Глава 3
Диагностика сети в Linux

В этой главе мы обсудим основы того, как работают сети, а также как устранять неполадки в сети с помощью рабочей станции Linux. К концу главы вы овладеете инструментами для устранения неполадок в локальных и удаленных сетевых службах, а также научитесь проводить полную инвентаризацию вашей сети и ее служб.

В частности, мы рассмотрим следующие темы:

- Основы функционирования сети: сетевая модель OSI.
- Уровень 2: связь между адресами IP и MAC с помощью ARP, а также более подробная информация о MAC-адресах.
- Уровень 4: как работают порты TCP и UDP, включая трехэтапное квитирование TCP и соответствующие команды Linux.
- Сканирование локальных портов TCP и UDP и их связь с запущенными службами.
- Сканирование удаленных портов с помощью двух видов команд, встроенных в Linux.
- Сканирование удаленных портов с помощью установленных средств сканирования (в частности, Netcat и Nmap).
- В конце главы мы рассмотрим основы беспроводных сетей и способы устранения их неполадок.

Технические требования

Чтобы запускать примеры из этого раздела, мы будем использовать уже существующий узел с ОС Ubuntu или **виртуальную машину (VM)**. В этой главе мы также затронем тему беспроводной связи, поэтому если у вас нет беспроводной сетевой карты на узле или на виртуальной машине, то для работы с примерами вам понадобится адаптер Wi-Fi.

Изучая методы устранения неполадок, мы будем использовать ряд инструментов, начиная со встроенных команд Linux:

`arp`	Работает с **протоколом разрешения адреса (ARP**, Address Resolution Protocol) и локальной таблицей ARP в памяти. Это позволяет связать физический MAC-адрес сетевой карты с ее IP-адресом
`netplan`	Инструмент на основе YAML для настройки сетевых параметров
`ip` и `ifconfig`	Настраивают и отображают различные параметры на локальных сетевых интерфейсах вашего узла
`netstat` и `ss`	Отображают прослушивающие порты TCP и UDP на узле и связывают их с запущенными процессами. Также выводят состояние входящих и исходящих TCP-соединений вашего узла
`telnet`	Несмотря на то что `telnet` в качестве терминального приложения вызывает неодобрение из-за своего открытого протокола обмена, в некоторых ситуациях это удобный инструмент для устранения неполадок
`nc` (Netcat)	Netcat не только содержит все инструменты для устранения неполадок, которые доступны в `telnet`, но и значительно расширяет их. С помощью этой команды можно подключаться к удаленным службам и глубоко анализировать их, а также размещать службы на локальном узле для тестирования

Мы также будем использовать дополнительные приложения:

Nmap	Сканирует и тестирует прослушивающие порты на удаленных узлах, а также запускает различные скрипты, которые обращаются к этим портам
Kismet	Отображает сведения о локальных беспроводных сетях, не подключаясь к ним. Kismet — это инструмент с текстовым интерфейсом, поэтому его можно запускать как локально, так и из сеанса SSH
Wavemon	Отображает сведения о беспроводной сети, к которой вы подключены, в частности мощность сигнала и другие показатели, связанные с производительностью
LinSSID	LinSSID — это шаг вперед по сравнению с Kismet, с великолепным графическим представлением уровня сигнала и загруженности каналов ближайших беспроводных сетей

Для работы с пакетами, которых по умолчанию нет в Ubuntu, вам понадобится подключение к интернету, чтобы устанавливать эти пакеты с помощью команд `apt`.

Основы функционирования сети: сетевая модель OSI

Работу сети и приложений удобно рассматривать с помощью многоуровневой модели, где верхние уровни отвечают за более высокоуровневые и абстрактные

функции, а нижние — за более примитивные. Следующая диаграмма описывает модель OSI в общих чертах:

Рис. 3.1. Модель OSI для сетевого взаимодействия с описаниями и примерами

На практике уровни часто обозначаются числами, считая снизу. Таким образом, проблемы на уровне 2 обычно связаны с MAC-адресами и коммутаторами и ограничены VLAN, в которой находится станция (что обычно означает локальную подсеть). Проблемы на уровне 3 связаны с IP-адресацией, маршрутизацией или пакетами (и следовательно, касаются маршрутизаторов и смежных подсетей более удаленных сетей).

Как и в любой модели, здесь присутствует неоднозначность. Например, исторически нет четкой границы между уровнями 6 и 7. А что касается уровней 5 и 6, то, скажем, протоколы IPsec определенно связаны с шифрованием и поэтому относятся к уровню 6, однако их можно считать и протоколами туннелирования (в зависимости от вашей точки зрения и от реализации). Даже на уровне 4 в TCP есть понятие сеанса, поэтому, возможно, он немного заходит на территорию уровня 5, хотя концепция портов прочно удерживает его на уровне 4.

И конечно же, дело не обходится без юмора: народная мудрость гласит, что уровень 8 в этой модели формируют *люди*. Таким образом, чтобы решить проблему уровня 8, может понадобиться позвонить в службу поддержки, обсудить бюджет или устроить совещание с руководством вашей организации.

На следующей диаграмме (рис. 3.2) проиллюстрирован чрезвычайно важный принцип модели OSI. При получении данных они перемещаются вверх по

стеку — от самых примитивных форм до все более абстрактных и высокоуровневых конструкций (например, от битов до кадров, пакетов, API и приложений). Наоборот, при отправке данных они перемещаются с прикладного уровня в сторону двоичного представления в физической среде передачи (то есть от верхних уровней к нижним).

Уровни 1–3 часто называют уровнями **среды** (media или network), а уровни 4–7 — уровнями **узла** (host или application):

Рис. 3.2. Перемещение вверх и вниз по стеку OSI с инкапсуляцией и декапсуляцией по мере продвижения

Этот принцип позволяет, например, одному производителю изготавливать коммутатор, который будет взаимодействовать с сетевой картой другого производителя, а коммутаторам — работать с маршрутизаторами. Модель OSI также гармонизирует экосистему приложений: по большей части их разработчикам не нужно беспокоиться об IP-адресах, маршрутизации или различиях между беспроводными и проводными сетями — все это уже обеспечено на других уровнях. Сеть можно рассматривать как черный ящик: вы отправляете данные на одном конце и можете быть уверены, что на другом конце они придут в нужное место и в нужном формате.

Теперь, когда мы освоили основы модели OSI, давайте подробно рассмотрим канальный уровень передачи данных, изучив команду `arp` и локальную таблицу ARP.

Уровень 2: связь между адресами IP и MAC с помощью ARP

Опираясь на модель OSI, легко заметить, что наше обсуждение IP-адресов до сих пор было сосредоточено вокруг уровня 3. Именно здесь начинаются и заканчиваются сетевые пути в понимании обычных людей и даже многих специалистов в области IT и сетей. Они доходят до этого уровня и считают все остальное черным ящиком. Но для профессиональных сетевых инженеров уровни 1 и 2 чрезвычайно важны, так что давайте начнем с уровня 2.

Теоретически MAC-адреса — это адреса, которые *зашиты* в каждый сетевой интерфейс. Хотя обычно это так, MAC-адрес довольно легко изменить. Что же такое MAC-адрес? Это 12-значный (6-байтовый/48-битный) адрес, который чаще всего отображается в шестнадцатеричном формате. При отображении байты или двойные байты, как правило, разделяются дефисом (`-`) или точкой (`.`). Таким образом, типичные MAC-адреса выглядят как `00-0c-29-3b-73-cb` или `9a93.5d84.5a69` (это два стандартных представления MAC-адресов).

На практике эти адреса используются для связи между узлами в одной VLAN или подсети. Если посмотреть на перехват пакетов (мы вернемся к этому позже, в главе 11 «Перехват и анализ пакетов в Linux»), можно увидеть, что в начале сеанса связи TCP отправитель рассылает широковещательный (то есть предназначенный для всех станций в подсети) **запрос ARP** вида «у кого в сети IP-адрес x.x.x.x?». **Ответ ARP** от узла с этим адресом будет иметь вид «Это я, а мой MAC-адрес — aaaa.bbbb.cccc». Если искомый IP-адрес находится в другой подсети, отправитель запросит шлюз для этой подсети (обычно это шлюз по умолчанию, если не определены локальные маршруты).

Далее отправитель и получатель обмениваются данными, используя MAC-адреса. Инфраструктура коммутатора, к которой подключены оба узла, использует MAC-адреса только внутри каждой VLAN, и это одна из причин, по которой коммутаторы намного быстрее маршрутизаторов. Когда мы посмотрим на фактические пакеты (в главе о перехвате пакетов), то увидим в каждом пакете MAC-адреса отправителя и получателя, а также их IP-адреса.

Данные ARP кэшируются на каждом узле в **кэше ARP** или в **таблице ARP**, которую можно просмотреть с помощью команды `arp`:

```
$ arp -a
? (192.168.122.138) at f0:ef:86:0f:5d:70 [ether] on ens33
? (192.168.122.174) at 00:c3:f4:88:8b:43 [ether] on ens33
```

```
? (192.168.122.5) at 00:5f:86:d7:e6:36 [ether] on ens33
? (192.168.122.132) at 64:f6:9d:e5:ef:60 [ether] on ens33
? (192.168.122.7) at c4:44:a0:2f:d4:c3 [ether] on ens33
_gateway (192.168.122.1) at 00:0c:29:3b:73:cb [ether] on ens33
```

Как видите, все это довольно просто. Команда `arp` связывает IP-адрес уровня 3 с MAC-адресом уровня 2 и **сетевой картой** уровня 1 (**NIC**, Network Interface Card). MAC-адреса, которые содержатся в таблице, обычно выясняются на основании трафика — как запросов, так и ответов ARP. Срок их действия не бесконечен: как правило, если для MAC-адреса не наблюдается входящего или исходящего трафика, то через небольшой промежуток времени адрес удаляется из таблицы. Продолжительность этого промежутка можно увидеть в соответствующем файле в каталоге `/proc`:

```
$ cat /proc/sys/net/ipv4/neigh/default/gc_stale_time
60
$ cat /proc/sys/net/ipv4/neigh/ens33/gc_stale_time
60
```

Обратите внимание, что существует как значение по умолчанию (в секундах), так и значение для каждого сетевого адаптера (обычно они совпадают). Эти интервалы могут показаться слишком короткими, ведь срок действия таблицы сопоставления MAC-адресов (так называемой таблицы CAM) на коммутаторах обычно составляет 5 минут, а таблицы ARP на маршрутизаторах — 14 400 секунд (4 часа). Эти значения обусловлены ресурсами оборудования. У большинства рабочих станций достаточно ресурсов для того, чтобы часто отправлять пакеты ARP. Коммутаторы узна́ют MAC-адреса из трафика (в том числе из запросов и ответов ARP), поэтому для них имеет смысл установить более долгий срок действия таблицы, чем для рабочей станции. Аналогично длительный таймер кэширования ARP на маршрутизаторах экономит ресурсы их процессора и сетевой карты. Время ожидания на маршрутизаторах так велико, потому что в прошлом их пропускная способность и ресурсы процессора были весьма ограниченны по сравнению практически со всем остальным оборудованием в сети. Хотя в наше время это уже не так, на маршрутизаторах по умолчанию сохраняется большой срок действия кэша ARP. Об этом часто забывают, когда заменяют маршрутизатор или брандмауэр: я много раз занимался техническим обслуживанием такого типа, когда после замены оборудования команда `clear arp` на нужном маршрутизаторе волшебным образом решала все проблемы.

Мы еще не говорили о каталоге `/proc` в Linux — это виртуальный каталог файлов с текущими настройками и состояниями различных компонентов узла Linux. Это не настоящие файлы, но они ведут себя подобно файлам, поэтому для них можно использовать файловые команды `cat`, `grep`, `cut`, `sort`, `awk` и т. д. Например, в `/proc/net/dev` можно просмотреть ошибки и параметры сетевого интерфейса (обратите внимание, что в этом листинге строки плохо выравниваются):

```
$ cat /proc/net/dev
Inter-|   Receive
|   Transmit
 face |bytes    packets errs drop fifo frame compressed
multicast|bytes     packets errs drop fifo colls carrier
compressed
    lo:   208116     2234    0      0   0    0     0
0    208116    2234     0    0      0    0      0     0
  ens33: 255945718   383290    0    662   0    0     0
0  12013178   118882    0    0      0    0      0     0
```

Можно даже посмотреть статистику по используемой памяти (хотя `meminfo` содержит гораздо больше информации):

```
$ cat /proc/meminfo | grep Mem
MemTotal:        8026592 kB
MemFree:         3973124 kB
MemAvailable:    6171664 kB
```

Вернемся к ARP и MAC-адресам. Можно добавить статический MAC-адрес — такой, срок действия которого не истечет и который может отличаться от реального MAC-адреса узла, к которому вы подключаетесь. Так часто делают для устранения неполадок. Также можно очистить запись ARP, что может потребоваться, если маршрутизатор был заменен (например, если у вашего маршрутизатора-шлюза по умолчанию тот же IP-адрес, но теперь другой MAC-адрес). Обратите внимание, что для просмотра таблицы ARP не нужны специальные права, но, чтобы ее изменить, они определенно требуются!

Чтобы добавить статическую запись, сделайте так (обратите внимание на состояние `PERM`):

```
$ sudo arp -s 192.168.122.200 00:11:22:22:33:33
$ arp -a | grep 192.168.122.200
? (192.168.122.200) at 00:11:22:22:33:33 [ether] PERM on ens33
```

Чтобы удалить запись ARP, выполните следующие действия (обратите внимание, что параметр `-i interfacename` в этой команде обычно пропускается):

```
$ sudo arp -i ens33 -d 192.168.122.200
```

А так можно замаскироваться под заданный IP-адрес — например, чтобы отвечать на запросы ARP для IP `10.0.0.1`:

```
$ sudo arp -i eth0 -Ds 10.0.0.1 eth1 pub
```

Наконец, также можно легко изменить MAC-адрес интерфейса. Может показаться, что это помогает избежать дублирования адресов, однако такая ситуация встречается крайне редко. Вот несколько обоснованных причин менять MAC-адрес:

- Вы заменили брандмауэр, а ваш MAC-адрес жестко «зашит» у интернет-провайдера.
- Вы заменили узел или его сетевую карту, и вышестоящий маршрутизатор вам недоступен, однако вы не готовы ждать 4 часа, пока на нем истечет срок действия кэша ARP.
- Вы заменили узел, и в DHCP зарезервирован старый MAC-адрес, который вам нужно использовать, но у вас нет доступа к редактированию этой записи DHCP.
- Устройства Apple меняют свои беспроводные MAC-адреса по соображениям конфиденциальности. Однако с учетом того, сколько существует других (и более простых) методов отслеживать личность человека, эта защита обычно не так эффективна.

К злонамеренным причинам изменения MAC-адреса относятся такие:

- Вы атакуете беспроводную сеть и выяснили, что после аутентификации точка доступа проверяет только MAC-адреса клиентов.
- То же, что в предыдущем пункте, но вы атакуете сеть Ethernet, которая защищена аутентификацией 802.1x, но с небезопасной или неполной конфигурацией (мы рассмотрим это подробнее в следующей главе).
- Вы атакуете беспроводную сеть, доступ к которой разрешен только для определенных MAC-адресов.

Надеюсь, этот список наглядно демонстрирует, что использовать MAC-адреса в целях безопасности — обычно не лучшее решение.

Есть четыре способа выяснить свой MAC-адрес:

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc fq_ codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:0c:29:33:2d:05 brd ff:ff:ff:ff:ff:ff
$ ip link show ens33 | grep link
    link/ether 00:0c:29:33:2d:05 brd ff:ff:ff:ff:ff:ff
$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1400
        inet 192.168.122.22 netmask 255.255.255.0 broadcast
192.168.122.255
        inet6 fe80::1ed6:5b7f:5106:1509 prefixlen 64 scopeid 0x20<link>
        ether 00:0c:29:33:2d:05  txqueuelen 1000  (Ethernet)
        RX packets 384968 bytes 256118213 (256.1 MB)
        RX errors 0  dropped 671  overruns 0  frame 0
        TX packets 118956 bytes 12022334 (12.0 MB)
```

```
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 2241 bytes 208705 (208.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2241 bytes 208705 (208.7 KB)
        TX errors 0  dropped 0  overruns 0  carrier 0 collisions 0

$ ifconfig ens33 | grep ether
        ether 00:0c:29:33:2d:05  txqueuelen 1000  (Ethernet)
```

Чтобы изменить MAC-адрес узла Linux, есть несколько методов:

В графическом интерфейсе Linux можно начать с того, чтобы щелкнуть на значке сети на верхней панели, а затем выбрать пункт Настройки (Settings) для вашего интерфейса. Кроме того, для узла с одной картой Ethernet можно выбрать Проводное (Wired Connection), затем Настройки подключения (Wired Settings):

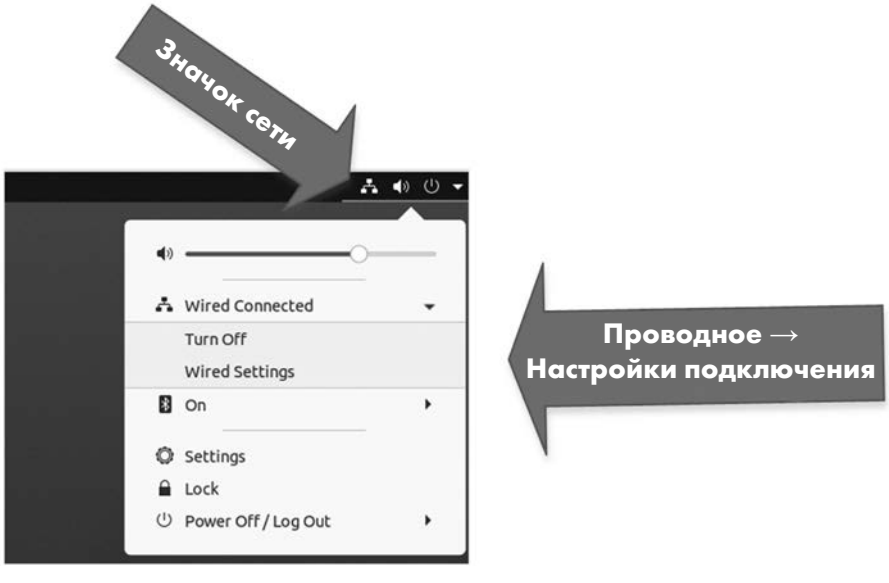

Рис. 3.3. Изменение MAC-адреса с помощью графического интерфейса, шаг 1

В появившемся окне откройте диалоговое окно Создать профиль (New Profile), щелкнув на значке «+», а затем просто добавьте MAC-адрес в поле Клонированный адрес (Cloned Address):

«+» Icon — Значок «+»

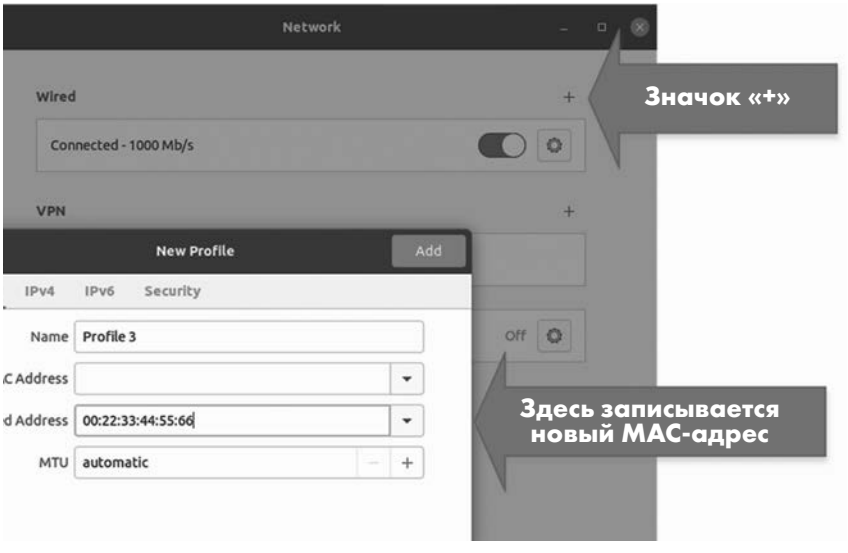

Рис. 3.4. Изменение MAC-адреса с помощью графического интерфейса, шаг 2

То же самое можно сделать из командной строки или с помощью скрипта таким образом (конечно, используя ваши имена интерфейсов и нужный MAC-адрес):

```
$ sudo ip link set dev ens33 down
$ sudo ip link set dev ens33 address 00:88:77:66:55:44
$ sudo ip link set dev ens33 up
```

Есть также утилита `macchanger`, с помощью которой можно изменить MAC-адрес вашего интерфейса на какое-то определенное или псевдослучайное значение.

Чтобы закрепить изменение MAC-адреса, можно использовать `netplan` и связанные с ним файлы конфигурации. Сначала сделайте резервную копию файла конфигурации `/etc/netplan/01-network-manager-all.yaml`, а затем отредактируйте его. Обратите внимание: чтобы изменить MAC-адрес, сначала нужно с помощью инструкции `match` указать MAC-адрес, встроенный в оборудование (BIA, Burned-In Address), а в следующей строке задать новый адрес:

```
network:
    version: 2
    ethernets:
      ens33:
          dhcp4: true
          match:
              macaddress: b6:22:eb:7b:92:44
          macaddress: xx:xx:xx:xx:xx:xx
```

Новую конфигурацию можно протестировать с помощью `sudo netplan try` и применить с помощью `sudo netplan apply`.

Кроме того, можно создать или отредактировать файл `/etc/udev/rules.d/75-mac-spoof.rules`, который будет выполняться при каждом запуске. Добавьте в него строку:

```
ACTION=="add", SUBSYSTEM=="net", ATTR{address}=="XX:XX:XX:XX:XX:XX",
RUN+="/usr/bin/ip link set dev ens33 address YY:YY:YY:YY:YY:YY"
```

После того как мы освоили основы использования MAC-адресов в ARP, давайте глубже рассмотрим, как MAC-адреса связаны с производителями различных сетевых адаптеров.

Уникальный идентификатор организации (OUI) в MAC-адресе

Итак, после того как мы рассмотрели ARP и сроки действия кэша, знаем ли мы все необходимое об уровне 2 и MAC-адресах? Еще нет! Давайте поговорим об **уникальном идентификаторе организации (OUI**, Organizationally Unique Identifier). Если вы помните, как IP-адреса разбиваются на сетевую и узловую части с помощью маски подсети, вы удивитесь, когда узнаете, что в MAC-адресах есть аналогичное разделение!

Старшие биты каждого MAC-адреса предназначены для того, чтобы идентифицировать производителя, — это значение называется OUI. Идентификаторы OUI зарегистрированы в официальном реестре, который поддерживает IEEE, и размещены по адресу http://standards-oui.ieee.org/oui.txt.

Стоит отметить, что проект Wireshark поддерживает более полный список по адресу https://gitlab.com/wireshark/wireshark/-/raw/master/manuf. Wireshark также предлагает веб-приложение для поиска по этому списку: https://www.wireshark.org/tools/oui-lookup.html.

Обычно MAC-адрес делится ровно пополам: первые 3 байта (6 символов) выделяются для OUI, а последние 3 байта — для уникальной идентификации устройства. Однако организации могут приобретать более длинные OUI (за меньшую плату), и в этом случае для устройств остается меньше адресов.

OUI — отличный инструмент для устранения неполадок в сети: когда возникают проблемы или появляются неизвестные станции, значения OUI могут помочь их идентифицировать. Знания об OUI пригодятся нам позже в этой главе, когда мы будем обсуждать сетевые сканеры (в частности, Nmap).

Если вам нужен синтаксический анализатор OUI с интерфейсом командной строки для Linux или Windows, я разместил его по адресу https://github.com/robvandenbrink/ouilookup.

На этом мы завершаем наш экскурс по уровню 2 модели OSI и изучение ее связи с уровнем 3. Давайте поднимемся на уровень 4 и рассмотрим протоколы TCP и UDP, а также связанные с ними службы.

Уровень 4: как работают порты TCP и UDP

Когда идет речь об уровне 4 модели OSI, и в частности о понятии *портов*, обычно имеют в виду **TCP (протокол управления передачей**, Transmission Control Protocol) и **UDP (протокол пользовательских датаграмм**, User Datagram Protocol).

Когда станция хочет связаться с другой станцией в той же подсети, используя IP-адрес получателя (этот адрес обычно определяется на прикладном уровне или на уровне представления), она ищет в своем кэше ARP MAC-адрес, соответствующий этому IP-адресу. Если для него нет записи, станция отправляет запрос ARP на локальный широковещательный адрес (как мы обсуждали в предыдущем разделе).

На следующем шаге протокол (TCP или UDP) устанавливает связь между портами. Станция выбирает доступный порт выше `1024` и ниже `65535` (максимальное значение порта), который называется **динамическим портом (ephemeral port)**. Затем она подключается с этого порта к фиксированному порту на сервере. Комбинация этих портов в сочетании с IP-адресами на обоих концах и используемым протоколом (либо TCP, либо UDP) всегда уникальна (из-за того, как выбран исходный порт) и называется **кортежем** (tuple). Эту структуру можно расширить, особенно в конфигурациях NetFlow, где добавляются различные параметры, например значения **QoS** (Quality of Service, **качество обслуживания**), **DSCP** (Differentiated Services Code Point, **кодовая точка дифференцированных услуг**) или **ToS** (Type of Service, **тип обслуживания**), имена приложений и интерфейсов и информация о маршрутизации, такая как **ASN** (Autonomous System Numbers, **номера автономных систем**), сведения об MPLS или VLAN, а также байты входящего и исходящего трафика. Из-за этой гибкости базовый кортеж из пяти значений, на котором строятся все остальные, часто называют **5-кортеж** (5-tuple).

Первые 1024 порта (пронумерованные от `0` до `1023`) почти никогда не бывают отправителями: они специально предназначены для использования в качестве серверных портов, и чтобы работать с ними, нужны права суперпользователя. Порты в диапазоне `1024–49151` считаются «пользовательскими», а `49152–65535` — динамическими или частными портами. Однако серверы не обязаны использовать порты с номерами ниже `1024`: например, почти каждый сервер баз данных использует более высокие порты. Номера серверных портов — просто традиция, которая восходит к тому времени, когда разрабатывались TCP и UDP, а все серверные порты были ниже `1024`. Если посмотреть на многие серверы, чья история начиналась в то время, можно увидеть такую картину:

DNS	udp/53, tcp/53
Telnet	tcp/23
SSH	tcp/22
FTP	tcp/20 и tcp/21
HTTP	tcp/80
HTTPS	tcp/443
SNMP	udp/162
Syslog	tcp/443

Полный список официально присвоенных портов поддерживается IANA и публикуется по адресу https://www.iana.org/assignments/service-names-port-number/service-names-port-numbers.xhtml. Соответствующая документация находится в RFC 6335.

На практике, однако, *присвоение* — слишком сильное слово для этого списка. Хотя было бы глупо размещать веб-сервер на порте TCP 53 или DNS-сервер на порте UDP 80, многих приложений вообще нет в этом списке. Если вы работаете с одним из таких приложений, просто выберите порт, который обычно свободен, и используйте его. Нередко поставщики используют порты, которые присвоены какой-то определенной службе в списке IANA, но назначают его малоизвестной или менее востребованной службе. Таким образом, по большей части этот список представляет собой набор рекомендаций с неявным намеком на то, что если поставщик выбирает хорошо известный порт для собственного использования, такого поставщика можно считать, скажем так, не очень разумным.

Уровень 4: TCP и трехэтапное квитирование

UDP просто берет значения из 5-кортежа и начинает отправлять данные. Принимающее приложение должно позаботиться о том, чтобы получить эти данные или проверить пакеты приложения с целью убедиться, что данные поступают в правильном порядке, а также обработать ошибки. На самом деле именно из-за отсутствия накладных расходов UDP так часто используется для приложений, критичных по времени, таких как VoIP (Voice over IP, голос поверх IP) и потоковое видео. Если в приложениях такого рода пропущен пакет, то при повторной попытке передачи поток данных прервется, и это заметит конечный пользователь, поэтому до некоторой степени ошибки просто игнорируются.

Зато TCP следит за порядком пакетов и поддерживает их порядковые номера по мере обмена данными. Это позволяет приложениям на основе TCP отслеживать отброшенные или поврежденные пакеты и повторно передавать их, параллельно

отправляя и получая другие данные. Первоначальное согласование всего этого процесса обычно называется **трехэтапным квитированием** («рукопожатием», handshake) — графически это выглядит примерно так:

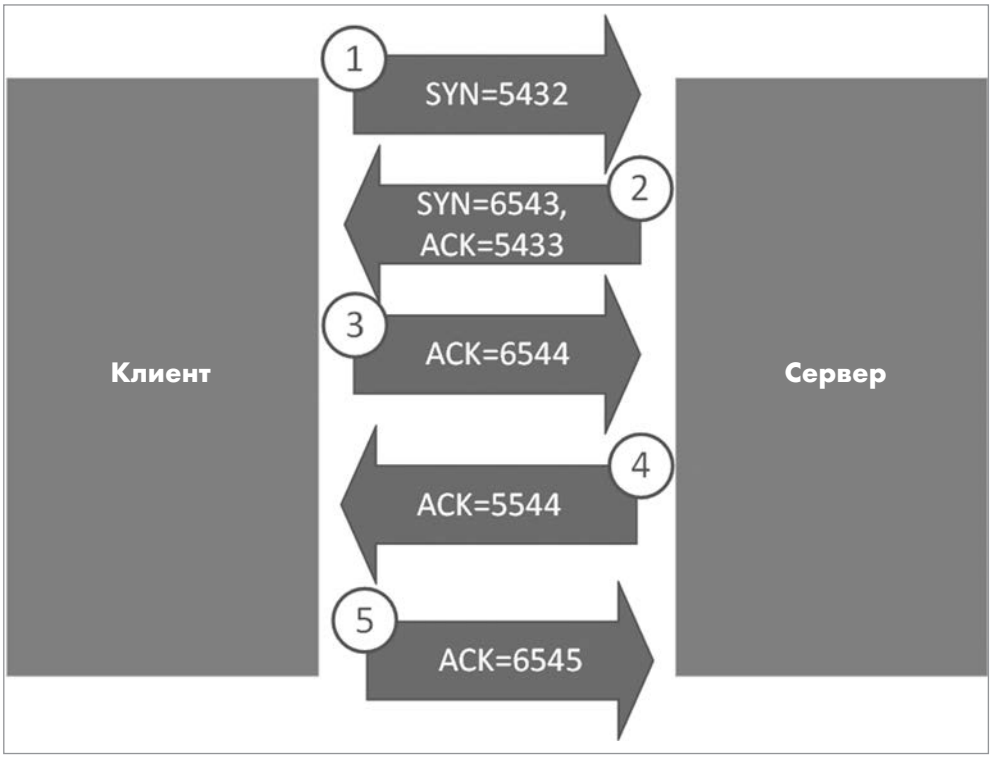

Рис. 3.5. Трехэтапное квитирование TCP с установленным сеансом TCP

Квитирование работает так:

1. Первый пакет поступает от клиента с динамического порта на фиксированный (как правило) порт сервера. В пакете установлен бит **SYN** (синхронизация), и ему случайно назначен порядковый номер **SEQ**, в данном случае **5432**.
2. В ответном пакете от сервера установлен бит **ACK** (подтверждение) со значением на единицу больше, чем **SYN** входящего пакета (в данном случае **5433**), а также бит **SYN** со своим собственным случайным значением, в данном случае **6543**. Помимо информации о квитировании, в этом пакете (и во всех последующих) могут содержаться данные.
3. Третий пакет — это подтверждение (ACK) первого **SYN** сервера, его номер на единицу больше **SYN** второго пакета (в данном случае **6544**).

4. Все последующие пакеты представляют собой пакеты **ACK**, которые отправляются от клиента к серверу или обратно, поэтому у каждого пакета есть уникальный порядковый номер и направление.

Формально пакет из пункта 2 может быть двумя отдельными пакетами, но обычно они объединяются в один пакет.

Штатное завершение обмена данными работает точно так же. Сторона, которая завершает обмен, передает **FIN**, другая в ответ передает **FIN-ACK**, на него поступает **ACK** от первой стороны, и все готово.

Менее элегантное завершение сеанса часто инициируется пакетом **RST** (reset, сброс): после передачи **RST** все заканчивается и другая сторона не должна отправлять на него ответ.

Мы будем использовать только что изученный материал позже в этой главе, а также на протяжении всей книги. Поэтому если у вас пока не сложилось четкого понимания, перечитайте раздел еще раз, обращая особое внимание на предыдущий рисунок, пока все не станет ясно.

Теперь, когда мы представляем себе, как порты TCP и UDP соединяются друг с другом и в каких случаях приложения могут использовать тот или иной порт, давайте посмотрим, как приложения вашего узла прослушивают различные порты.

Сканирование локальных портов и их связь с запущенными службами

Многие ключевые шаги по устранению неполадок в сети выполняются на одном или на другом конце канала связи, а именно на клиентском или на серверном узле. Например, если веб-сервер недоступен, то в первую очередь полезно проверить, запущен ли его процесс и прослушивает ли он соответствующие порты в ожидании клиентских запросов.

Чтобы оценить состояние текущих сеансов связи и служб на локальном узле, традиционно применяется команда `netstat`. Список всех прослушиваемых портов и подключений можно вывести с помощью следующих параметров:

t	Порты TCP
u	Порты UDP
a	Все порты (прослушивающие и нет)
n	Не выполнять разрешение (resolution) DNS для задействованных IP-адресов. Без этого параметра вывод может значительно замедлиться, потому что команда попытается разрешить каждый IP-адрес в списке.