ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М. А. БОНЧ-БРУЕВИЧА»

С. С. Владимиров, И. А. Небаев

ИНТЕРНЕТ-ТЕХНОЛОГИИ И МУЛЬТИМЕДИА

Лабораторный практикум

СПб ГУТ)))

Санкт-Петербург
2015

УДК 004.738.5 (076)
ББК 32.97 я73
В 57

Рецензент
заведующий кафедрой ОПДС, профессор,
доктор технических наук *О. С. Когновицкий*

Рекомендовано к печати редакционно-издательским советом СПбГУТ

Владимиров, С. С.

В 57 Интернет-технологии и мультимедиа : лабораторный практикум / С. С. Владимиров, И. А. Небаев ; СПбГУТ. — СПб, 2015. — 59 с.

Учебное пособие призвано ознакомить студентов старших курсов с интернет-технологиями. Представленный материал служит справочным и методическим пособием при выполнении курса практических лабораторных работ по дисциплинам «Интернет-технологии и мультимедиа», «Интернет-технологии» и «Мультимедийные технологии и протоколы».

Предназначено для студентов, обучающихся по направлениям 11.03.02 (210700.62) «Инфокоммуникационные технологии и системы связи», 09.03.01 (230100.62) «Информатика и вычислительная техника» и 09.03.04 (231000.62) «Программная инженерия».

УДК 004.738.5 (076)
ББК 32.97 я73

Содержание

Лабораторная работа 1
Изучение принципов работы утилит для исследования и мониторинга состояния сети

1.1. Цель работы

Получение базовых навыков по использованию основных сетевых утилит, применяемых для исследования и мониторинга состояния сети. Изучение принципов их работы.

1.2. Задачи

Проверить доступность заданных (согласно варианту) узлов в сети Интернет и различными способами проверить маршрут прохождения пакетов до этих узлов.

1.3. Теоретический материал

Существует огромный спектр различных утилит, предназначенных для работы с сетью и сетевыми интерфейсами. Одни из них используются при настройке сетевых интерфейсов компьютера. Другие — для анализа текущего состояния локальной и глобальной компьютерных сетей. Третьи — для мониторинга состояния компьютерных сетей. Сложность этих утилит также разнится. Используются как узкоспециализированные утилиты, так и программы-«комбайны», решающие сразу набор задач. Выбор используемого программного обеспечения зависит от стоящих задач. В данной работе предлагается рассмотреть работу ряда широко используемых узкоспециализированных сетевых утилит.

1.3.1. Утилита `ifconfig`

Утилита `ifconfig` предназначена для настройки сетевых интерфейсов ПК и для проверки их состояния. Утилита считается устаревшей, но тем не менее до сих пор широко используется. Ей на замену пришла утилита `ip`, входящая в набор программ `iproute2`.

Для настройки сетевого интерфейса утилитой `ifconfig` необходимо иметь права суперпользователя. Просмотр настроек и состояния сетевых интерфейсов возможен от лица обычного пользователя.

От лица обычного пользователя утилита `ifconfig` запускается следующим образом:

```
user@host:[~]$ /sbin/ifconfig
```

В результате на экран будут выведены параметры работающих в данный момент сетевых интерфейсов ПК. Для того, чтобы просмотреть настройки всех интерфейсов, необходимо запустить программу с флагом `-a`:

```
user@host:[~]$ /sbin/ifconfig -a
```

Более подробную информацию об использовании утилиты можно узнать из официальной man-страницы. Команда:

```
user@host:[~]$ man ifconfig
```

1.3.2. Утилита `ping`

Утилита `ping` предназначена для тестирования сетей, управления сетями и измерения производительности. Из-за нагрузок, которые она создает в сети, не всегда разумно использовать ping в рабочее время или в автоматических сценариях.

Она отправляет запросы (ICMP Echo-Request) протокола ICMP указанному узлу сети и фиксирует поступающие ответы (ICMP Echo-Reply). Время между отправкой запроса и получением ответа (RTT — Round Trip Time) позволяет определять двусторонние задержки (RTT) по маршруту и частоту потери пакетов, т. е. косвенно определять загруженность на каналах передачи данных и промежуточных устройствах.

Полное отсутствие ICMP-ответов может также означать, что удаленный узел (или какой-либо из промежуточных маршрутизаторов) блокирует ICMP Echo-Reply или игнорирует ICMP Echo-Request.

Название происходит от английского названия звука импульса, издаваемого сонаром при отражении импульса от объекта.

Также есть несколько альтернативных толкований:

- PING — акроним «Packet InterNet Grouper (Groper)»;
- Ping — часть названия игры пинг-понг. Это толкование подразумевает, что компьютеры обмениваются сигналами аналогично тому, как игроки в пинг-понг отбивают друг другу мяч.

Утилита `ping` запускается в терминале. Формат запуска команды:

```
user@host:[~]$ ping [options] host
```

Пример: запуск утилиты для проверки хоста `www.google.ru`:

```
user@host:[~]$ ping www.google.ru
```

Таблица 1.1

Основные опции (флаги) утилиты `ping`

Опция/флаг	Описание
`-b`	Разрешить использование широковещательного адреса в качестве целевого
`-c` *количество*	Остановить работу после передачи заданного количества пакетов `ECHO REQUEST`. Если задано *ограничение-на-время-работы* (`-w`), программа будет ждать указанное количество ответных пакетов `ECHO REPLY` в указанный временной интервал
`-i` *интервал*	Интервал в секундах между отправкой пакетов. По умолчанию между отправкой пакетов делается пауза в 1 с, либо, в случае лавинообразного режима, отправка производится без пауз. Задавать значения меньше 0,2 может только суперпользователь
`-L`	Подавлять циклические петли для широковещательных пакетов. Этот ключ применяется только если в качестве целевого адреса указан широковещательный
`-n`	Только цифровой вывод. Не расшифровывать имена (символьный вид) адресов
`-q`	Выводить только начальные и итоговые данные (не выводить информацию об отдельных запросах)
`-s` *размер-пакета*	Размер пакетов для пересылки. По умолчанию — 56, что соответствует размеру 64 байта после добавления 8 байтов заголовка ICMP
`-t` *TTL*	Время актуальности пакета IP (TTL — Time to Live)
`-w` *огр.-на-вр.-раб.*	Время, по истечении которого ping завершит свою работу независимо от количества посланных и принятых пакетов. При указании этого параметра время ожидания для одного пакета игнорируется и работа может быть завершена ранее указанного срока только в случае получения информации об ошибке (т. е. уведомления о том, что ответных пакетов точно не будет)
`-W` *время-ожид.-ответа*	Время ожидания (в секундах) ответного пакета. Принимается во внимание только если не было принято ни одного ответа. В противном случае программа ожидает получения двух ответов

Пример: запуск утилиты для проверки доступности хоста `8.8.8.8`; остановить работу после передачи 6 пакетов:

```
user@host:[~]$ ping -c 6 8.8.8.8
```

Более подробную информацию об использовании утилиты можно узнать из официальной man-страницы. Команда:

```
user@host:[~]$ man ping
```

1.3.3. Утилита `traceroute`

Программа `traceroute` предназначена для определения маршрутов следования данных в сетях TCP/IP. Она основана на использовании протоколов ICMP и UDP, а также механизма TTL (Time to Live).

`Traceroute` выполняет отправку данных указанному узлу сети, при этом отображая сведения о всех промежуточных маршрутизаторах, через которые прошли данные на пути к целевому узлу. В случае проблем при доставке данных до какого-либо узла программа позволяет определить, на каком именно участке сети возникли неполадки. Программа работает только в направлении от источника пакетов и является весьма грубым инструментом для выявления неполадок в сети. В силу особенностей работы протоколов маршрутизации в сети Интернет, обратные маршруты часто не совпадают с прямыми, причем это справедливо для всех промежуточных узлов. Поэтому, ICMP ответ от каждого промежуточного узла может идти своим собственным маршрутом, затеряться или прийти с большой задержкой, хотя в реальности с пакетами, которые адресованы конечному узлу, этого не происходит. Кроме того, на промежуточных маршрутизаторах часто стоит ограничение числа ответов ICMP в единицу времени, что приводит к появлению ложных потерь.

Утилита `traceroute` входит в поставку большинства современных сетевых операционных систем. В системах Microsoft Windows эта программа носит название `tracert`, а в системах GNU/Linux, Cisco IOS и Mac OS — `traceroute`.

Для определения промежуточных маршрутизаторов `traceroute` отправляет серию (обычно три) UDP датаграмм целевому узлу, при этом каждый раз увеличивая на 1 значение поля TTL («время жизни») в заголовке IP-пакета. Это поле указывает максимальное количество маршрутизаторов, которое может быть пройдено пакетом. Первая серия пакетов отправляется с TTL, равным 1, и поэтому первый же маршрутизатор возвращает обратно сообщение ICMP «время жизни истекло» (тип 11), указывающее на невозможность доставки данных. `Traceroute` фиксирует адрес маршрутизатора, а также время между отправкой пакета и получением ответа (эти сведения выводятся на монитор компьютера). Затем `traceroute` повторяет отправку серии пакетов, но уже с TTL, равным 2, что позволяет первому маршрутизатору пропустить их дальше.

Процесс повторяется до тех пор, пока при определенном значении TTL пакет не достигнет целевого узла. При получении ответа от этого узла процесс трассировки считается завершенным.

На оконечном хосте IP-пакет с TTL $= 1$ не отбрасывается и должен быть передан приложению. Достижение пункта назначения определяется следующим образом: отсылаемые `traceroute` пакеты содержат датаграмму UDP с

таким номером порта адресата (превышающим 30 000), что он заведомо не используется на адресуемом хосте. В пункте назначения модуль UDP, получая подобные дейтаграммы, возвращает сообщения ICMP «порт недоступен» (тип 3, код 3). Таким образом, чтобы узнать о завершении работы, программе `traceroute` достаточно обнаружить, что поступило сообщение ICMP об ошибке этого типа.

Формат запуска команды:

```
user@host:[~]$ traceroute [options] host
```

Пример: запуск утилиты для определения маршрутов следования данных до хоста `www.sut.ru`

```
user@host:[~]$ traceroute www.sut.ru
```

Результат работы команды показан в лист. 1.1.

Листинг 1.1

Результат работы утилиты `traceroute`

```
traceroute to www.sut.ru (85.142.45.77), 30 hops max, 40 byte
   packets
 1   (172.16.100.19)  0.241 ms  0.341 ms  0.448 ms
 2   (172.16.100.2)  3.345 ms  3.424 ms  3.502 ms
 3  172.12.12.2 (172.12.12.2)  3.580 ms  3.658 ms  4.321 ms
 4  84.204.14.193 (84.204.14.193)  4.717 ms  4.806 ms  4.885 ms
 5  82.196.95.177 (82.196.95.177)  5.602 ms  6.605 ms  6.681 ms
 6  82.196.95.165 (82.196.95.165)  6.761 ms  6.984 ms  7.031 ms
 7  194.190.255.209 (194.190.255.209)  7.345 ms  4.423 ms  2.612 ms
 8  194.190.255.142 (194.190.255.142)  3.044 ms  3.138 ms  3.216 ms
 9  gw-run.sut.ru (85.142.44.2)  4.061 ms  4.161 ms  4.467 ms
10  hub.segmenta.ru (85.142.45.77)  8.503 ms  8.581 ms  9.208 ms
```

Можно видеть время достижения каждого из промежуточных узлов для каждого из трех отправляемых пакетов.

Более подробную информацию об использовании утилиты можно получить из официальной man-страницы. Команда:

```
user@host:[~]$ man traceroute
```

1.3.4. Утилита `mtr`

`Mtr` (My Traceroute) — это служебная компьютерная программа, предназначенная для определения маршрутов следования данных в сетях TCP/IP. Она постоянно показывает сведения о маршруте, потерях, минимальной и максимальной задержке. При помощи `mtr` можно узнавать, где в сети происходят потери, задержки и обрывается связь.

При запуске `mtr` начинает исследование сетевого соединения между локальной машиной и хостом, заданным пользователем. После того, как она определит адрес каждого транзитного узла на пути следования пакетов, она отправляет каждому из этих узлов последовательности запросов ICMP ECHO, пытаясь определить качество канала связи с каждым из них. По окончании исследования выдается статистика по каждому исследованному узлу.

Формат запуска команды:

```
user@host:[~]$ mtr [options] host
```

Таблица 1.2

Основные опции (флаги) утилиты `mtr`

Опция/флаг	**Описание**
`-r`	Переводит `mtr` в режим report. В этом режиме `mtr` отрабатывает количество циклов, заданное флагом `-c`, затем выводит статистику и завершает работу
`-c` *число*	Задает количество циклов опроса маршрута. Каждый цикл длится 1 с
`-i` *число*	Задает время в секундах на один цикл опроса. По умолчанию — 1 с

Пример: запуск утилиты для определения маршрута следования данных до хоста `www.sut.ru` (в режиме report с двумя циклами опроса):

```
user@host:[~]$ mtr -c 2 -r www.sut.ru
```

Результат работы команды показан в лист. 1.2.

Листинг 1.2

Результат работы утилиты `mtr`

```
HOST: opds                      Loss%   Snt   Last   Avg  Best  Wrst StDev
  1. 172.16.100.19               0.0%     2    0.2   0.2   0.2   0.2   0.0
  2. 172.16.100.2                0.0%     2    0.7   0.7   0.7   0.7   0.0
  3. 172.12.12.2                 0.0%     2    1.0   1.1   1.0   1.1   0.1
  4. 84.204.14.193               0.0%     2    1.3   1.4   1.3   1.4   0.0
  5. 82.196.95.177               0.0%     2    1.5   1.5   1.5   1.5   0.0
  6. M120-1-321-SPB-RU.xe-0-0    0.0%     2    1.8   1.8   1.8   1.8   0.0
  7. bm18-1-gw.spb.runnet.ru     0.0%     2    2.2   2.2   2.2   2.2   0.0
  8. sut.spb.runnet.ru           0.0%     2    2.6   2.6   2.6   2.6   0.0
  9. gw-run.sut.ru               0.0%     2    3.7   3.9   3.7   4.1   0.3
 10. hub.segmenta.ru             0.0%     2    4.5   4.1   3.7   4.5   0.6
```

1.3.5. Программа `tracemap`

`Tracemap` — программа, позволяющая выполнить трассировку пути на несколько хостов сразу и представить полученные данные в виде графической карты. Она написана Игорем Чубиным на языке Perl. В лабораторной работе

используется измененная версия программы, не требующая прав суперпользователя.

Программу `tracemap` необходимо скачать с локального сервера кафедры. Для этого можно воспользоваться командой:

```
user@host:[~]$ wget http://opds.spbsut.ru/data/_uploaded/prog/
   tracemap.pl
```

Программа будет сохранена в текущем каталоге.

Формат запуска команды:

```
user@host:[~]$ (echo host) | perl tracemap.pl
```

В результате выполнения программы в текущем каталоге появятся файлы `tracemap.svg` и `tracemap.png` с графической картой путей.

Пример: запуск утилиты для построения карты маршрутов до хостов `www.sut.ru` и `www.ya.ru`:

```
user@host:[~]$ (echo www.sut.ru; echo www.ya.ru) | perl tracemap.pl
```

1.4. Порядок выполнения лабораторной работы

1. Просмотреть параметры сетевого интерфейса лабораторного ПК, воспользовавшись утилитой `ifconfig`. Выведенную на экран информацию сохранить для отчёта.

Важно: Для того, чтобы сразу записать вывод команды в файл, можно воспользоваться утилитой `tee`. Пример использования для данного задания:

```
user@host:[~]$ /sbin/ifconfig | tee it-lab01-ifconfig.txt
```

2. С помощью утилиты `ping` проверить состояние связи с узлами, заданными в табл. 1.3. Результаты выполнения сохранить для отчета. Количество отправляемых до каждого узла «эхо-запросов» следует ограничить 4–5 пакетами.

По результатам проверки состояния связи заполнить таблицу, приведенную ниже.

Доменное имя	IP-адрес	Страна	Число потерянных запросов	Среднее время прохождения запроса	TTL
…	…	…	…	…	…

3. При помощи утилиты `traceroute` произвести трассировку узлов, заданных в табл. 1.3. Результаты протоколировать в файл.

По результатам трассировки составить графики времени прохождения маршрутизаторов для каждого узла (для 3 пакетов), указать наиболее узкие места в сети.

Таблица 1.3

Исследуемые узлы в сети Internet
(номер варианта выбирается по последней цифре номера студ. билета)

№ варианта	Исследуемые узлы	№ варианта	Исследуемые узлы
0	www.yandex.ru www.gismeteo.ua www.uefa.com	5	tv.sut.ru www.iana.org www.jp-australia.com
1	meta.ua www.pl.md adobe.com	6	www.rambler.ru www.cisco.com opera.com
2	www.ubuntu.ru www.vlc.ru www.japantoday.com	7	www.kontest.ru www.ewm1.pl www.nic.ad.jp
3	www.industry.su www.tdpoland.pl www.oracle.com	8	linux.org.ru www.skype.com www.runtu.org
4	mob.ua www.unfoundation.org www.viruslist.com	9	www.gaw.ru www.drweb.com www.o2.pl

4. Получить маршрут прохождения пакетов до одного из заданных в варианте узлов при помощи утилиты `ping`. Результаты протоколировать в файл.

Для выполнения этого задания необходимо последовательно посылать «эхо-запросы» на искомый узел, последовательно увеличивая параметр TTL на 1. Начиная с TTL = 1 и заканчивая TTL, на котором будет достигнут искомый узел. Количество отправляемых на каждом шаге пакетов следует ограничить 2–3.

5. Определить маршрут прохождения пакетов до узла, выбранного в предыдущем пункте при помощи утилиты `mtr`. Результаты протоколировать в файл.

Провести сравнение результатов определения маршрута, полученных с помощью утилит `traceroute`, `ping` и `mtr`.

Важно: Утилита `mtr` должна быть запущена в режиме report с ограничением количества циклов опроса маршрута.

6. Построить графическую карту трассировки одновременно ко всем заданным в табл. 1.3 узлам при помощи программы `tracemap`.

1.5. Содержание отчета

1. Заголовок, согласно приложению.
2. Цель работы.
3. Задание, согласно варианту.
4. Листинги результатов работы:
   а) листинг параметров сетевого интерфейса лабораторного ПК;

б) листинг результатов, полученных при работе с утилитой `ping`, плюс таблица с результатами исследований при использовании утилиты `ping`;

в) листинг результатов, полученных при работе с утилитой `traceroute`, плюс графики времени прохождения шлюзов (для 3 пакетов) с указанием наиболее узких мест в сети;

г) листинг результатов, полученных при определении маршрута прохождения пакетов утилитой `ping`;

д) листинг результатов, полученных при работе с утилитой `mtr`, и привести сравнение результатов определения маршрута, полученных с помощью утилит `traceroute`, `ping` и `mtr`.

Важно: Перед каждым листингом результатов работы должна быть написана соответствующая команда.

5. Карта трассировки, полученная при помощи утилиты `tracemap`.

1.6. Контрольные вопросы

1. Утилита `ping`. Назначение и принцип работы.
2. Утилита `traceroute`. Назначение и принцип работы.
3. Утилита `mtr`. Назначение и принцип работы.
4. Механизм TTL. Назначение и принцип работы.
5. Протокол ICMP.
6. Формат пакета ICMP.
7. Виды пакетов ICMP.

Лабораторная работа 2
Работа с программным анализатором протоколов `tcpdump`

2.1. Цель работы

Получение базовых навыков по работе с анализатором протоколов `tcpdump`. Изучение принципов фильтрации пакетов.

2.2. Задачи

Захватить при помощи анализатора протоколов `tcpdump` заданные сетевые пакеты.

2.3. Теоретический материал

`Tcpdump` (от TCP и англ. dump — свалка, сбрасывать) — утилита UNIX, позволяющая перехватывать и анализировать сетевой трафик, проходящий через компьютер, на котором запущена данная программа.

Программа `tcpdump` была написана Van Jacobson, Craig Leres и Steven McCanne, во время их работы в лаборатории Lawrence Berkeley, Калифорнийского университета, Беркли.

Программа `tcpdump` разработана таким образом, чтобы переводить сетевую плату в смешанный режим (promiscuous mode), при этом, каждый пакет, проходящий по кабелю, фиксируется. Обычно сетевые платы для сред передачи, таких как Ethernet, захватывают только фреймы канального уровня, адресованные конкретному интерфейсу или отправленные на широковещательный адрес.

Операционная система должна позволить поместить интерфейс в смешанный режим и позволить пользовательскому процессу захватывать фреймы. Поэтому для выполнения программы требуется наличие прав суперпользователя и прямой доступ к устройству.

Основные назначения `tcpdump`:

- отладка сетевых приложений;
- отладка сети и сетевой конфигурации в целом.

Утилита `tcpdump` запускается в терминале от имени суперпользователя. Формат запуска команды:

```
user@host:[~]$ sudo tcpdump [options] [filter]
```

Фильтр необходим для того, чтобы отображать на экране (или сохранять в файл) только определенные пакеты (т. е. те, которые соответствуют фильтру). Фильтр представляет из себя один или несколько примитивов филь-

трации, объединенных логическими операторами. Предпочтительно (но не обязательно) записывать строку фильтрации в апострофах.

Таблица 2.1

Основные опции (флаги) утилиты `tcpdump`

Опция/флаг	**Описание**
`-i`	Указывает на то, какой сетевой интерфейс будет использоваться для захвата пакетов. По-умолчанию — `eth0`
`-w` *имя-файла*	Сохраняет данные `tcpdump` в двоичном формате. Преимуществами использования данного способа по сравнению с обычным перенаправлением в файл являются высокая скорость записи и возможность чтения подобных данных другими программами, например, `snort` или `wireshark`, но этот файл нельзя прочитать человеку
`-r` *имя-файла*	Этот параметр позволяет tcpdump прочесть трафик из файла, если он был предварительно сохранен параметром `-w`
`-x`	Делает распечатку пакета в шестнадцатеричной системе, полезно для более детального анализа пакета. Количество отображаемых данных зависит от опции `-s`
`-xx`	То же, что и предыдущий параметр, но включает в себя заголовок канального уровня
`-X`	Выводит пакет в ASCII- и hex-формате. Полезно в случае анализа инцидента, связанного со взломом, так как позволяет просмотреть, какая текстовая информация передавалась во время соединения
`-XX`	То же, что и предыдущий параметр, но включает заголовок канального уровня
`-s` *число*	Количество байтов пакета, которые будет обрабатывать `tcpdump`. При установке большого числа отображаемых байтов информация может не уместиться на экране, и ее будет трудно изучать. В зависимости от того, какие цели вы преследуете, и следует выбирать значение этого параметра. По-умолчанию `tcpdump` сохраняет первые 68 байт, однако если вы хотите увидеть содержимое всего пакета, используйте значение в 1500 байт (максимально допустимый размер пакета в сети Ethernet)
`-c` *число*	`tcpdump` завершит работу после получения указанного числа пакетов
`-v`	Вывод подробной информации (TTL; ID; общая длина заголовка, а также его параметры; производит проверку контрольных сумм IP- и ICMP-заголовков)
`-vv`	Вывод еще более полной информации, в основном касается NFS и SMB
`-vvv`	Вывод максимально подробной информации
`-l`	Использование стандартного потокового вывода `tcpdump` (stdout), например, для записи в файл: `tcpdump -l | tee out.log` отобразит работу `tcpdump` и сохранит результат в файле `out.log`

Таблица 2.2

Примитивы фильтрации пакетов утилиты `tcpdump`

Примитив	**Описание**
`dst host` *хост*	Будет отбирать пакеты, в которых поле адреса получателя IPv4/v6 содержит адрес хоста, заданного в примитиве
`src host` *хост*	Будет выбирать все пакеты, в которых поле отправителя содержит адрес указанного хоста
`host` *хост*	Будет отбирать все пакеты, для которых адрес хоста указан в поле получателя или отправителя
`ether dst` *MAC*	Будет выбирать все кадры, в которых поле MAC-адреса получателя содержит значение MAC
`ether src` *MAC*	Будет выбирать все кадры, в которых поле MAC-адреса отправителя содержит значение MAC
`ether host` *MAC*	Будет отбирать все пакеты с адресом, указанным значением MAC в поле отправителя или получателя
`gateway` *хост*	Будет отбирать все пакеты, использующие указанный именем хост в качестве шлюза
`dst net` *сеть*	Отбирает все пакеты IPv4/v6, направленные в указанную сеть
`src net` *сеть*	Выбирает все пакеты IPv4/v6, отправленные из указанной сети
`net` *сеть*	Выбирает все пакеты IPv4/v6, содержащие адреса из указанной сети в поле отправителя или получателя
`dst port` *порт*	Отбирает все пакеты IP/TCP, IP/UDP, IPv6/TCP и IPv6/UDP, направленные в указанный порт
`src port` *порт*	Отбирает все пакеты, отправленные из указанного порта
`port` *порт*	Отбирает все пакеты, содержащие указанный номер порта в поле отправителя или получателя. Любое из трех перечисленных правил для портов может включать в качестве префикса идентификатор протокола TCP или UDP (например, `tcp src port` порт, будет отбирать пакеты TCP, отправленные из указанного порта)
`less` *размер*	Будет собирать пакеты, размер которых не превышает указанного значения
`greater` *размер*	Будет собирать пакеты, размер которых не меньше указанного значения
`ip proto` *протокол*	Отбирает все пакеты IP, содержащие заданный идентификатор типа в поле типа протокола (icmp, icmp6, igmp, igrp, pim, ah, esp, vrrp, udp, tcp). Поскольку tcp, udp и icmp используются также в качестве ключевых слов, перед этими идентификаторами следует помешать символ «\». Этот примитив не проверяет цепочки протокольных заголовков
`ether proto` *прот.*	Отбирает кадры Ethernet с заданным типом протокола (ip, ip6, arp, rarp, atalk, aarp, decnet, sca, lat, mopdl, moprc, iso, stp, ipx, netbeui)

Примитивы фильтрации могут группироваться с помощью логических операторов:

- скобки;
- отрицание (`!` или `not`);
- логическое пересечение (`&&` или `and`);
- логическое объединение (`||` или `or`).

Оператор отрицания имеет высший уровень приоритета, операции объединения и пересечения имеют одинаковый приоритет и выполняются слева направо в порядке следования. Для операции логического пересечения недостаточно просто указать операнды рядом, а требуется явно задать операцию (`&&` или `and`).

В качестве примера рассмотрим использование `tcpdump` для перехвата трафика, создаваемого утилитой `traceroute`.

`Traceroute` использует пакеты протокола UDP в качестве «запросов» и получает пакеты протокола ICMP в качестве «ответов». Таким образом, фильтр для отсеивания пакетов будет состоять из двух частей: фильтр перехвата «запросов» и фильтр перехвата «ответов». Ограничим количество захватываемых пакетов двадцатью пятью. Команда будет иметь вид:

```
user@host:[~]$ sudo tcpdump -i eth0 -c 25 '( dst host target-IP \
> and src host local-IP and ip proto \udp ) or \
> ( dst host local-IP and ip proto \icmp )'
```

Важно: Обратный слэш (\), присутствующий в приведенных командах, обозначает перевод строки. Например, команды

```
user@host:[~]$ efax -d /dev/cua1 \
> -t T222 file.001 file.002
```

и

```
user@host:[~]$ efax -d /dev/cua1 -t T222 file.001 file.002
```

одинаковы.

Знак «\» служит для экранирования следующего за ним символа. В данном случае для экранирования символа перевода строки. В том случае, если при вводе команды после «\» не перевести строку, будет экранирован следующий после «\» символ и команда будет считана неверно, что вызовет сообщение об ошибке.

2.4. Порядок выполнения лабораторной работы

Важно: Команды, запускаемые для выполнения заданий, и результаты их работы необходимо сохранять для отчета.

1. Запустить `tcpdump` в режиме захвата всех пакетов, проходящих по сети (без фильтра). Количество захватываемых пакетов ограничить семью.

2. Запустить tcpdump в режиме перехвата широковещательного трафика. Фильтровать трафик и по широковещательному аппаратному MAC-адресу (FF:FF:FF:FF:FF:FF), и по широковещательному IP-адресу (можно посмотреть с помощью утилиты ifconfig). Фильтры должны быть связаны логическим объединением. Количество захватываемых пакетов ограничить пятью. Включить распечатку пакета в шестнадцатеричной системе (включая заголовок канального уровня).

3. Запустить tcpdump так, чтобы он перехватывал только пакеты протокола ICMP, отправленные на IP-адрес одного из лабораторных компьютеров. При этом включить распечатку пакета в шестнадцатеричной системе и ASCII-формате (включая заголовок канального уровня). Количество захватываемых пакетов ограничить восемью. Для генерирования пакетов воспользоваться утилитой ping.

4. По образцу рассмотренного в теории примера перехватить трафик утилиты traceroute при определении маршрута до какого-либо узла в сети Интернет. IP-адрес узла можно узнать с помощью сетевой утилиты nslookup:

```
user@host:[~]$ nslookup domain-name
```

5. Используя утилиту tcpdump, отобрать дейтаграммы, принадлежащие соединению TCP между локальным лабораторным ПК и кафедральным сервером 172.16.100.88, и содержащие флаг SYN в заголовке транспортного уровня. Количество дейтаграмм ограничить двумя.

Следует напомнить, что для обработки полей флагов сегмента TCP необходимо использовать выражение tcp[tcpflags] с указанием конкретных значений заданных флагов. Например, для выполнения части данного задания можно воспользоваться конструкцией

```
user@host:[~]$ sudo tcpdump -lvnnSXX 'tcp[tcpflags] & tcp-syn !=0'
```

где опция S указывает утилите tcpdump отображать реальные номера последовательностей сегментов TCP (по умолчанию указываются номера относительно первого перехваченного сегмента), а аргумент регулярного выражения (в скобках) tcp-syn !=0 указывает отбирать только те сегменты TCP, в поле флагов которых бит SYN не равен нулю.

6. Отобрать дейтаграммы, принадлежащие соединениям TCP между локальным лабораторным ПК и сервером 172.16.100.88, и содержащие флаг PSH (tcp-push) в заголовке транспортного уровня. Количество дейтаграмм ограничить тремя.

7. Отобрать дейтаграммы, принадлежащие соединению TCP между локальным лабораторным ПК и сервером 172.16.100.88, и содержащие флаг сброса соединения RST в заголовке транспортного уровня. Количество дейтаграмм ограничить двумя.

8. Отобрать дейтаграммы, принадлежащие соединению TCP между локальным лабораторным ПК и сервером `172.16.100.88`, и содержащие флаги завершения соединения `FIN` или `FIN,ACK` в заголовке транспортного уровня. Количество дейтаграмм ограничить четырьмя.

9. Отобрать дейтаграммы, принадлежащие соединениям TCP между локальным лабораторным ПК и сервером `172.16.100.88`, и содержащие флаги `PSH` или `PSH,ACK` в заголовке транспортного уровня, отправленные с порта TCP службы `http` на диапазон портов назначения 30000–65000. Количество дейтаграмм ограничить десятью.

Следует напомнить, что для обработки дипазонов портов транспортного уровня необходимо использовать выражение `portrange X-Y`, где переменные `X` и `Y` определяют нижнюю и верхнюю границу диапазона портов. Например, для выполнения части данного задания можно воспользоваться конструкцией

```
user@host:[~]$ sudo tcpdump -lvnnSXX 'tcp and portrange 100-150'
```

10. Отобрать дейтаграммы UDP, пересылаемые между локальным лабораторным ПК и сервером `172.16.100.86`, отправленные с номера порта UDP службы DNS (порт 53), на диапазон портов назначения 10000–65535. Количество дейтаграмм ограничить десятью.

11. Отобрать дейтаграммы, принадлежащие соединениям TCP между локальным лабораторным ПК и сервером `172.16.100.88`, установленные между номерами исходящих портов TCP со значением меньше 1024. Количество дейтаграмм ограничить двумя.

12. Отобрать дейтаграммы, пересылаемые между локальным лабораторным ПК и сервером `172.16.100.88`, и использующие номера портов назначения (UDP или TCP) со значениями большими 1024. Количество дейтаграмм ограничить двумя.

13. Отобрать дейтаграммы, UDP пересылаемые между локальным лабораторным ПК и сервером `172.16.100.88`, размер которых больше 50 байт, но не превышает 100 байт. Количество дейтаграмм ограничить десятью.

Следует напомнить, что для отбора дейтаграмм в соответствии с размером необходимо использовать выражение `less X` или `greater X`, отображающее дейтаграммы размером (в байтах) меньше или больше `X`, соответственно.

14. Отобрать дейтаграммы IP, пересылаемые между локальным лабораторным ПК и сервером `172.16.100.88`, принадлежащие соединению TCP, отправленные с порта источника менее 1024 на порт назначения более 10000, размер которых не превышает 100 байт.

15. Отобрать сегменты TCP, пересылаемые между локальным лабораторным ПК и сервером `172.16.100.88`, содержащие в заголовке транспортного уровня флаги `PSH` или `PSH,ACK`, отправленные с номера портов TCP ме-

нее 1024, на диапазон портов назначения 49000–65535, размер которых не превышает 200 байт.

2.5. Содержание отчета

1. Заголовок, согласно приложению.
2. Цель работы.
3. Результаты работы с утилитой `tcpdump`.

Важно: Перед каждым листингом необходимо указать задание и написать соответствующую команду.

2.6. Контрольные вопросы

1. Назначение и принцип работы анализаторов протоколов.
2. Структура заголовка кадра Ethernet.
3. MAC-адрес.
4. Протокол IP.
5. Структура заголовка IP-пакета.
6. IP-адрес.
7. Протоколы транспортного уровня TCP и UDP.
8. Структура заголовка TCP.
9. Структура заголовка UDP.
10. Понятие порта в протоколах транспортного уровня.
11. Виды и назначение флагов в заголовках протоколов транспортного уровня.

Лабораторная работа 3
Изучение протокола ARP. Получение навыков работы с генераторами пакетов

3.1. Цель работы

Рассмотреть принципы работы протокола ARP. Получить практические навыки в анализе кадров ARP и их составлении. Получение базовых навыков по работе с генератором пакетов `packit`.

3.2. Задачи

Просмотреть изменение ARP-таблицы компьютера. Провести разрешение адресов, используя механизм протокола ARP и генератор пакетов `packit`. Провести сетевую атаку вида ARP-спуфинг.

3.3. Теоретический материал

3.3.1. Протокол ARP

ARP (Address Resolution Protocol — протокол определения адреса) — протокол в компьютерных сетях, предназначенный для определения MAC-адреса сетевого устройства по известному IP-адресу.

Наибольшее распространение ARP получил благодаря повсеместности сетей IP, построенных поверх Ethernet, поскольку в подавляющем большинстве случаев при таком сочетании используется ARP. В семействе протоколов IPv6 протокола ARP не существует, его функции возложены на ICMPv6.

Описание протокола было опубликовано в ноябре 1982 г. в RFC 826. ARP был спроектирован для случая передачи IP-пакетов через сегмент Ethernet. При этом общий принцип, предложенный для ARP, был использован и для сетей других типов.

Существуют следующие типы сообщений ARP: запрос ARP (ARP-request) и ответ ARP (ARP-reply). Система-отправитель при помощи запроса ARP запрашивает физический адрес системы-получателя. Ответ (физический адрес узла-получателя) приходит в виде ответа ARP.

Принцип работы протокола: узел (хост *A*), которому нужно выполнить отображение IP-адреса на MAC-адрес, формирует ARP-запрос, вкладывает его в кадр протокола канального уровня, указывая в нем известный IP-адрес (хост *B*), и рассылает запрос широковещательно (в поле MAC-адрес назначения заголовка Ethernet указывается широковещательный MAC-адрес FF:FF:FF:FF:FF:FF). Все узлы локальной сети получают ARP-запрос и сравнивают указанный там IP-адрес с собственным. В случае их совпадения узел (хост *B*) формирует ARP-ответ, в котором указывает свой IP-адрес и свой

локальный адрес и отправляет его уже направленно, так как в ARP запросе отправитель (хост *A*) указывает свой локальный адрес.

Схема работы показана на рис. 3.1.

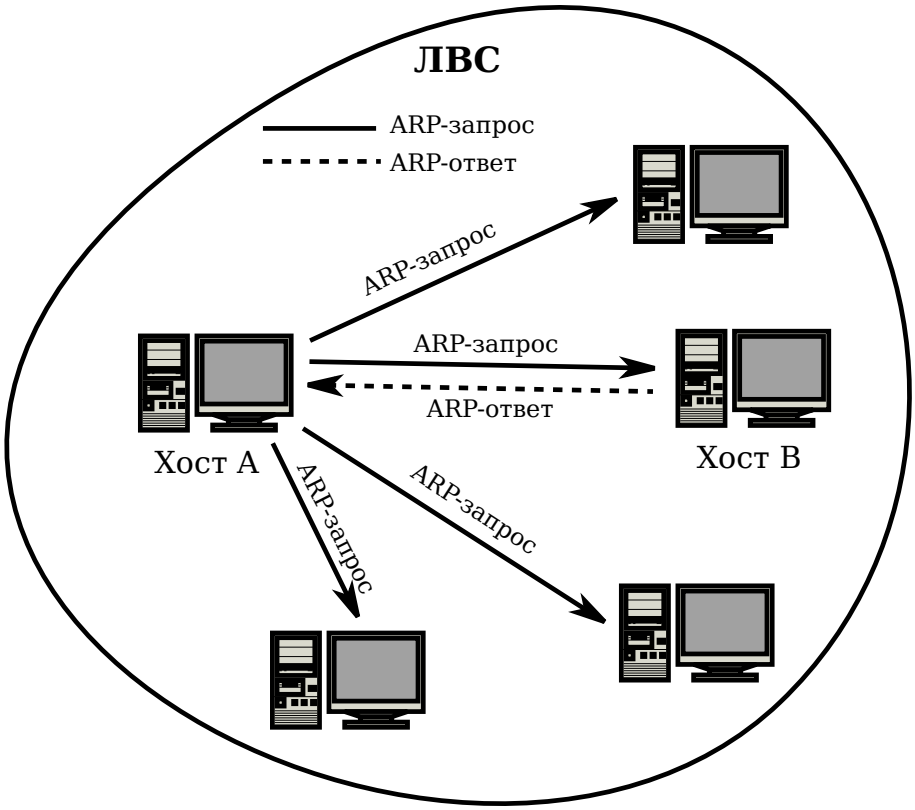

Рис. 3.1. Схема работы протокола ARP

При получении ARP-ответа хост *A* записывает в кэш ARP запись с соответствием IP-адреса хоста *B* и MAC-адреса хоста *B*, полученного из ARP-ответа. Время хранения такой записи ограничено. По истечении времени хранения хост *A* посылает повторный запрос, теперь уже адресно, на известный MAC-адрес хоста *B*. В случае, если ответ не получен, снова посылается широковещательный запрос.

Структура кадра ARP с учетом заголовка Ethernet показана на рис. 3.2.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Destination MAC						Source MAC						ETH TYPE		HTYPE	
PTYPE		HLEN	PLEN	OP CODE		Sender MAC						Sender IP			
Target MAC						Target IP									

Рис. 3.2. Кадр протокола ARP

Значения полей заголовка кадра ARP приведены в табл. 3.1.

Таблица 3.1

Значения полей заголовка кадра ARP

Поле	Значение
HTYPE	Номер протокола передачи канального уровня (0x0001 для протокола Ethernet)
PTYPE	Код протокола сетевого уровня (0x0800 для протокола IPv4)
HLEN	Длина физического адреса в байтах. Адреса Ethernet имеют длину 6 байт
PLEN	Длина логического адреса в байтах. IPv4 адреса имеют длину 4 байта
OP CODE	Код операции: 0x01 в случае ARP-запроса и 0x02 в случае ARP-ответа
Sender MAC	Физический адрес отправителя
Sender IP	Сетевой адрес отправителя
Target MAC	Физический адрес получателя. При запросе поле заполняется нулями
Target IP	Сетевой адрес получателя

Самопроизвольный ARP (gratuitous ARP) — такое поведение ARP, когда ARP-ответ присылается, когда в этом (с точки зрения получателя) нет особой необходимости. Самопроизвольный ARP-ответ это пакет-ответ ARP, присланный без запроса. Он применяется для определения конфликтов IP-адресов в сети: как только станция получает адрес по DHCP или адрес присваивается вручную, рассылается ARP-ответ gratuitous ARP.

Самопроизвольный ARP может быть полезен в следующих случаях:

- обновление ARP-таблиц, в частности, в кластерных системах;
- информирование коммутаторов;
- извещение о включении сетевого интерфейса.

Несмотря на эффективность самопроизвольного ARP, он является особенно небезопасным, поскольку с его помощью можно уверить удаленный узел в том, что MAC-адрес какой-либо системы, находящейся с ней в одной сети, изменился, и указать, какой адрес используется теперь.

3.3.2. Утилита `arp`

Утилита предназначена для работы с кэшем ARP (ARP таблицей). Основными ее возможностями являются просмотр кэша, удаление записи и добавление записи вручную. Для операций удаления и добавления записей требуются права суперпользователя. Просмотр кэша возможен от лица обычного непривилегированного пользователя.

От лица обычного пользователя утилита `arp` запускается следующим образом:

```
user@host:[~]$ /usr/sbin/arp
```

При таком запуске утилиты `arp` без параметров на экран будет выведено содержимое кэша ARP.

Более подробную информацию об использовании утилиты можно узнать из официальной man-страницы. Команда:

```
user@host[~]$ man arp
```

3.3.3. *Генератор пакетов* `packit`

Утилита `packit` объединяет в себе возможности генератора пакетов и анализатора трафика. Она позволяет перехватывать и отправлять в сеть кадры Ethernet, IP- и TCP-пакеты.

В рамках лабораторной работы будет использоваться только режим генератора пакетов.

Утилита `packit` запускается в терминале от имени суперпользователя. Формат запуска команды:

```
user@host:[~]$ sudo packit [options]
```

Поскольку в лабораторной предполагается работа только с протоколом ARP, рассмотрим соответствующую команду запуска:

```
user@host:[~]$ sudo packit -t arp -A packet-type -y target-IP \
> -Y target-MAC -x sender-IP -X sender-MAC \
> -e source-MAC -E destination-MAC -i interface
```

Параметр *`packet-type`* для ARP-запроса должен быть равен 1, а для ARP-ответа — 2.

Параметр *`interface`* определяет сетевой интерфейс, который необходимо использовать. Как правило, в ОС Debian Linux это `eth0` для проводного интерфейса и `wlan0` для беспроводного. Точное имя сетевого интерфейса можно посмотреть командой `ifconfig`.

Параметры `source-MAC` и `destination-MAC`, задаваемые флагами `-e` и `-E` соответственно, относятся к заголовку кадра Ethernet.

IP-адреса указываются в десятичном виде.

Более подробную информацию об использовании утилиты можно узнать из официальной man-страницы. Команда:

```
user@host:[~]$ man packit
```

3.3.4. *Сетевая атака ARP-спуфинг*

Сетевая атака ARP-спуфинг (ARP-spoofing) основана на использовании самопроизвольного ARP.

Чтобы перехватить сетевые пакеты, которые атакуемый хост (*A*) отправляет на хост *B*, атакующий хост (*C*) формирует ARP-ответ, в котором

ставит в соответствие IP-адресу хоста *B* свой MAC-адрес. Далее этот пакет отправляется на хост *A*. В том случае, если хост *A* поддерживает самопроизвольный ARP, он модифицирует собственную ARP-таблицу и помещает туда запись, где вместо настоящего MAC-адреса хоста *B* стоит MAC-адрес атакующего хоста *C*.

Теперь пакеты, отправляемые хостом *A* на хост *B*, будут передаваться хосту *C*.

Схема атаки показана на рис. 3.3.

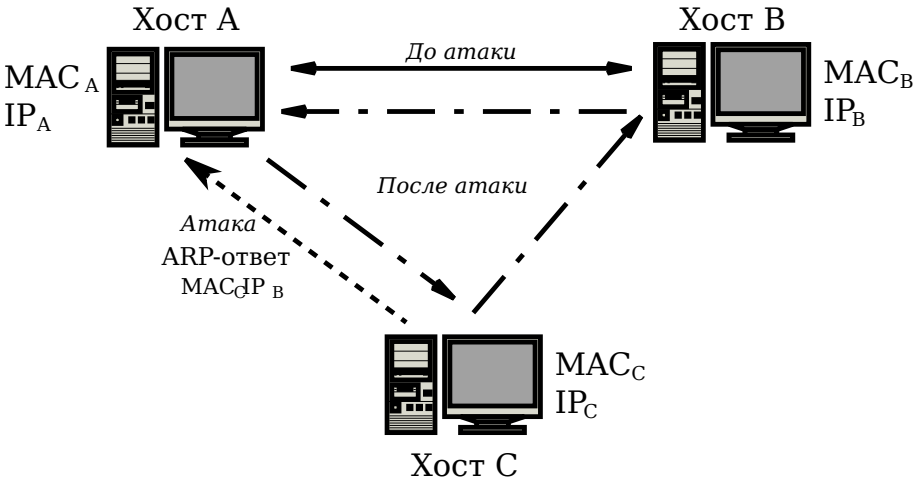

Рис. 3.3. Схема сетевой атаки ARP-спуфинг

3.4. Порядок выполнения лабораторной работы

3.4.1. Утилита `arp`. ARP-кэш

1. Просмотреть ARP-таблицу лабораторного компьютера, воспользовавшись утилитой `arp`. Выведенную на экран информацию сохранить для отчета.
2. При помощи утилиты `ping` проверить доступность одного из лабораторных ПК, отсутствующих в кэше ARP.
3. Повторно просмотреть ARP-таблицу лабораторного компьютера, воспользовавшись утилитой `arp`. Отметить разницу. Выведенную на экран информацию сохранить для отчета.

3.4.2. Отправка ARP-запроса и получение ARP-ответа

1. Подготовить и записать в 16-ричном виде пример кадра для широковещательной передачи ARP-запроса хостом *A* и кадра ARP-ответа хостом *B* хосту *A*. В кадре ARP-ответа поля для MAC-адреса хоста *B* не заполнять.

Хост *A* — это ПК, за которым работает бригада студентов. В качестве хоста *B* используется ПК другой бригады студентов. IP-адреса хостов *A* и *B* можно узнать с помощью команды `ifconfig`. Также, IP-адрес может быть написан на самом компьютере или выведен на рабочий стол.

2. Начать захват пакетов при помощи анализатора протоколов `tcpdump`.

```
user@host:[~]$ sudo tcpdump -i interface -lvnnSXX -s 1500 \
> 'ether proto \arp and host IP-address-A \
> and host IP-address-B'
```

3. Сформировать кадр ARP-запроса с помощью утилиты `packit` и отправить его в сеть. Команду и выведенную на экран информацию сохранить для отчета.

4. Убедиться, что был получен кадр ARP-ответа, соответствующий посланному запросу. Захваченные кадры ARP-запроса и ARP-ответа сохранить для отчета.

5. Прекратить захват пакетов.

3.4.3. ARP-спуфинг: подмена соответствия MAC-адреса IP-адресу

В этой части работы необходимо исполнить роль атакующего хоста (хост *C* на рис. 3.3) и перехватить трафик, направленный от атакуемого хоста *A* (ПЭВМ другой бригады студентов) на хост *B* (сервер кафедры или любая другая работающая ПЭВМ). Для генерирования трафика необходимо воспользоваться утилитой `ping`.

1. Подготовить и записать в 16-ричном виде кадр ARP-ответа, направляемый атакующим хостом *C* на атакуемый хост *B*. Кадр должен быть составлен так, чтобы IP-адресу хоста *B* соответствовал MAC-адрес хоста *C*.

2. Начать захват пакетов при помощи анализатора протоколов `tcpdump`:

```
user@host:[~]$ sudo tcpdump -i interface -XX -s 1500 \
> 'ip proto \icmp and dst host IP-address-B'
```

3. Сформировать кадр ARP-ответа с помощью `packit` и отправить его хосту *A*. Кадр необходимо отправлять хосту *A* раз в несколько секунд. Команду генерирования пакетов и выведенную на экран информацию сохранить для отчета.

Формат команды `packit`:

```
user@host:[~]$ sudo packit -t arp -A packet-type -y target-IP \
> -Y target-MAC -x sender-IP -X sender-MAC \
> -e source-MAC -E destination-MAC \
> -c packet-count -w time-interval -i interface
```

4. Проверить доступность хоста *B* с хоста *A* при помощи утилиты `ping`. Убедиться, что эхо-запросы от хоста *A*, направленные на хост *B*, поступают хосту *C*.

5. Прекратить захват пакетов.

6. Сохранить для отчета отправленный кадр ARP-ответа и несколько перехваченных эхо-запросов.

3.5. Содержание отчета

1. Заголовок согласно приложению.
2. Цель работы.
3. Листинги ARP-таблицы согласно п. 3.4.1
4. Результаты работы по п. 3.4.2:

а) кадр ARP-запроса и кадр ARP-ответа в 16-ричном виде;

б) команда `packit` для отправки в сеть сформированного ARP-запроса и информацию, выведенную на экран после ее выполнения;

в) захваченные утилитой `tcpdump` кадры ARP-запроса и ARP-ответа.

5. Результаты работы по п. 3.4.3:

а) схема проведения сетевой атаки по аналогии с рис. 3.3 с указанием MAC- и IP-адресов лабораторных компьютеров;

б) кадр ARP-ответа с подменой аппаратного адреса в 16-ричном виде;

в) команда `packit` для отправки в сеть сформированного ARP-ответа и информация, выведенная на экран после ее выполнения;

г) захваченные утилитой `tcpdump` эхо-запросы.

3.6. Контрольные вопросы

1. Протокол ARP.
2. Формат пакета ARP.
3. Самопроизвольный ARP.
4. IP-адрес.
5. MAC-адрес.
6. ARP-спуфинг.

Лабораторная работа 4
Технологии управления и протоколы обработки удаленного доступа

4.1. Цель работы

Лабораторная работа предназначена для целей практического ознакомления с программным обеспечением удаленного доступа к системам обработки данных. Рассматривается создание шифрованных транспортных туннелей для обеспечения средств управления и передачи файлов между центром управления и удаленной системой обработки данных.

4.2. Задачи

Создать подключение удаленного доступа к системе обработки данных, сформировать шифрованные ключи и произвести их обмен с удаленной системой, передать файл по шифрованному туннелю, воспользовавшись беспарольным доступом с аутентфикацией по публичным ключам.

4.3. Теоретический материал

Программное обеспечение систем обработки удаленного доступа строится на основе ахитектуры клиент-сервер. После успешной авторизации клиентской части ПО пользователя, серверный процесс передает управление указанной командной оболочке операционной системы. Окружение пользователя в случае доступа к удаленной системе, практически полностью аналогично работе в окружении локальной системы.

Наиболее распространенными реализациями программного обеспечения обработки удаленного доступа являются службы `telnet` и `ssh`. В работе каждой из указанных служб существует ряд определенных особенностей.

Протокол прикладного уровня эталонной сетевой модели OSI TELNET реализует сетевой текстовый (псевдографический) интерфейс между удаленными хостами. В качестве конечных точек взаимодействия могут выступать виртуальные терминалы, системные процессы и проч. Серверный процесс службы `telnet` по-умолчанию ожидает соединений на сокете TCP, порт 23. Проведя регистрацию пользователя (логин – пароль), служба предоставляет доступ к интерфейсу командной строки. Следует отметить, что TELNET не имеет встроенной поддержки шифрования и является критически уязвимым к проведению любых видов сетевых атак. Применение и исследование службы `telnet` оправдано с академической и методологической точки зрения. Кроме указанных причин, клиент `telnet`, также применяется для исследования доступности некоторых текстовых протоколов и сетевых служб (SMTP, POPx, FTP и т. д.) на удаленных хостах.

Реализация протокола TELNET, в рамках лабораторной работы, осуществляется серверной службой `telnetd` и клиентом `telnet`.

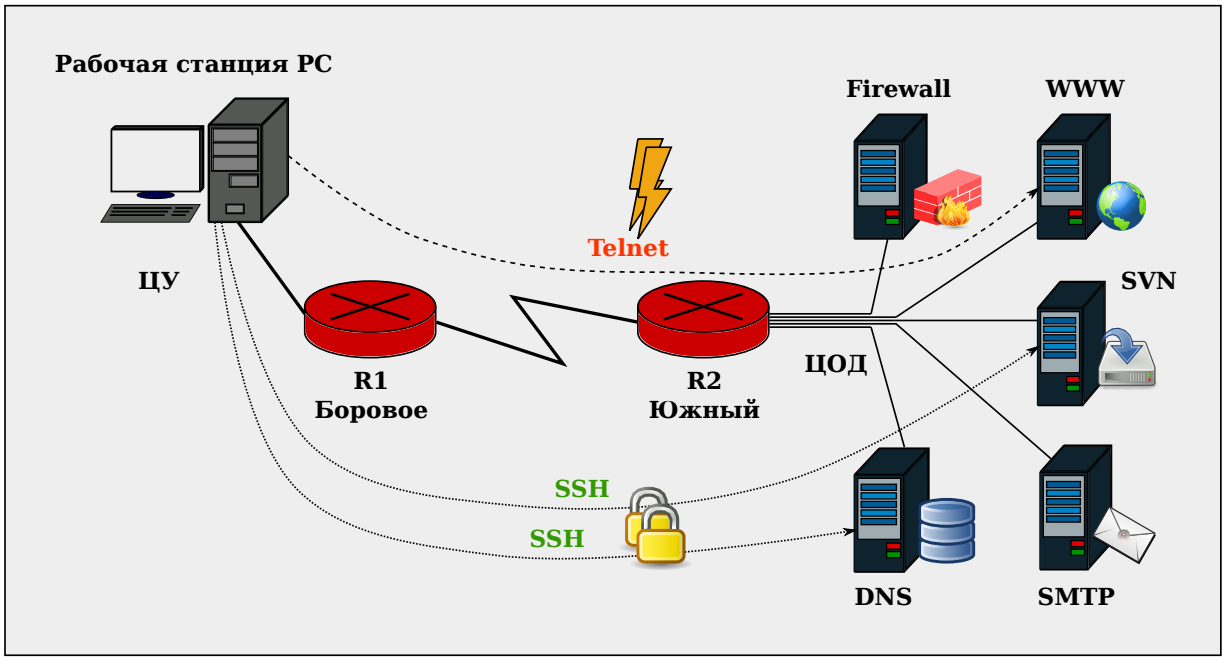

Рис. 4.1. Удаленный доступ в фрагменте сети

В отличие от службы `telnet`, протокол SSH предоставляет шифрованное транспортное соединение к удаленной системе обработки данных. Передаваемая информация подвергается симметричному блочному шифрованию с использованием одного из стойких алгоритмов шифрования: Blowfish или Triple DES (3DES). Особенностью работы протокола SSH является создание частного и публичного зашифрованных ключей. В процессе установления соединения и проверки подключения, удаленные узлы обмениваются публичными ключами, подтверждая таким образом свою подлинность. Серверный процесс ожидает подключений на сокете TCP, порт 22. После процедур обмена шифрованными ключами и регистрации пользователя, сервер передает управление интерфейсу командной оболочки. Любая информация передаваемая по протоколу SSH инкапсулируется в транспортные сегменты, подвергается шифрованию и с помощью процедур сетевого уровня передается на удаленную систему.

В данной лабораторной работе рассматривается реализация протокола SSH средствами пакета OpenSSH. Помимо предоставления удаленного доступа к интерфейсу командной строки и некоторых других служб, в пакет OpenSSH также входят утилиты передачи файлов между системами обработки данных внутри шифрованного транспортного канала.

4.3.1. Протокол TELNET

Для подключения к удаленной системе с использованием протокола TELNET достаточно ввести команду `telnet` в оболочке интерпретатора командной строки. Управление командной оболочкой будет передано клиенту `telnet` локальной системы. Для управления подключением к удаленным системам существует набор команд, краткий список которых представлен в табл. 4.1.

Таблица 4.1

Краткий список команд внутренней оболочки клиента `telnet`

Команда	Назначение
`close`	Разорвать соединение с удаленным узлом
`logout`	Отключиться от оболочки командного интерпретатора удаленной системы
`open` *hostname*	Установить соединение с узлом *hostname*
`status`	Вывести текущий статус клиента telnet
`?`	Вывести список и справку по всем доступным командам

Пример использования внутренних командных конструкций `telnet` приведен в лист. 4.1.

Листинг 4.1

Использование клиента `telnet`

```
user@host:[~]$ telnet
telnet> ?
Commands may be abbreviated. Commands are:
...
close - close current connection
logout - logout remote user, close the connection
display - display operating parameters
mode - enter line or character mode
...
telnet> open
(to) lsrv.opdsnet 23
Trying 172.16.100.88...
Connected to lsrv.opdsnet.
Escape character is '^]'.

FreeBSD/i386 (lsrv.opdsnet) (pts/1)

login: ...
...
```

Помимо режима упрощенной командной оболочки (лист. 4.1), клиент `telnet` поддерживает краткую форму с помощью аргументов командной строки. В простейшем случае командная конструкция трансформируется к виду `telnet` *`удаленный_узел порт`*, где в качестве аргумента *`удаленный_узел`* следует указать IP-адрес удаленной системы или ее доменное имя. Аргумент *`порт`* можно опустить, если серверная служба использует порт по умолчанию (TCP 23).

Листинг 4.2

Подключение к удаленному узлу с помощью клиента `telnet`

```
user@host:[~]$ telnet mirror.opdsnet 23
Trying 172.16.100.18...
Connected to mirror.opdsnet.
Escape character is '^]'.

FreeBSD/i386 (mirror.opdsnet) (pts/3)

login: ...
...
```

4.3.2. Протокол SSH

Существует несколько типов подключения к удаленной системе с использованием службы `ssh`. В общем случае, для создания сетевого подключения к удаленной системе обработки данных предназначена команда `ssh`, в качестве одного из аргументов которой служит IP-адрес или доменное имя удаленной системы (лист. 4.3).

Листинг 4.3

Подключение к удаленному узлу

```
user@host:[~]$ ssh 172.16.100.19
```

В примере из лист. 4.3 происходит попытка установления соединения с узлом 172.16.100.19, где в качестве имени пользователя будет использоваться логин `user`. В случае успешной попытки подключения к удаленному узлу, необходимо указать пароль. Опция `-l`, команды `ssh`, используется в том случае, если необходимо произвести подключение к удаленной системе под другим логином и паролем (лист. 4.4). С кратким набором доступных опций команды можно ознакомиться в табл. 4.2.

Листинг 4.4

Альтернативный логин

```
user@host:[~]$ ssh -l admin 172.16.100.19
```

Приведенные примеры описывают аутентификацию с *публичным ключом узла*. Это означает, что соединение устанавливается на основании публичного ключа удаленного узла и регистрационной информации пользователя. Однако для создания более надежного подключения применяется метод аутентификации с *частным и публичным ключом*. В этом случае необходимо сгенерировать собственную пару из публичного и частного ключа, а затем распространить публичный ключ на удаленные узлы. Таким образом становится возможным проводить защищенную аутентификацию без использования регистрационного пароля пользователя системы удаленной обработки данных.

Таблица 4.2

Краткий список опций команды `ssh`

Опция	Назначение
`-1`	Использовать протокол SSH версии 1
`-2`	Использовать протокол SSH версии 2
`-4`	Использовать только адреса IPv4
`-6`	Использовать только адреса IPv6
`-c blowfish|3des|des`	Указать алгоритм для шифрования сеанса
`-C`	Использовать сжатие данных
`-l пользователь`	Войти на удаленную систему под учетной записью заданного пользователя
`-p`	Задать порт для подключения на удаленной системе
`-q`	Не выводить подробную информацию о подключении
`-v`	Выводить подробную информацию о подключении
`-V`	Вывести версию ПО

Для выполнения данных процедур необходимо использовать команду `ssh-keygen`, которая сформирует пару ключей в корневом каталоге пользователя — `~/.ssh/id_rsa` и `~/.ssh/id_rsa.pub`. Данные, хранящиеся в файле `id_rsa.pub`, следует передать на удаленный узел и записать (дописать) в файл `~/.ssh/authorized_keys`. Пример использования команд приведен в лист. 4.5.

Листинг 4.5

Создание пары ключей

```
user@host:[~]$ ssh_keygen
Generating public/private rsa key pair.
...
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
```

Далее необходимо передать публичный ключ локальной системы на удаленный узел. С помощью команды `scp` можно произвести передачу файлов по шифрованному транспортному соединению `ssh`. Команда `scp` поддерживает

множество способов передачи данных, однако в данном случае, в качестве аргументов достаточно указать систему отправки и удаленный узел приемки файла. Пример приведен в лист. 4.6. Краткий список опций команды `scp` приведен в табл. 4.3.

Листинг 4.6

Передача файлов по шифрованному каналу

```
user@host:[~]$ scp ~/.ssh/id_rsa.pub \
> 172.16.100.86:~/.ssh/authorized_keys
```

Таблица 4.3

Краткий список опций команды `scp`

Опция	Назначение
`-4`	Использовать только адреса IPv4
`-6`	Использовать только адреса IPv6
`-B`	Пакетный режим передачи
`-C`	Использовать сжатие данных
`-P`	Подключаться к указанному порту на удаленной системе
`-q`	Не показывать ход выполнения передачи данных
`-r`	Выполнять рекурсивное копирование каталогов
`-v`	Режим подробного вывода

В лист. 4.6 в качестве первого аргумента команды `scp` выступает файл `id_rsa.pub`, который необходимо передать узлу получателю с IP-адресом 172.16.100.86, в каталог `.ssh` и сохранить под именем `authorized_keys`.

По завершению указанных процедур, становится возможным подключиться к удаленной системе, произведя авторизацию по отпечатку публичного ключа (лист. 4.7).

Листинг 4.7

Беспарольное подключение по отпечатку публичного ключа

```
user@host:[~]$ ssh 172.16.100.86
...
user@remotehost[~]:
```

4.4. Порядок выполнения лабораторной работы

1. Открыть эмулятор терминала ОС и запустить анализатор сетевого трафика `tcpdump` с фильтром пакетов, получаемых и передаваемых от узла `172.16.100.88` с TCP-портом источника или назначения 23. С помощью команды `tee` вывести отфильтрованные IP-пакеты на экран эмулятора терминала и сохранить данные в файл `telnet.log` в домашнем каталоге пользователя.

Для этого следует воспользоваться командой

```
user@host:[~]$ sudo tcpdump -lvnXX host 172.16.100.88 and \
> host IP_NN and tcp and (src port 23 or dst port 23 ) | \
> tee telnet.log
```

где переменная IP_NN обозначает IP-адрес локального интерфеса ПК. Конвеерная передача данных утилите tee позволит сохранить кадры в указанный файл и одновременно вывести их на экран терминала.

2. Открыть второй эмулятор терминала и попытаться установить соединение с удаленным сервером по адресу 172.16.100.88 по сетевому протоколу TELNET. Для авторизации на сервере следует использовать логин вида student_nn, где переменная nn обозначает номер ПК (например, student_01, student_11 и т. д.).

3. Воспользовавшись окном сетевого монитора tcpdump, анализировать прохождение сетевых пакетов между узлами назначения. Отметить пакеты инициации соединения telnet.

4. Подключившись к удаленной системе, ввести пароль stud и выполнить команду uname -a, выведя тем самым информацию об удаленной системе. Для разрыва соединения использовать команду logout.

5. В окне сетевого монитора отметить пакеты, инициирующие разрыв сессии telnet. Прервать фильтрацию пакетов сетевым анализатором tcpdump, воспользовавшись комбинацией Ctrl-C. В файле telnet.log выделить записи установления и разрыва соединения с сервером telnet.

6. Снова запустить анализатор сетевого трафика с фильтром пакетов, получаемых и передаваемых от узла 172.16.100.88 с TCP-портом источника и назначения 22. С помощью команды tee вывести отфильтрованные IP-пакеты на экран эмулятора терминала и сохранить данные в файл ssh.log, в домашнем каталоге пользователя. Для выполнения этого задания следует воспользоваться командой

```
user@host:[~]$ sudo tcpdump -lvnXX host 172.16.100.88 and \
> host IP_NN and tcp and (src port 22 or dst port 22 ) | \
> tee ssh.log
```

где переменная IP_NN обозначает IP-адрес локального интерфеса ПК.

7. Открыть второй эмулятор терминала и с помощью команды

```
user@host:[~]$ ssh -l student_nn 172.16.100.88
```

попытаться установить шифрованное соединение с удаленным сервером. Переменная nn, как и прежде, обозначает номер лабораторного ПК. Проследить передачу и прием пакетов между узлами в окне сетевого анализатора. Отметить взаимодействующие TCP-порты.

8. Подключившись к удаленной системе, ввести пароль stud и выполнить команду uname -a, выведя информацию об удаленной системе

9. В домашнем каталоге пользователя создать текстовый файл с содержанием ФИО и номера лабораторного ПК. С помощью команды

```
scp -v -o User=student_nn /home/student/имя_файла \
> 172.16.100.88:/home/student_nn/
```

передать его по шифрованному каналу на удаленную систему. Проверить наличие копии переданного файла на удаленном узле, воспользовавшись файловым менеджером Midnight Commander (команда mc на удаленной системе).

10. Отключившись от удаленного узла (команда logout), на локальном хосте сформировать зашифрованные ключи, воспользовавшись командой ssh-keygen.

11. Используя команду scp с указанием места расположения файла публичного ключа на локальной системе (публичный ключ хранится в файле /home/student/.ssh/id_rsa.pub), произвести его передачу по шифрованному туннелю на удаленный узел в каталог /home/student_nn/.ssh/ и сохранить файл публичного ключа под именем authorized_keys. Проследить процесс пересылки пакетов между удаленными узлами в окне анализатора пакетов tcpdump.

12. Воспользовавшись командой ssh -l student_nn 172.16.100.88, снова сделать попытку подключения к удаленной системе. Отметить отличия в процедурах подключения и регистрации пользователя на удаленной системе по сравнению с аутентификацией по паролю.

13. Аналогично, с помощью команды scp произвести повторную передачу текстового файла на удаленный узел. Убедиться в наличии переданной копии файла на удаленном хосте. Отметить отличия в процедуре передачи файла.

14. Остановить анализатор сетевых пакетов, воспользовавшись комбинацией Ctrl-C. Просмотреть содержимое файла ssh.log, отметить пакеты инициации сетевого взаимодействия и разрыва соединений TCP.

4.5. Содержание отчета

1. Заголовок согласно приложению.
2. Цель работы.
3. Задание.
4. Экранные копии и листинг работы с командной оболочкой эмулятора терминала.
5. Журналы (telnet.log и ssh.log) передачи данных собранные утилитой tcpdump.

4.6. Контрольные вопросы

1. Определите основные цели и задачи, решаемые с помощью ПО удаленного доступа.

2. Выделите отличительные особенности между режимами работы удаленного доступа по протоколам TELNET и SSH.

3. Опишите способы установления соединения при использовании протокола SSH. Охарактеризуйте положительные и отрицательные аспекты приведенных методов.

4. Основываясь на заданиях лабораторной работы, приведите практический пример использования систем удаленного доступа.

5. Перечислите распространенные сетевые службы, основанные на использовании шифрованного соединения по протоколу SSH. Приведите пример использования службы передачи файлов по безопасному туннелю.

Лабораторная работа 5
Протоколы и службы передачи файлов

5.1. Цель работы

Лабораторная работа ставит цели закрепления теоретического материала по протоколам и программному обеспечению службы передачи файлов. В рамках заданий данной лабораторной работы рассматриваются протоколы FTP, TFTP, SFTP и пакеты прикладного ПО для работы с ними.

5.2. Задачи

Ознакомившись с принципами работы основных протоколов передачи файлов, с помощью клиентов служб `ftp`, `tftp` и `sftp` произвести выгрузку заданных файлов с удаленной системы хранения данных и выполнить загрузку указанных файлов на удаленный узел.

5.3. Теоретический материал

Прикладное программное обеспечение, предназначенное для работы с телематическими службами передачи файлов между системами обработки данных, занимает значительное функциональное положение в инфраструктуре современных компьютерных сетей.

Протоколы, реализующие процедуры и функции службы передачи файлов между узлами сети, ведут свою историю со времени зарождения первичных компьютерных сетей (1971 г.). В отсутствие на тот момент функций, предоставляемых сервисами электронной почты (SMTP, POP, IMAP) и пересылки гипертекстовой информации (HTTP), служба передачи файлов (FTP) использовалась как основное средство обмена данными между связанными мейнфреймами и удаленными терминалами обработки информации.

Несмотря на текущее, постепенное внедрение и распространенние протоколов, поддерживающих широкие возможности электронного документооборота (например, протоколы WebDAV, SMB, интегрированные сервисы Google Docs и т. д.), службы и протоколы передачи файлов остаются важным приложением в информационном пространстве глобальных и локальных сетей, выполняющим задачи от загрузки файлов на Web-хостинг до распространения программного обеспечения через сетевые репозитории.

Существует несколько вариантов реализаций протокола передачи файлов (FTP), каждая из которых предназначена для наиболее эффективного выполнения задач сетевого обмена информацией. В рамках лабораторной работы будут рассматриваться протоколы TFTP (Trivial File Transfer Protocol), SFTP (SSH File Transfer Protocol) и стандартный протокол IETF FTP. Следует отметить, что протоколы TFTP и FTP имеют официальный статус (стандарт

Internet) и подтверждены спецификациями IETF с номерами RFC 1350 и RFC 959, соответственно. Протокол SFTP разрабатывался в рамках рассмотренного ранее прикладного протокола SSH (IETF RFC 4251), и до настоящего времени является черновиком стандарта (Draft), однако, де-факто, обладает функциональным рядом реализаций (в том числе принятых промышленностью отрасли) и широким распространением среди стандартных служб сети Internet.

Простейший протокол передачи файлов, в качестве сетевого транспорта использует UDP-дейтаграммы. Серверный процесс ожидает получения дейтаграмм на сокете UDP, порт 69. Протоколы FTP и SFTP манипулируют TCP-сокетами. SFTP-служба использует туннелирование канала в транспортный TCP поток SSH, т. е. работа сервера SFTP осуществляется поверх работы службы SSH. По-умолчанию для пересылки файлов используется тот же TCP-порт, что и для службы SSH (TCP 22).

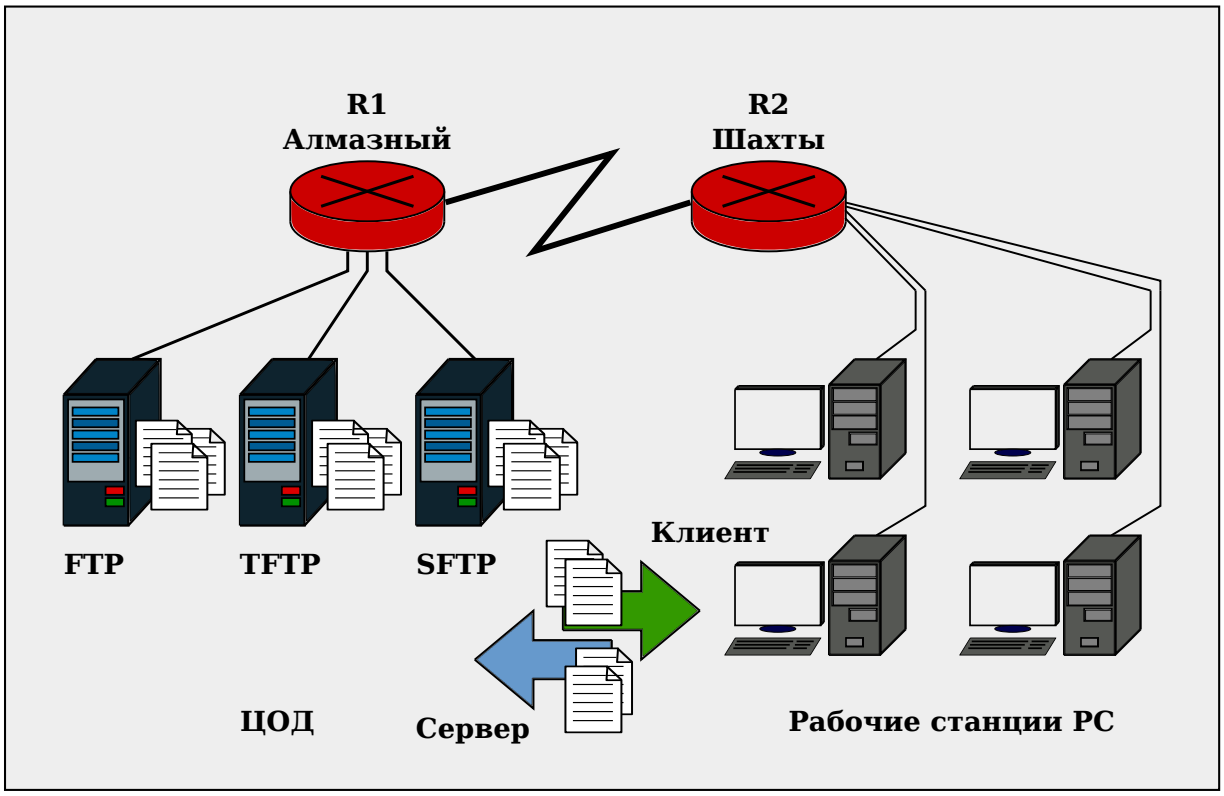

Рис. 5.1. Пересылка файлов в структуре распределенной сети

В отличие от расмотренной ранее утилиты `scp` из пакета OpenSSH, предназначенной для простого, односеансового, защищенного копирования файлов, служба SFTP реализует не только полный набор возможностей стандартного протокола FTP, но и добавляет шифрование транспортного туннеля, а также возможность аутентификации клиентов по уникальным пользовательским ключам.

В независимости от конкретной реализации, протоколы FTP имеют клиент-серверную структуру. Однако базовый протокол FTP отличается достаточно динамичной схемой взаимодействия между клиентским и серверным ПО. По умолчанию используется два соединения между удаленными узлами. Управляющее соединение, инициируемое клиентом, характеризуется пересылкой TCP-сегментов между динамическим сокетом клиента и 21 портом сетевой службы сервера. Данный канал предназначен только для организации текстовой оболочки управления и пересылки командных конструкций (например, загрузка файла, выгрузка, просмотр директорий и т. д.). Направление непосредственной пересылки полезных данных (файлов) зависит от режима работ сервера и клиента FTP.

В первом случае, клиентское ПО, используя управляющее соединение, отсылает серверу номер динамического порта, выделенного ОС для установления внешнего TCP-соединения. После получения соответствующих сегментов TCP-потока, серверная служба FTP попытается установить соединение, используя локальный TCP-порт 20 и указанный динамический порт клиента, т. е. организовать исходящее соединение от сервера к клиенту. После передачи файлов (любых иных данных) данное соединение будет закрыто, а повторная передача потребует выделения клиентом нового динамического порта, для ожидания входящего подключения с стандартного TCP-порта 20 сервера. Такой режим передачи файлов получил название «активный». На рис. 5.2 изображена условная диаграмма управляющего соединения и пересылки файлов.

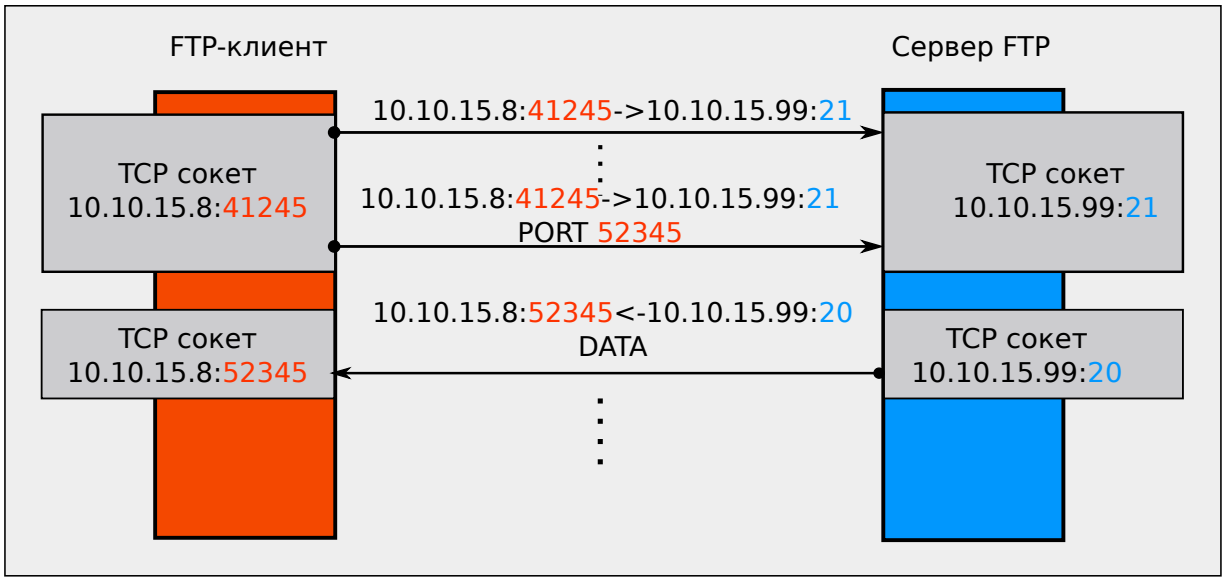

Рис. 5.2. Активный режим передачи файлов

Основным и наиболее очевидным недостатком рассмотренного режима является проблема создания соединения с клиентским ПО, расположенным за сетевыми экранами (firewall, netfilter). Политика сетевого экрана, по сложив-

шейся общественной практике, заключается в разрешении всех легитимных исходящих соединений (от клиентов внутренней сети) и запрете любого входящего подключения (извне внутренней сети), не принадлежащего текущим (открытым) исходящим TCP-потокам. Таким образом, все попытки удаленного FTP-сервера установить TCP-подключение к внутреннему клиенту будут завершаться неудачно. Следует подчеркнуть, что указанное ограничение, в подавляющем большинстве случаев, распространяется только на клиентов корпоративных сетей, расположенных, как правило, за сетевыми экранами организации и использующих различные варианты трансляции сетевых адресов (NAT). Кроме этого, следует отметить, что некоторые распространенные реализации серверов служб FTP, функционируя в активном режиме, не используют порт TCP 20, тем самым частично нарушая алгоритм указанный в RFC 959. В таком случае, для создания исходящего соединения сервером FTP используется один из свободных портов TCP динамического диапазона ОС, что дополнительно осложняет взаимодействие удаленных систем через межсетевой экран, поскольку возможность подбора номера порта TCP, с которого будет осуществляться входящее соединение, представляется маловероятной. В любом другом сценарии, когда не существует явного запрета на создание входящих TCP-соединений, допустимо использование активного режима передачи файлов по протоколу FTP.

Для преодоления существующего ограничения в работе с клиентами, расположенными за сетевыми экранами, применяется обратный метод установления транспортного соединения («пассивный» режим). По аналогии с рассмотренным алгоритмом, используется два соединения TCP. Подключение, управляющее сессией FTP, организуется между динамическим портом клиента и 21 TCP-портом FTP-службы сервера. При запросе передачи файлов клиентом, сервер отправляет сегмент с номером динамического порта (по-умолчанию, серверная служба FTP использует диапазон TCP-портов 49152–65534), на котором будет ожидаться входящее подключение, по направлению от клиента к серверу. ОС клиента выделяет еще один динамический TCP-порт и инициирует исходящее подключение к указанному динамическому TCP-порту сервера FTP. На рис. 5.3 изображена диаграмма работы в пассивном режиме.

Помимо представленного выше алгоритма установления соединений, серверная служба стандартного протокола FTP поддерживает две схемы аутентификации пользователей.

В общем случае, аутентификация пользователя необходима для выделения ряда полномочий учетной записи. Использование набора учетных записей позволяет контролировать и разделять доступные ресурсы сервера между всеми пользователями данной службы. Под ресурсами следует понимать не

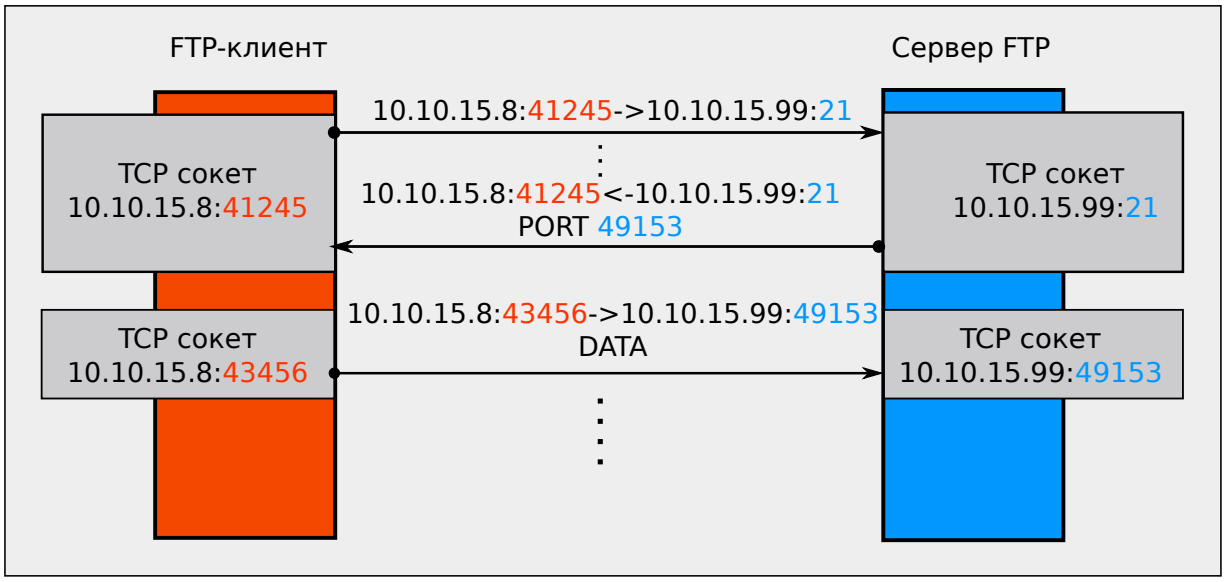

Рис. 5.3. Пассивный режим передачи файлов

только материально-технические возможности аппаратуры, но и, например, наличие разрешений (прав) на запись в каталог, изменение структуры директорий, модификации файлов и т. д. Таким образом, в зависимости от того, какую учетную запись использует пользователь при регистрации в системе серверной службы FTP, клиенту будет выделен определенный набор прав на использование ресурсов сервера.

Наиболее распространенный способ доступа к ресурсам сервера FTP в глобальных сетях — использование анонимного доступа. В данном случае любой пользователь, подключившийся к удаленной службе FTP, автоматически получает ограниченные права, соответствующие ограниченной системной записи `ftp`. По распространенной сетевой практике служба FTP разрешает чтение общедоступных файлов, а в редких случаях допускается запись данных в определьные каталоги. Подобный режим работы сервера FTP принято называть «анонимным», так как для работы с файловым содержимым используется абстрактная обобщенная учетная запись с максимально ограниченными возможностями.

Для авторизации в системе анонимного сервера FTP следует использовать логин «anonymous» и пароль в виде электронной почты, например, `user@mail.com`. Необходимо подчеркнуть, что система авторизации не требует указания реально существующего электронного почтового ящика, и в качестве пароля можно использовать произвольную строку символов.

Второй способ аутентификации подразумевает использование индивидуальных учетных записей пользователей. Разрешения на используемые ресурсы сервера в данном случае полностью соответствуют правам регулярной учетной записи в ОС сервера FTP. Авторизация пользователя требует указания конкретного логина и соответствующего ему пароля. Подобная схе-

ма аутентификации пользователей получила широкое распространение в среде корпоративных и коммерческих сетей (например, службы web-хостинга и проч.). Серверную службу, работающую в таком режиме, принято именовать «приватной» (частной).

5.3.1. Протокол TFTP

Как уже было упомянуто ранее, протокол TFTP использует транспортные средства дейтаграмм UDP, а серверная служба ожидает входящие пакеты на порт UDP 69. После получения UDP-дейтаграммы на указанный порт серверная служба выделяет произвольный динамический порт (UDP) и отвечает на порт клиента, с которого был принят запрос. Дальнейшее взаимодействие осуществляется только с использованием динамических портов, а порт 69 необходим для первоначального инициирования службы (рис. 5.4).

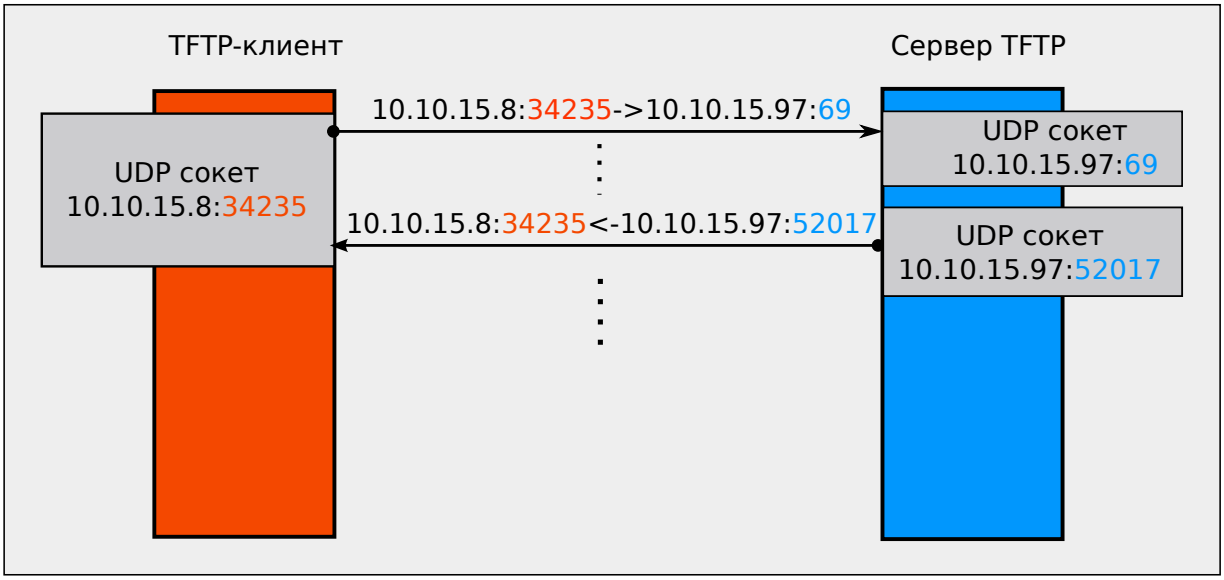

Рис. 5.4. Взаимодействие клиента и сервера TFTP

Вследствие очень маленькой и простой реализации, служба TFTP получила широкое распространение во встраиваемых устройствах и «системах на чипе» (SoC) — начиная от бездисковых сетевых рабочих станций и заканчивая домашними IP-маршрутизаторами. В отличие от рассматриваемых далее протоколов, TFTP не поддерживает средств аутентификации и не использует процедуры авторизации пользователя. Единственный способ ограничить доступ к серверной службе — указать допустимые диапазоны IP-адресов клиентов.

Служба TFTP вносит ограничения на запись файлов и позволяет модифицировать только файлы явно доступные для записи. Наиболее распространенное применение данной службы — загрузка небольших файлов с сервера на клиент. Стандартный протокол TFTP поддерживает файлы размером до

32 Мб. Однако, если сервер и клиент поддерживают расширение стандарта, то максимально допустимый размер файла увеличивается до 4 Гб.

Выполнение заданий лабораторной работы предлагается производить с помощью расширенного клиента `atftp`.

Для подключения к удаленному узлу в режиме оболочки, достаточно указать команде `atftp` IP-адрес удаленного узла или его доменное имя. Если серверная служба использует UDP-порт по умолчанию, то порт в опциях команды можно опустить. Например, команда в лист. 5.1 произведет подключение к узлу 10.10.15.1, на UDP-порт 69, и, в случае успешного соединения, переведет эмулятор терминала в режим командной оболочки клиента `tftp`.

Листинг 5.1

Подключение к TFPT-серверу

```
user@host:[~]$ atftp 10.10.15.1
tftp> ...
```

С помощью команды `status` (лист. 5.2), можно вывести информацию о текущем соединении.

Листинг 5.2

Статус соединения

```
tftp> status
Connected:  10.10.15.1 port 69
Mode:       octet
Verbose:    off
Trace:      off
Options
 tsize:     disabled
 blksize:   disabled
 timeout:   disabled
 multicast: disabled
mtftp variables
 client-port:   76
 mcast-ip:      0.0.0.0
 listen-delay:  2
 timeout-delay: 2
```

Список всех доступных команд в режиме оболочки `tftp` выводится командой `help`.

Наиболее распространенный способ работы с сервером TFTP, заключается в непосредственном указании команде `atftp` имени файла, требуемого для загрузки на локальную систему. В таком случае командная конструкция принимает вид, показанный в лист. 5.3.

Листинг 5.3

Загрузка файла из сети (общий вид команды)

```
atftp --get -r remote-file-name -l local-file-name IP-address
```

После ключей `-r` и `-l` указываются имя файла на удаленной системе и имя файла, под которым его следует сохранить на локальной системе, соответственно. Например, чтобы загрузить файл `firmware.bin` с удаленного сервера `10.0.0.1` и сохранить файл под именем `update.bin` на локальной системе, можно использовать следующую команду, показанную в лист. 5.4.

Листинг 5.4

Загрузка файла из сети (пример)

```
user@host:[~]$ atftp --get -r firmware.bin -l update.bin 10.0.0.1
```

Для загрузки файла на сервер TFTP используется опция `--put`. В случае наличия разрешения на запись в корневой каталог сервера, можно загрузить файл, воспользовавшись командой, показанной в лист. 5.5.

Листинг 5.5

Загрузка файла на сервер TFTP (общий вид команды)

```
atftp --put -r remote-file-name -l local-file-name IP-address
```

Смысл ключей `-r` и `-l` сохраняется.

5.3.2. Протокол FTP

Принцип взаимодействия клиент-серверного ПО и алгоритм работы протокола FTP расмотрен ранее, поэтому данный раздел посвящен описанию функциональных возможностей стандартного клиента `ftp`.

Консольный клиент `ftp` — это стандартное, межплатформенное ПО для работы с удаленной службой FTP. Для запуска клиента достаточно указать команде `ftp` IP-адрес (или доменное имя) узла, к которому необходимо произвести подключение. Процедура установления соединения завершится выводом приветственных (информационных) данных и приглашением к авторизации пользователя. Например, как в лист. 5.6.

Листинг 5.6

Использование `ftp`

```
user@host:[~]$ ftp 10.0.0.2
...
Connected to 10.0.0.2.
220- Welcome to private ftp-service.
220- 10.0.0.2 FTP server (Version 6.00LS) ready.
Name (10.0.0.2:user):
..
```

Для подключения к серверу FTP в пассивном режиме передачи следует воспользоваться опцией `-p`. Следует напомнить, что транспортное соединение FTP между узлами по умолчанию устанавливается в активном режиме. По традиции опция `-v` позволяет вывести дополнительную информацию (так называемый режим подробного вывода).

После выполнения процедур авторизации клиент `ftp` переключается в режим командной оболочки, сигнализирующий о готовности принимать непосредственные команды пользователя (лист. 5.7).

Листинг 5.7

Оболочка `ftp`

```
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
...
```

Команда `help` выводит список всех доступных директив.

Команда `help` *`встроенная_команда`* — позволяет получить справку по использованию конкретной команды.

Наиболее востребованные директивы перечислены в табл. 5.1.

Таблица 5.1

Краткий список встроенных команд клиента `ftp`

Опция	**Назначение**
`ascii`	Текстовый режим передачи файлов
`binary`	Двоичный режим передачи файлов (вкл. по умолчанию)
`passive`	Пассивный режим соединения (по умолчанию вкл. активный)
`bye`	Завершить сеанс связи (с небольшими различиями аналогична командам `quit`, `disconnect`, `exit`)
`cd`	Перейти в указанный каталог (`cd` *`/путь_назначения`*)
`delete`	Удалить указанный файл (`delete` *`имя_файла`*)
`dir`	Список содержимого указанного каталога
`get`	Загрузка указанного файла с удаленного сервера на локальный хост
`mdelete`	Удаление нескольких указанных файлов
`mget`	Загрузка нескольких указанных файлов с сервера
`mkdir`	Создание каталога с указанным именем
`mput`	Загрузка указанных файлов с локального узла на удаленный сервер
`put`	Загрузка указанного файла на удаленный сервер
`pwd`	Указывает текущий каталог
`rstatus`	Статус удаленного сервера
`rmdir`	Удаление указанного каталога (перед удалением каталог необходимо очистить от файлов)
`status`	Статус подключения локального узла
`system`	Версия ПО сервера FTP

Для копирования файла из удаленного каталога сервера используется команда `get`, с опциями, указывающими имя и путь до файла на удаленном сервере и название файла, под которым следует сохранить копию на локальном узле. Например, в лист. 5.8 произведено копирование файла `image.iso`

из каталога /pub в локальный каталог узла /home/user/iso/ с именем файла image.iso.

Листинг 5.8

Копирование файлов

```
ftp> get /pub/image.iso /home/user/iso/image.iso
```

Загрузка файлов в удаленный каталог сервера происходит аналогичным образом. Команде put необходимо передать путь до передаваемого файла и указать место и имя, под которым файл будет записан в системе удаленного сервера (лист. 5.9).

Листинг 5.9

Загрузка файлов на сервер

```
ftp> put /home/user/updates/changelogs.txt /uploads/changelogs.txt
```

Применение прочих команд аналогично рассмотренным примерам. Следует отметить лишь необходимость понимания текущего местоположения клиента ftp в пределах корневого каталога удаленной системы. Как указано выше, для перемещения по структуре каталогов используется команда cd, которой в качестве опции передается имя каталога на удаленной системе (например, cd /uploads, cd /pub/bin/ и проч.).

5.3.3. Протокол SFTP

Как уже было упомянуто выше, помимо реализации всех функций стандартного клиента FTP, протокол SFTP поддерживает туннелирование сетевого трафика в сеанс SSH и аутентификацию пользователя по нескольким защищенным схемам. Практические аспекты работы с клиентом sftp из рассматриваемого ранее пакета OpenSSH в целом не отличаются от применения стандартного клиента ftp. Основное отличие заключается в методах аутентификации подключаемых пользователей.

Основные опции команды sftp (табл. 5.2) сходны по логике действия с рассматриваемыми ранее опциями клиента удаленного доступа ssh.

Таблица 5.2

Краткий список опций команды sftp

Опция	Назначение
-1	Использовать туннелирование по протоколу SSH версии 1
-2	Использовать тунеллирование по протоколу SSH версии 2 (вкл. по умолчанию)
-4	Использовать только адреса IPv4
-6	Использовать только адреса IPv6
-c blowfish|3des|des	Указать алгоритм для шифрования сеанса SSH
-o	Указать специальные опции ssh

Окончание табл. 5.2

Краткий список опций команды `sftp`

Опция	Назначение
`-P`	Указать порт службы `ssh` на удаленном узле
`-r`	Использовать рекурсивный обход каталогов при передаче файлов между удаленными узлами
`-q`	Не выводить подробную информацию о подключении
`-v`	Выводить подробную информацию о подключении

Для подключения к удаленной службе достаточно указать в качестве параметров команды `sftp` имя пользователя и IP-адрес сервера в стандартном формате — `sftp` *`учетная_запись`*`@10.0.0.1`. Если указанная учетная запись пользователя активна (не заблокирована) и существует на удаленной системе, последует приглашение ввести пароль (лист. 5.10). Как и в предыдущих случаях, успешная авторизация оканчивается переключением клиента в режим командной оболочки с соответствующим приглашением клиента `sftp` к вводу команд.

Листинг 5.10

Оболочка `sftp`

```
user@host:[~]$ sftp user@10.0.0.1
Password: ********
...
sftp>
```

Если в домашнем каталоге (на локальном узле) указанной учетной записи (в данном случае `user`) существует зашифрованный приватный ключ (`id_rsa`) SSH, а на удаленной системе присутствует его отпечаток в виде публичного ключа (`id_rsa.pub`), то вход на удаленный сервер будет осуществлятся с помощью процедуры подтверждения ключей, что не только исключает необходимость ввода пароля, но и существенно увеличивает степень безопасности удаленного подключения. Подробнее о процедурах генерации зашифрованных ключей и способе их обмена со службой удаленного доступа `ssh` — в п. 4.3.2.

Встроенные команды клиента `sftp` аналогичны по функциональности командам стандартного клиента (табл. 5.1). Список встроенных команд извлекается директивой `help`. Наиболее распространенные команды клиента `sftp` приведены в табл. 5.3.

Таблица 5.3

Краткий список встроенных команд клиента `sftp`

Опция	Назначение
`bye`	Завершить сеанс связи (с небольшими различиями аналогична командам `quit`, `exit`)

Окончание табл. 5.3

Краткий список встроенных команд клиента `sftp`

Опция	**Назначение**
`cd`	Осуществить переход в указанный каталог на удаленном сервере (`cd` *`/путь_назначения`*)
`lcd`	Осуществить переход в указанный каталог на локальном узле (`lcd` *`/путь_назначения`*)
`get`	Загрузка указанного файла с удаленного сервера на локальный хост (с опцией `-r` будет произведено рекурсивное копирование каталога
`mkdir`	Создание каталога с указанным именем
`put`	Загрузка указанного файла на удаленный сервер (с опцией `-r` будет произведено рекурсивное копирование каталога)
`pwd`	Указывает текущий каталог на удаленном сервере
`lpwd`	Указывает текущий каталог на локальном узле
`rmdir`	Удаление указанного каталога
`rm`	Удаление указанного файла

Работа с файлами на удаленной системе основополагается на правах и разрешениях используемой учетной записи пользователя. Таким образом, например, если текущему пользователю разрешена запись в конкретный каталог, то становится возможным модифицировать его содержимое. В случае, если доступные права ограничиваются только чтением (и исполнением), то допускается копирование файлов, но не их изменение. Просмотреть права текущего пользователя по отношению к содержимому каталога (или самой директории) можно с помощью команды `ls` с опциями `-la`, примененной в оболочке клиента (лист. 5.11).

Листинг 5.11

Просмотр прав пользователя в текущем каталоге

```
sftp> ls -la
drwxr-xr-x  6 user staff       512 Sep 29 00:50 .
drwxr-xr-x  9 root wheel      1024 Sep 30 13:41 ..
-rw-r--r--  1 user staff   4034695 Sep 13 22:54 textbook.pdf
-rw-r--r--  1 user staff       403 Sep  2 23:20 footprint.txt
drwxr-xr-x 11 user staff       512 Sep 26 17:03 stock
```

В примере из лист. 5.11, файлы `textbook.pdf`, `footprint.txt` и каталог `stock` принадлежат пользователю `user` и группе `staff`, о чем свидетельствуют записи в 3 («пользователь») и 4 («группа») колонке листинга. Символ точки в конце первой строки также указывает, что текущий каталог принадлежит указанному пользователю. Буквы латинского алфавита, перечисленные в первой колонке каждой строки, обозначают права, соответствующие владельцу, участнику группы и прочим лицам. В данном случае владелец файлов (пользователь `user`) имеет право на чтение (`r`), запись (`w`) и исполнение (`x`). Участники пользовательской группы `staff` имеют права на чтение (`r`) и и исполнение

(x). Прочие пользователи имеют право лишь на чтение файлов и исполнение команд в директории (например, переход по каталогам, вывод содержимого и проч.). Символ d в начале строки указывает, что данная запись является каталогом (dir).

5.4. Порядок выполнения лабораторной работы

1. Открыть эмулятор терминала командной оболочки ОС и запустить анализатор трафика tcpdump с фильтром дейтаграмм UDP, получаемых и передаваемых между сервером 172.16.100.88 и локальным узлом. Для этого следует использовать команду

```
user@host:[~]$ sudo tcpdump -vlnXX udp and \
> ip host 172.16.100.88 and ip host IP_NN | tee tftp.log
```

Переменную IP_NN следует заменить на номер локального ПК.

2. Открыть второй эмулятор терминала и, воспользовавшись клиентом atftp, следует загрузить файлы flash.bin и update.txt с удаленного сервера 172.16.100.88 в домашний каталог локального узла.

3. Убедившись в наличии копий указанных файлов, следует остановить сетевой анализатор пакетов и провести анализ данных, накопленных в файле tftp.log. Необходимо отметить UDP-дейтаграммы инициирующие подключение к серверу, а также выделить взаимодействующие сокеты.

4. Перезапустить анализатор tcpdump, настроив правила фильтрации на отображение трафика TCP между сервером 172.16.100.88 и локальным узлом. При этом следует указать конкретные порты назначения и отправки (TCP 20 и 21), применив командную конструкцию

```
user@host:[~]$ sudo tcpdump -vlnXX tcp and \
> ip host 172.16.100.88 and ip host IP_NN and \
> (dst port 21 or src port 21 or src port 20 or \
> dst port 20) | tee ftp_act.log
```

Данные, полученные, с помощью фильтра будут выводится на экран терминала и сохраняться в файле ftp_act.log.

5. Подключившись к удаленному серверу 172.16.100.88, используя стандартный клиент ftp, произвести копирование файла writeme.txt с сервера в домашний каталог на локальном узле. Для авторизации на сервере FTP следует использовать учетную запись с логином student_nn, где переменная nn обозначает номер лабораторного ПК (например, student_01, student_07 и т. д.). В качестве пароля использовать комбинацию stud. Открыв файл в текстовом редакторе leafpad, записать ФИО и строку «FTP в активном режиме передачи файлов».

6. Аналогично, используя клиент ftp, произвести загрузку модифицированного файла writeme.txt на FTP-сервер.

7. Остановить сбор сетевых пакетов анализатором tcpdump. Настроить правила фильтрации на сбор информации о передаче TCP-данных между узлами 172.16.100.88 и локальной системой, указав в данном случае порт TCP источника и назначения 21. Полученные данные следует выводить на экран терминала и сохранять в файле ftp_pasv.log. Для этого необходимо применить команду

```
user@host:[~]$ sudo tcpdump -vlnXX tcp and \
> ip host 172.16.100.88 and host IP_NN and \
> (dst port 21 or src port 21) | tee ftp_pasv.log
```

8. С помощью опции -p, явно указав клиенту ftp режим пассивной передачи данных, подключиться к серверу 172.16.100.88 и скопировать текстовый файл writemetoo.txt в домашний каталог локальной системы. Добавить ФИО и строку «FTP в пассивном режиме передачи данных» в файл writemetoo.txt, используя текстовый редактор leafpad.

9. Воспользовавшись клиентом ftp в пассивном режиме передачи данных, записать измененный файл на сервер. Остановить сбор сетевых пакетов анализатором tcpdump и проверить содержимое файлов ftp_act.log и ftp_pasv.log.

10. Используя фильтр, отбирающий пересылку TCP-данных с портом источником и назначения 22, запустить анализатор сетевых пакетов командой

```
user@host:[~]$ sudo tcpdump -vlnXX tcp and \
> ip host 172.16.100.88 and IP_NN and \
> (dst port 22 or src port 22) | tee sftp.log.
```

11. Командой sftp student_nn@172.16.100.88 произвести подключение к SFTP-серверу. Как и ранее, переменная nn обозначает номер лабораторного ПК.

12. После успешной авторизации с помощью встроенной в оболочку sftp команды mkdir создать каталог .ssh в домашнем каталоге пользователя на удаленной системе. Сгенерировав шифрованные ключи (п. 4.3.2 ЛР 4) на локальном узле, скопировать созданный публичный ключ (id_rsa.pub) в созданный каталог .ssh, переименовав его как authorized_keys. Разорвать соединение клиента sftp командой quit.

13. Повторно воспользовавшись командой

```
user@host:[~]$ sftp student_nn@172.16.100.88
```

подключиться к серверу SFTP, используя на этот раз автоматическую авторизацию по проверке шифрованных ключей. Отметить различия между двумя способами аутентификации.

14. Используя текстовый редактор leafpad, создать текстовый файл securesftp.txt и внести в него строку «Этот файл передан по защищенному

каналу SSH». Скопировать указанный файл на удаленный сервер, применив встроенную в клиент `sftp` команду `put`.

15. Создать каталог `data` на удаленном сервере и перенести в него файлы `writeme.txt`, `writemetoo.txt` и файл `securesftp.txt`. Снова используя команду `put`, загрузить файлы `flash.bin` и `update.txt` с локального узла в папку `data` удаленного сервера SFTP.

16. Остановить сбор сетевых пакетов анализатором `tcpdump` и проверить содержимое файла `sftp.log`, отметив при этом используемые при обмене файлами TCP-сокеты.

5.5. Содержание отчета

1. Заголовок, согласно приложению.
2. Цель работы.
3. Задание.
4. Экранные копии и листинг работы с командной оболочкой эмулятора терминала.
5. Журналы (`tftp.log`, `ftp_act.log`, `ftp_pasv.log` и `sftp.log`) передачи данных, собранные утилитой `tcpdump`.

5.6. Контрольные вопросы

1. Выделите основные отличия протоколов FTP от протокола прикладного уровня HTTP. Укажите основные области применения.

2. Перечислите способы аутентификации пользователей в рассмотренных протоколах — TFTP, FTP, SFTP. Обозначьте положительные и отрицательные стороны перечисленных методов.

3. Назовите зарегистрированные (стандартные) номера портов рассмотренных протоколов передачи файлов. Дайте определение понятию «сокет».

4. Используя диаграммы в описании лабораторной работы, оформите алгоритмы работы протоколов TFTP, FTP, SFTP.

5. Укажите основные отличия в активном и пассивном режиме передачи данных протокола FTP. Объясните необходимость наличия обоих режимов передачи данных.

6. В лабораторной работе рассматривалось стандартное ПО систем телеобработки данных с консольным интерфейсом (интерфейсом командной строки). Перечислите аналогичное ПО, использующее графический интерфейс пользователя.

7. Опишите преимущества и недостатки рассматриваемого стандартного ПО СТД для передачи файлов в сравнении с указанными утилитами, использующими графический интерфейс.

Лабораторная работа 6
Протокол сетевой удаленной синхронизации каталогов и дистрибуции данных

6.1. Цель работы

В лабораторной работе рассматриваются принципы работы прикладного ПО, предназначенного для выполнения задач сетевой синхронизации и резервирования информации между удаленными системами обработки данных.

6.2. Задачи

Создать локальную копию заданной структуры каталогов, произвести сетевую синхронизацию с удаленным сервером обработки данных, синхронизировать каталоги пользователей соседних лабораторных компьютеров с помощью центрального сетевого сервера и выполнить очистку сетевой и локальной копии обрабатываемых каталогов.

6.3. Теоретический материал

Одной из главных функций современных систем обработки данных является задача синхронизации и хранения резервных копий обработанной информации как в пределах структуры сети распределенной СОД (локально), так и на системах удаленного доступа. Располагая несколькими распределенными архивами данных, возможно обеспечить параллельную, бесперебойную обработку и передачу информации между узлами сети с высокой степенью защищенности.

Отказоустойчивость системы обработки данных складывается из нескольких слагаемых — степень избыточности самой информации, количество копий архивов и распределенность систем обработки и хранения данных, защищенность каналов передачи данных от ошибок и воздействия внешних помех, количество резервных линий (транков) и т. д. В основные функции ПО синхронизации и резервного хранения информации СОД входят задачи обеспечения своевременной синхронизации данных между узлами распределенной сети, корректная обработка обновлений в структуре архива резервного хранения и предоставление сетевого доступа к копии информации, обрабатываемой СОД.

В качестве примера функциональных возможностей ПО синхронизации и резервного хранения информации систем обработки данных, на рис. 6.1 приведен случай практического применения данного ПО в качестве инструмента сетевой передачи и архивного хранения пользовательских данных на удаленной распределенной системе обработки данных (РСОД).

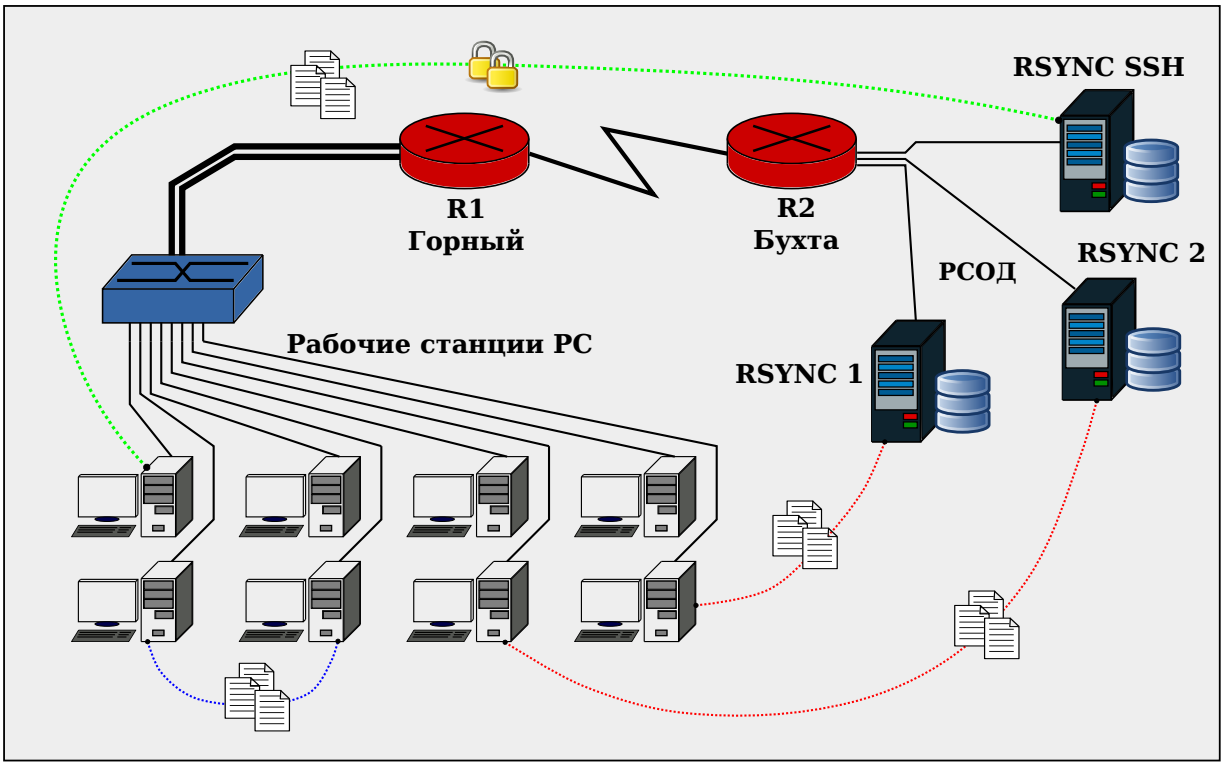

Рис. 6.1. Хранение информации на удаленных системах обработки данных

В лабораторной работе для целей практического ознакомления с рассматриваемым классом ПО СОД будет использоваться прикладной пакет `rsync`.

Серверный прикладной процесс службы `rsync` ожидает подключений от клиентского ПО, используя 873 порт TCP. В качестве протокола транспортного уровня может быть использован как протокол TCP, так и протокол UDP. Отличительной особенностью рассматриваемого ПО системы обработки данных является минимизация передачи служебного и пользовательского трафика. Это возможно за счет вычисления различий в структуре синхронизирующихся каталогов и передачи только тех частей информации (файлов), которые или отсутствуют, или были изменены в структуре системы назначения (сервера или клиента). Данный фактор можно считать решающим при необходимости синхронизации и резервного копирования файлов между системами, объединенными линиями связи с большими задержками (любой вид модемного подключения — Dial-up, DSL, беспроводное сотовое подключение и т. д.). В качестве дополнительной опции следует отметить так же поддержку сжатия передаваемых данных.

Помимо обеспечения средств сетевой синхронизации и резервного хранения, ПО `rsync` может использоваться для синхронизации данных расположенных на локальных СОД.

В общем случае командная конструкция для выполнения синхронизации посредством `rsync` должна содержать исходный путь и целевую систе-

му назначения. В лист. 6.1 приведен пример загрузки копии дерева каталогов с удаленного сервера `remote.host` из каталога `backup` в локальный каталог пользователя `/home/user`.

Листинг 6.1

Пример сетевой синхронизации каталогов

```
user@host:[~]$ rsync -avz remote.host::backup/ /home/user
```

Расшифровка опций, примененных к команде `rsync` в лист. 6.1 приведена в табл. 6.1. Следует обратить особое внимание, что подключение к удаленному модулю «backup» на системе `remote` происходит под учетной записью локального пользователя, т. е. в данном случае `user`. Если необходимо использовать специальную учетную запись (например, учетную запись пользователя на удаленной системе), в конструкции из лист. 6.1 необходимо явно указать требуемую учетную запись. Например, как в лист. 6.2.

Листинг 6.2

Явное указание учетной записи

```
user@host:[~]$ rsync -avz staff@remote.host::backup/ /home/user
```

Таким образом для подключения к удаленному модулю «backup» будет использоваться учетная запись пользователя `staff`, а не запись локального пользователя (`user`).

В конструкции команды используется символ двойного двоеточия «::» для строгого указания команде `rsync` произвести подключение к 873 порту TCP удаленной системы. При указании исходной системы в формате

```
remote.host:backup/
```

клиент `rsync` попытается установить соединение с оболочкой удаленного доступа (`ssh`) указанной системы, и использовать данную службу в качестве транспорта. Важной особенностью также является присутствие символа косой черты «/» в конце исходного пути — в данном случае, команде `rsync` явно указывается произвести синхронизацию содержимого каталога `backup`. В противном случае будет произведено копирование самого каталога `backup` в локальный каталог `/home/user`.

Таблица 6.1

Краткий список опций команды `rsync`

Опция	Назначение
`-a`	Выполнить рекурсивную обработку и осуществить синхронизацию всего содержимого каталогов
`-b`	Выполнить резервное копирование файлов перед их копированием
`--backup-dir=каталог`	При использовании с опцией `-b` указывает каталог, в котором необходимо сохранить резервные копии

Продолжение табл. 6.1

Краткий список опций команды `rsync`

Опция	Назначение
`-c`	Использовать 128-разрядные контрольные суммы MD4, для повышения надежности передачи
`--progress`	Выводить информацию о ходе выполнения передачи
`-z`	Выполнить сжатие данных перед отправкой на удаленную систему
`-q`	Не показывать ход выполнения передачи данных
`-r`	Выполнять рекурсивное копирование каталогов
`-v`	Режим подробного вывода

Помимо задач сетевой синхронизации данных, `rsync` может быть использован для сохранения структур локальных каталогов. Конструкция команды, в данном случае, будет состоять из локальных путей до синхронизируемых объектов. Например, для синхронизации данных, хранящихся в каталоге `rootdir` (исходный каталог) и `backupdir` (целевой каталог) можно использовать команду из лист. 6.3.

Листинг 6.3

Пример синхронизации локальных каталогов

```
user@host:[~]$ rsync -avr --delete ~/rootdir/ ~/backupdir/
```

В лист. 6.3 используется опция `delete`, которая указывает команде `rsync` удалить каталоги из директории `backupdir`, отсутствующие в директории исходного каталога (`rootdir`). Следует особо отметить наличие завершающего символа косой черты «/» — как уже было отмечено ранее, это приведет к копированию содержимого каталога, а не самого каталога. Символ «∼» в конструкции пути, представляет собой сокращенную форму записи домашнего каталога пользователя, и в данном случае эквивалентен записи `/home/user/`.

6.4. Порядок выполнения лабораторной работы

1. Открыть эмулятор терминала ОС и запустить в нем анализатор трафика `tcpdump` с фильтром пакетов, получаемых и передаваемых от узла `172.16.100.88` с TCP-портом источника или назначения 873. С помощью команды `tee`, вывести отфильтрованные IP-пакеты на экран эмулятора терминал и сохранить данные в файл `rsync.log`, в домашнем каталоге пользователя. Для этого следует воспользоваться командой

```
user@host:[~]$ sudo tcpdump -lvnXX host 172.16.100.88 and \
> host IP_NN and tcp and (src port 873 or dst port 873) | \
> tee rsync.log
```

где переменная IP_NN обозначает IP-адрес локального интерфеса ПК.

2. С помощью файлового менеджера `mc` проверить наличие и иерархию каталогов в домашней директории пользователя на локальном узле, изображенную на диаграмме рис. 6.2. Добавить индивидуальные данные в файл `name.txt` (ФИО, группа) и `address.txt` (номер лабораторного ПК). В домашнем каталоге пользователя создать папку с именем `backup_nn`, где переменные `nn` определяют номер лабораторного ПК (например, `backup_01` или `backup_10`).

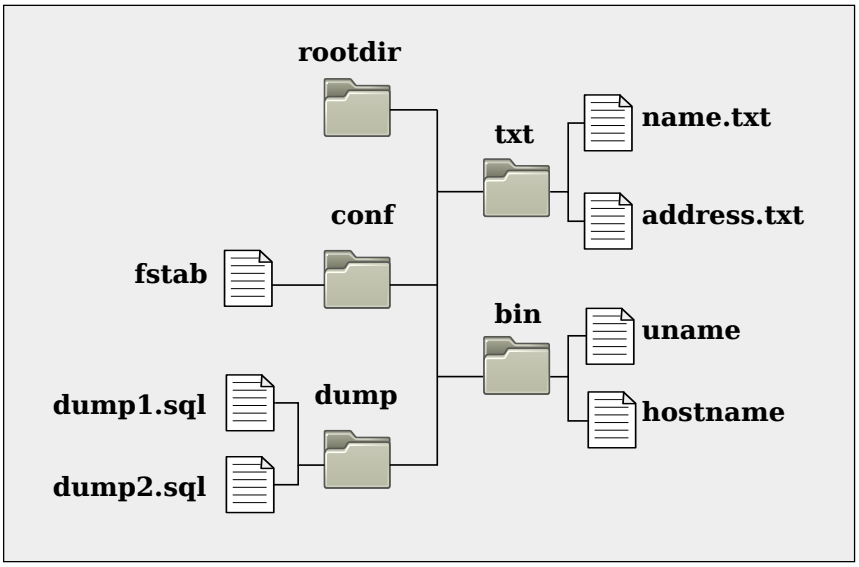

Рис. 6.2. Структура корневого каталога

3. Используя команду `rsync` и опции из табл. 6.1, произвести синхронизацию содержимого каталога `rootdir` и `backup_nn`, создав, таким образом, локальную архивную копию файлов. С помощью используемой ранее команды `tee` вывести данные в окно терминала и сохранить в текстовый файл `rsync_local.log`. Для этого можно воспользоваться следующей командой:

```
user@host:[~]$ rsync -avr --progress /home/student/rootdir/ \
> /home/student/backup_nn/ | tee rsync_local.log
```

После выполнения процедуры, проверить наличие копий файлов в директории `backup_nn`.

4. Произвести сетевую синхронизацию содержимого локального каталога `backup_nn` с удаленным хранилищем данных на кафедральном сервере `172.16.100.88` и сохранить протокол сетевой передачи в текстовый файл `rsync_remote.log`. Для подключения к модулю синхронизации следует использовать учетную запись удаленного пользователя с именем `student_nn` (значение `nn` в данном случае должно быть идентично значению переменной `nn` из предыдущих пунктов задания). Для этого необходимо использовать команду

```
user@host:[~]$ rsync -avrz --progress /home/student/backup_nn \
> student_nn@172.16.100.88::students | tee rsync_remote.log
```

В качестве пароля для доступа к удаленному модулю синхронизации students следует использовать комбинацию stud. В терминале сетевого монитора проследить процедуры установления соединения с серверной службой rsync, передачу данных и отметить TCP-порты, участвующие в обмене информацией на обоих узлах.

5. Создать каталог nethood на локальном узле и загрузить данных с сетевого хранилища из указанного каталога, синхронизированного с соседнего лабораторного компьютера. Для этого следует воспользоваться командой

```
user@host:[~]$ rsync -avzr --progress \
> student_nn@172.16.100.88::students/backup_nn \
> /home/student/nethood
```

где в качестве переменной nn следует указать каталог backup, расположенный на сетевом хранилище 172.16.100.88, с номером, отстоящим на две цифры далее локального номера лабораторного ПК (например, backup_10 или backup_02 и т. д.).

6. Используя файловый менеджер mc, проверить наличие новых текстовых файлов name.txt и address.txt в структуре каталога nethood, убедившись в их принадлежности соответствующему удаленному узлу. В панели сетевого монитора tcpdump отметить процедуры передачи и приема сетевых пакетов между указанными узлами сети.

7. По завершению всех процедур обмена файлами, очистить содержимое локальных и удаленных каталогов на сетевом хранилище. Для этого, с помощью файлового менеджера mc очистить локальные каталоги, а затем произвести процедуры сетевой синхронизации с сервером, указав команде rsync опцию --delete. Например,

```
user@host:[~]$ rsync -avr --progress --delete \
> /home/student/backup_nn student_nn@172.16.100.88::students
```

6.5. Содержание отчета

1. Заголовок, согласно приложению.
2. Цель работы.
3. Задание, согласно варианту.
4. Экранные копии и листинг работы с командной оболочкой эмулятора терминала.
5. Журналы (rsync.log, rsync_local.log и rsync_remote.log) передачи данных, собранные утилитой tcpdump.

6.6. Контрольные вопросы

1. В краткой форме, опишите алгоритм манипуляции данными, лежащий в основе рассматриваемого ПО.

2. Приведите пример альтернативных служб, применимых для выполнения задач сетевой синхронизации и резервирования данных.

3. Укажите два способа синхронизации данных с помощью клиента `rsync` и сетевых служб, выполняющихся на сервере. Определите используемые TCP-порты.

4. Выделите основные отличия в алгоритме работы и назначении ПО `rsync` и распространенной сетевой службы сети Internet — `FTP`.

5. Воспользовавшись заданиями лабораторной работы, приведите практический пример использования рассматриваемого ПО.

Приложение

Правила оформления отчета к лабораторным работам

1. Структура отчета должна соответствовать требованиям представленным в соответствующем пункте лабораторной работы.
2. Размер основного шрифта отчета: 11–12 pt.
3. Заголовок отчета должен иметь вид:

Отчет к лабораторной работе №1
Изучение принципов работы утилит
для исследования и мониторинга состояния сети

Группа: ГР-00
Студент: Пупкин В. И.

Цель работы: ...

4. Результаты работы консольных программ (листинги), сами запускаемые команды и диаграммы, отображаемые в текстовом виде, должны быть оформлены моноширинным шрифтом (Courier New, Lusida Console, FreeMono и т. п.). Они должны вмещаться в ширину страницы (шрифт можно уменьшать до 9 pt). Если ширины вертикально расположенного листа A4 не хватает, то можно разместить диаграмму на нескольких горизонтально расположенных листах A4.

Например:

```
student@comp:~\$ ping -c 4 www.ya.ru
PING ya.ru (87.250.250.3) 56(84) bytes of data.
64 bytes from www.yandex.ru (87.250.250.3): icmp_seq=1 ttl=52 time=16.8 ms
64 bytes from www.yandex.ru (87.250.250.3): icmp_seq=2 ttl=52 time=16.8 ms
64 bytes from www.yandex.ru (87.250.250.3): icmp_seq=3 ttl=52 time=18.7 ms
64 bytes from www.yandex.ru (87.250.250.3): icmp_seq=4 ttl=52 time=13.5 ms

--- ya.ru ping statistics ---
4 packets transmitted, 4 received, 0\% packet loss, time 3014ms
rtt min/avg/max/mdev = 13.542/16.484/18.759/1.874 ms
```

5. Текст на диаграммах и графиках должен быть свободно читаем.
6. На графиках должны быть подписаны оси и единицы измерения.

Владимиров Сергей Сергеевич
Небаев Игорь Алексеевич

ИНТЕРНЕТ-ТЕХНОЛОГИИ И МУЛЬТИМЕДИА

Лабораторный практикум

Редактор *Л. К. Паршина*
Компьютерная верстка *Е. А. Головинская*

План изданий 2015 г., п. 24

Подписано к печати 23.01.2015
Объем 3,75 усл.-печ. л. Тираж 15 экз. Заказ 538

Редакционно-издательский центр СПбГУТ
191186 СПб., наб. р. Мойки, 61
Отпечатано в СПбГУТ