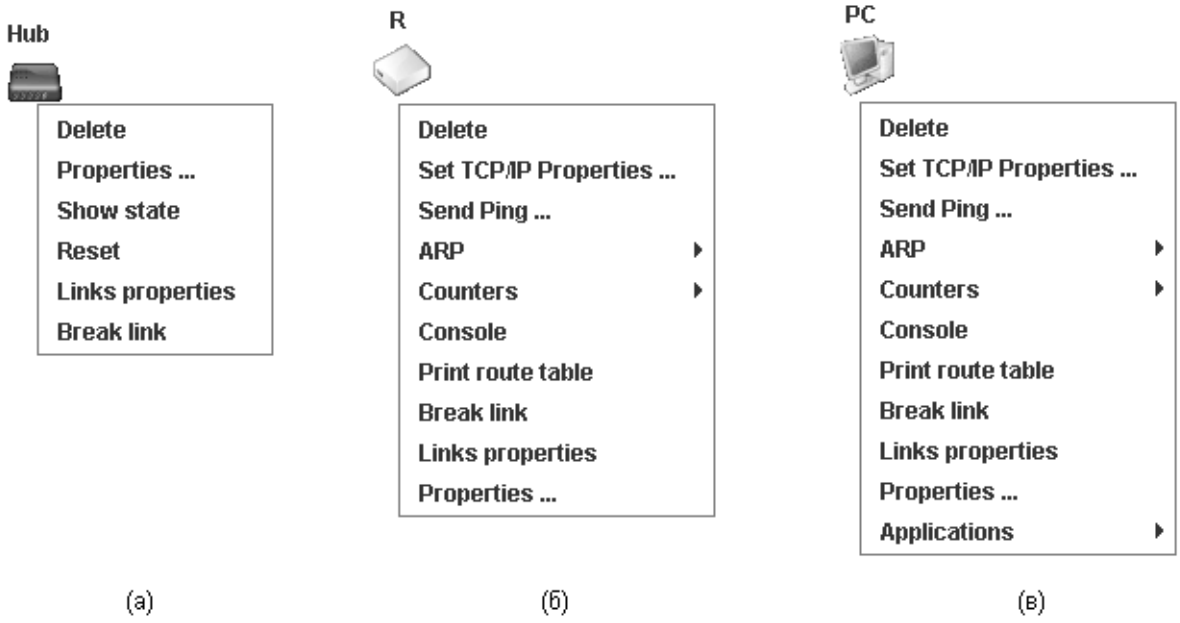


Рис. 6.3 Контекстное меню

Основные пункты контекстного меню, общие для всех устройств перечислены ниже.

- Delete(Удалить) – без подтверждения удаляет выбранное сетевое устройство из текущей конфигурации.
- Properties(Свойства) – вызывает диалог, показывающий сетевые настройки выбранного устройства. Для каждого интерфейса показывается MAC адрес, IP адрес, маска подсети, название подключенной линии связи. Также для устройства указаны имя и шлюз по умолчанию.
- Break Link(Разорвать линию связи) – вызывает диалог, в котором можно выбрать интерфейс, линию связи которого требуется разорвать.
- Links Properties(Свойства линий связи) – позволяет установить свойства линии связи.

При выборе пункта Link Properties(Свойства линий связи) вызывается диалог, который позволяет установить коэффициент пропускания

для интерфейса, показывающий какой процент пакетов линия связи подключенная к этому интерфейсу будет пропускать. Коэффициент пропускания задается для интерфейса (eth0, eth1 и т.д.).
В меню концентратора имеются два дополнительных пункта, позволяющих следить за его состоянием и, в случае необходимости, восстанавливать исходное состояние.

- Show state(Показать состояние) – показывает текущее состояние концентратора, может принимать два значения normal(концентратор работает) и freezed(концентратор был остановлен из-за ошибки).
- Reset(Перезагрузить) – если концентратор находится в состоянии останова, то эта команда вернет его в рабочее состояние.

В меню устройств работающих на сетевом уровне (маршрутизаторы и компьютеры) в дополнение к основным имеются ещё шесть пунктов:

- Set TCP/IP Properties(Установка свойств TCP/IP) – вызывает диалог позволяющий изменить свойства TCP/IP;
- Send Ping...(Послать эхо-запрос) – позволяет послать эхо-запрос адресату;
- ARP – подменю позволяет работать с таблицей протокола ARP на выбранном устройстве;
- Counters – подменю содержит два пункта:
  - Show Packet Counters(Показать счетчики пакетов) – показывает счетчики для пакетов протоколов ARP, IP, UDP, TCP
  - Reset Packet Counters(Сбросить счетчики пакетов) – устанавливает все счетчики на выбранном устройстве в ноль;
- Console – вызывает командную строку, позволяющую настраивать таблицы маршрутизации, ARP таблицы и др.;
- Print route table(Показать таблицу маршрутизации) – выводит на вкладку "консоль" таблицу маршрутизации выбранного сетевого устройства.

Пункт контекстного меню Send Ping...(Послать эхо-запрос) – вызывает диалог, в котором можно настроить параметры эхо-запроса. Во время передвижения пакетов по сети во вкладке "консоль" должны появится сообщения, аналогичные приведенным ниже:

```
PC1 Echo Request Packet Network  Created Echo
                                 Request packet to 10.0.0.2
...
PC1 Echo Reply Packet   Network  Echo reply packet
```

```
received from 10.0.0.2
```

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 заноситься 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.

В контекстное меню компьютеров, т.е. устройств поддерживающих уровень приложений добавляется еще один пункт: подменю Applications(Приложения), которое позволяет работать с протоколами: Echo(UDP,TCP), SNMP и TELNET.

6.3. Командная строка

Для запуска командной строки из контекстного меню выберем "Console", появится окно консоли (рисунок 6.4). Окно разделено на 2 части:

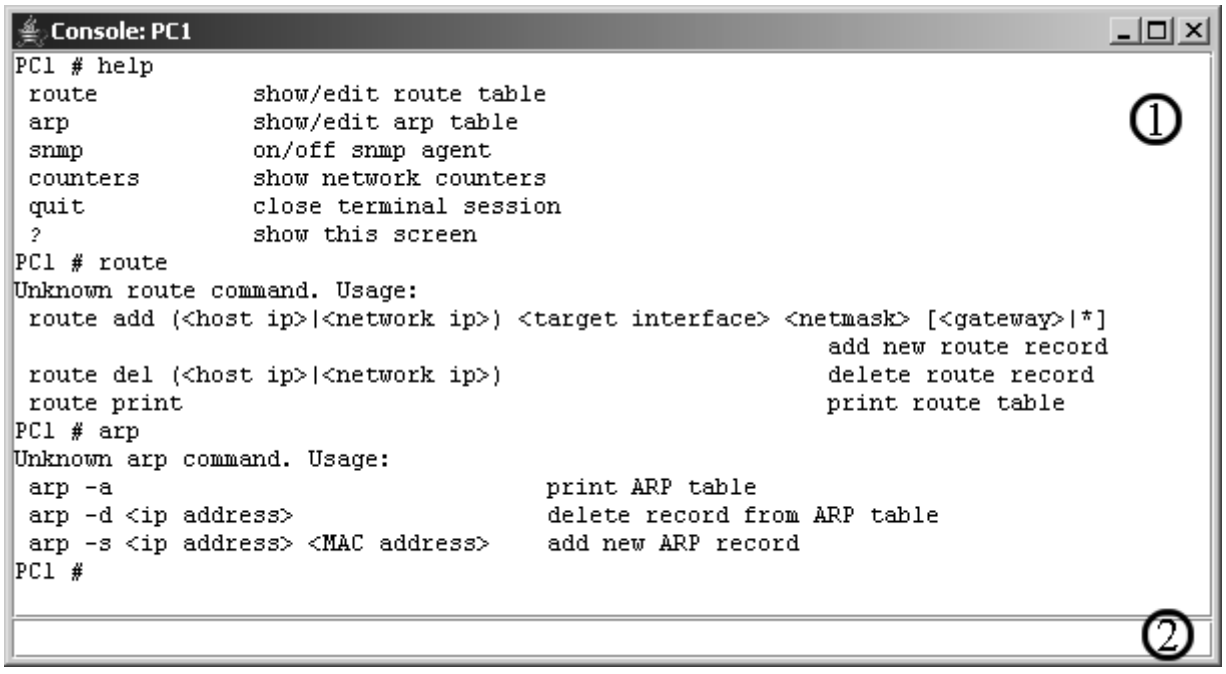


Рис. 6.4 Консоль

Цифры на рисунке 6.4 обозначают следующее:

1 – область для сохранения результата выполнения команд;

2 – командная строка, в которой можно вводить команды на выполнение.
В консоли могут использоваться следующие специальные клавиши:

- <Enter> – выполнить введенную команду;
- стрелки вверх/вниз – просмотр истории команд;
- <ESC> – очистить командную строку;
- Ctrl+D – закрыть консоль.

В командной строке с помощью команды *route* можно выполнить настройку статической таблицы маршрутизации. Для этого предназначена команда *route*.
Описание синтаксиса команды *route*:

- *route add (<ip адрес устройства>|<ip адрес сети>) <интерфейс> <маска подсети> [<шлюз>| *]* – добавить новый маршрут для сети или устройства;
- *route del (<ip адрес устройства>|<ip адрес сети>)* – удалить существующий маршрут для указанного IP адреса устройства или сети;
- *route print* – просмотреть список существующих маршрутов.

с помощью команды *arp* можно выполнить настройку таблицы ARP:

- *arp -a* просмотреть ARP таблицу;
- *arp -d <ip address>* удалить из ARP таблицы запись об IP адресе;
- *arp -s <ip address> <MAC address>* добавить ARP запись связывающую IP и MAC адреса.

с помощью команды *snmp* можно управлять snmp агентом:

- *snmp (on|<port number>) [community name]* включить SNMP агента. Если порт не указан (значение *on*), то по умолчанию выбирается порт 161. Если не указано имя группы доступа, то берется значение по умолчанию *public*;
- *snmp off* выключить SNMP агента.

При вводе команд *route*, *arp* и *snmp* без параметров будет выведена краткая информация по их использованию.

6.4. Работа 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 и TELNET.

6.4.1. Работа с протоколом Echo

Имитаторе javaNetSim позволяет использовать протоколы UDP или TCP в качестве транспортных протоколов для протокола Echo. Для установки echo-сервера в режим прослушивания порта в контекстном

меню надо выбрать пункт: "Applications > Start udp echo server to listen" для Echo-UDP или "Applications > Start tcp echo server to listen" для Echo-TCP. После этого в появившемся диалоговом окне следует вести номер порта, на котором выбранное приложение будет ожидать сообщения. После этого с любого другого узла можно отсылать сообщения на тот узел, на котором запущен echo-сервер и получать ответы. Для того, чтобы послать эхо-запрос, необходимо в контекстном меню выбрать "Applications > Send data via udp echo client" для Echo-UDP или "Applications > Send data via tcp echo client" для Echo-TCP и ввести четыре параметра:

- IP-адрес компьютера, на котором запущен echo-сервер;
- номер порта на котором echo-сервер ожидает сообщения;
- сообщение – любой текст;
- количество посылаемых сообщений, т.е. количество копий сообщения отправляемых echo-серверу.

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.

6.4.2. Работа с протоколом SNMP

В имитаторе javaNetSim предусмотрено несколько функций для работы с протоколом SNMP:

- запуск SNMP агента на объекте управления;
- остановка SNMP агента на объекте управления;
- посылка SNMP запросов агенту.

Для запуска SNMP агента необходимо выбрать пункт контекстного меню "Application > Start SNMP Agent" и задать два параметра:

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

Для остановки SNMP агента необходимо выбрать пункт контекстного меню "Application > Stop SNMP Agent".

Для того, чтобы послать запрос SNMP агенту необходимо выбрать пункт контекстного меню "Application > Send SNMP message" и заполнить поля диалога, приведенные на рисунке 6.5).

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

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

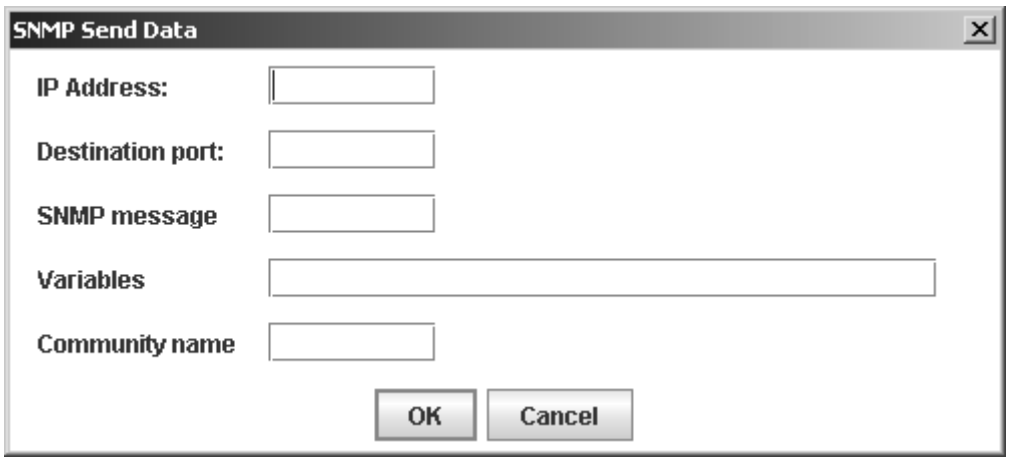


Рис. 6.5 Создание SNMP запроса.

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 "консоль". Например:

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

Список SNMP переменных, поддерживаемых имитатором javaNet-Sim, которые имеют режим доступа "только для чтения"приведен ниже.

- 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.ARPTable – 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.

6.4.3. Работа с протоколом TELNET

В имитаторе javaNetSim предусмотрены следующие функции для работы с протоколом TELNET:

- запуск TELNET сервера на управляемом компьютере;
- остановка TELNET сервера;
- запуск TELNET клиента.

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 остановки TELNET сервера необходимо выбрать пункт контекстного меню "Application > Stop telnet server".

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 сеансе telnet доступны следующие команды:

- *route* – просмотр и редактирование сетевых маршрутов;
- *arp* – просмотр и редактирование ARP таблиц;
- *snmp* – запуск о 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;

После завершения работы необходимо закрыть сеанс telnet. Закрытие сеанса telnet можно произвести тремя способами:

- набрать команду quit.
- нажать комбинацию клавиш Ctrl+D.
- просто закрыть окно терминала.

Несмотря на то, что протокол TELNET в javaNetSim реализован на очень простом уровне, это не мешает ему выполнять свои функции. В качестве примера можно привести подключения telnet-клиента к Echo-TCP-серверу.

Список рекомендуемой литературы

1. TCP/IP: учебный курс. Официальное пособие Microsoft по сертификационному экзамену MSCE 70-059. — М.: Русская редакция, 1999. — 344 с.
2. *Гладцын В. А., Кринкин К. В., Яновский В. В.* Программирование протоколов стека TCP/IP в ОС Windows NT 4.0: Практикум. — СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2002. — 72 с.
3. *Гладцын В. А., Яновский В. В.* Сетевые технологии: Учеб. пособие / СПбГЭТУ «ЛЭТИ». — СПб., 1998. — 116 с.
4. *Гук М.* Аппаратные средства локальных сетей. — СПб.: Питер, 2002. — 576 с.
5. *Кирх О.* Linux: Руководство Администратора Сети. — СПб.: Питер, 2000. — 368 с.
6. *Кульгин М.* Технологии корпоративных сетей: Энциклопедия. — СПб.: Питер, 2000. — 704 с.
7. *Леинванд А., Пински Б.* Конфигурирование маршрутизаторов Cisco. — М.: Издательский дом «Вильямс», 2004. — 368 с.
8. *Олифер В., Олифер Н.* Компьютерные сети: учебник. — СПб.: Питер, 2001. — 672 с.
9. Программа сетевой академии Cisco CCNA 1 и 2. Вспомогательное руководство. — М.: Издательский дом «Вильямс», 2005. — 1168 с.
10. *Танненбаум Э.* Основы компьютерных сетей. — СПб.: Питер, 2005. — 992 с.
11. *Шиндер Д. Л.* Основы компьютерных сетей. — М.: Издательский дом «Вильямс», 2004. — 656 с.

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-сокет 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.

Демультиплексирование 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.

Оглавление

Исследование уровней организации IP-сетей

Лабораторный практикум

Редактор И. Б. Синишева

Подписано в печать xx.xx.xx. Формат 60x84$^{1}/_{16}$. Бумага офсетная.
Печать офсетная. Гарнитура "Arial Cyr". Печ. л. x,xx.
Тираж xxx экз. Заказ

Издательство СПбГЭТУ "ЛЭТИ"
197376, С.-Петербург, ул. Проф. Попова, 5