Сетевые сервисы

Мы научимся предоставлять некоторые сетевые сервисы, такие как DHCP-сервер для раздачи IP-адресов клиентам, SSH-туннели и socks- прокси, а также коснемся механизма агрегации каналов.

Оглавление

[DHCP-сервер](#_rvvlzeaaa9v2)

[Принцип работы DHCP](#_q8wuok6l020w)

[Обновление адреса (RENEWING)](#_z5ijprylq7lv)

[Обновление конфигурации (REBINDING)](#_jpldwm5g0quz)

[Освобождение адреса](#_2w4nausquh8z)

[Установка и настройка DHCP](#_3qqqojbfq591)

[NIC Bonding](#_axvccsbwcfsv)

[SSH](#_ed7tejht6d85)

[Passwordless SSH](#_u40r1iqmtjsn)

[SOCKS](#_rpjadtltg796)

[SSH-туннелирование](#_v0v0zlvoe404)

[Удаленное копирование файлов](#_w7wklukavdnn)

[Домашнее задание](#_4i7ojhp)

[Дополнительные материалы](#_2xcytpi)

[Используемая литература](#_i520njibh5x9)

# 

# DHCP-сервер

Dynamic Host Configuration Protocol (DHCP) — клиент-серверный протокол динамической настройки хостов. Чаще всего используется для автоматического предоставления IP-адреса, а также иных настроек конечному хосту. В основном это сетевые настройки, но на самом деле возможности DHCP гораздо шире: при помощи этого протокола можно загружать операционную систему по сети или передавать конфигурационный файл для устройства. Мы будем рассматривать DHCP-сервер в классическом варианте — централизованное и динамическое управление сетевыми настройками.

В этом случае при помощи DHCP решаются две основные проблемы:

1. Автоматизация. Присутствие в сети DHCP-сервера позволяет не производить настройки на каждом новом клиенте вручную. Достаточно лишь настроить DHCP-сервер и дальнейшая настройка IP-адресов и прочих сетевых параметров будет производиться автоматически.
2. Централизация управления. DHCP-сервер производит контроль за выданными адресами, что позволяет избежать дублирования IP-адресов, а также высвобождать неиспользуемые. В случае смены адресного пространства DHCP-сервер позволяет сделать это в несколько несложных действий.

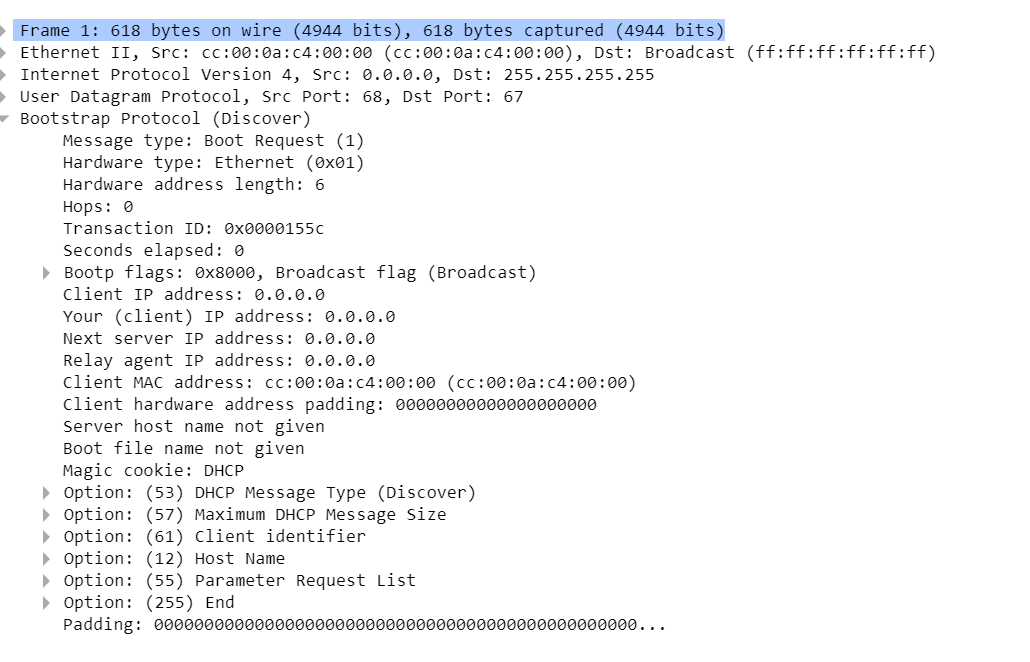
Протокол DHCP — открытый стандарт, описан в RFC 2131 <https://tools.ietf.org/html/rfc2131>

### Принцип работы DHCP

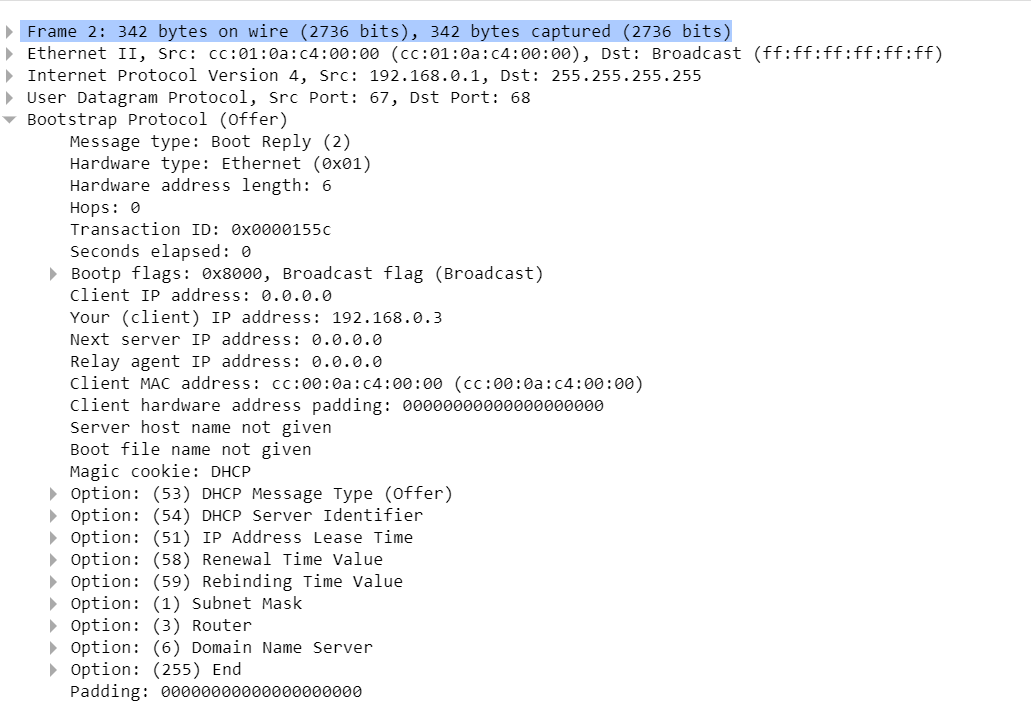
Протокол DHCP работает по схеме «клиент-сервер» и использует для этого широковещательные (броадкаст) сообщения. Сам процесс взаимодействия происходит в 4 сообщения и описывается схемой DORA (Discover-Offer-Request-Acknowledge). Для своей работы DHCP использует протокол UDP. Сообщения от клиента к серверу передаются по порту 67 UDP, а сообщения от сервера клиенту — на порт UDP 68:



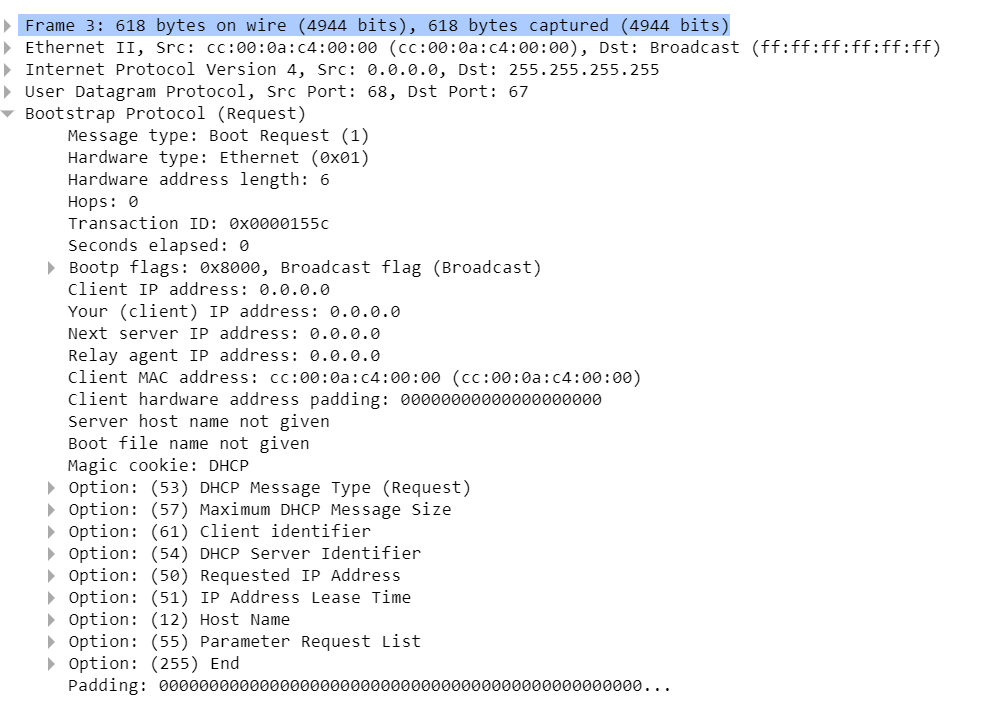
* Discover (Обнаружение) — DHCP-клиент подключается к сети и приступает к инициализации (состояние INIT). Для этого ему необходимо найти в сети подходящий DHCP-сервер. С этой целью отправляется широковещательный UDP-запрос DHCPDISCOVER на широковещательный адрес получателя (destination address) 255.255.255.255. В качестве адреса отправителя (source address) клиент указывает 0.0.0.0, так как в этот момент адреса у него на интерфейсе нет.. Также в запросе клиент указывает свой MAC-адрес, хостнейм и некоторые другие параметры. Так как запрос является широковещательным, его получают все хосты, находящиеся с клиентом в одном широковещательном домене (броадкаст-домене). Пример пакета изображен на рисунке:



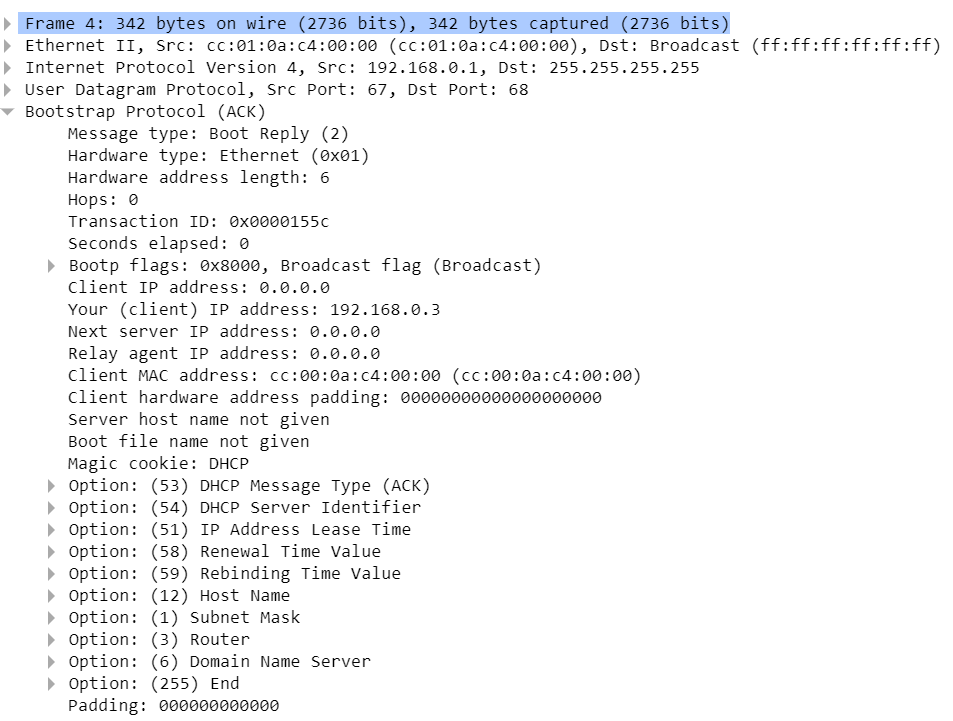
* Offer (Предложение) — ответ от DHCP-сервера на запрос DHCPDISCOVER. Сервер, проанализировав пришедший DHCPDISCOVER-пакет, выбирает подходящую конфигурацию сети для хоста и отправляют ее в ответном сообщении DHCPOFFER. На втором уровне модели OSI ответ может быть отправлен на конкретный MAC-адрес получателя, полученный в DHCPDISCOVER, хотя, зачастую используется широковещательный MAC-адрес (ff:ff:ff:ff:ff:ff). На третьем уровне модели OSI всегда используется широковещательный адрес (255.255.255.255), так как в этот момент у хоста нет адреса на интерфейсе. Если в сети находятся несколько DHCP-серверов, клиент получает несколько ответов DHCPOFFER и выбирает из них один, как правило, полученный первым.



* Request (Запрос) — клиент, получив ответ сервера, проверяет, подходят ли ему присланные настройки и отвечает серверу сообщением DHCPREQUEST, в котором уже официально запрашивает у сервера предоставленные настройки. В сообщении DHCPREQUEST содержится та же информация, что и в DHCPDISCOVER, а также IP-адрес выбранного DHCP-сервера. DHCPREQUEST отправляется на широковещательный адрес и те DHCP-сервера, чей адрес отсутствует в сообщении, понимают, что их предложение отвергнуто.



* Acknowledge (Подтверждение) — DHCP-сервер, адрес которого указан в DHCPREQUEST, получает сообщение и понимает, что его выбрал клиент в качестве сервера. Он фиксирует привязку для клиента и отвечает сообщением DHCPACK, подтверждая выданные клиенту настройки. DHCPACK все так же отправляется на широковещательный адрес. Клиент получает сообщение DHCPACK, проверяет настройки и применяет конфигурацию (состояние BOUND), которая была получена в сообщении DHCPOFFER.

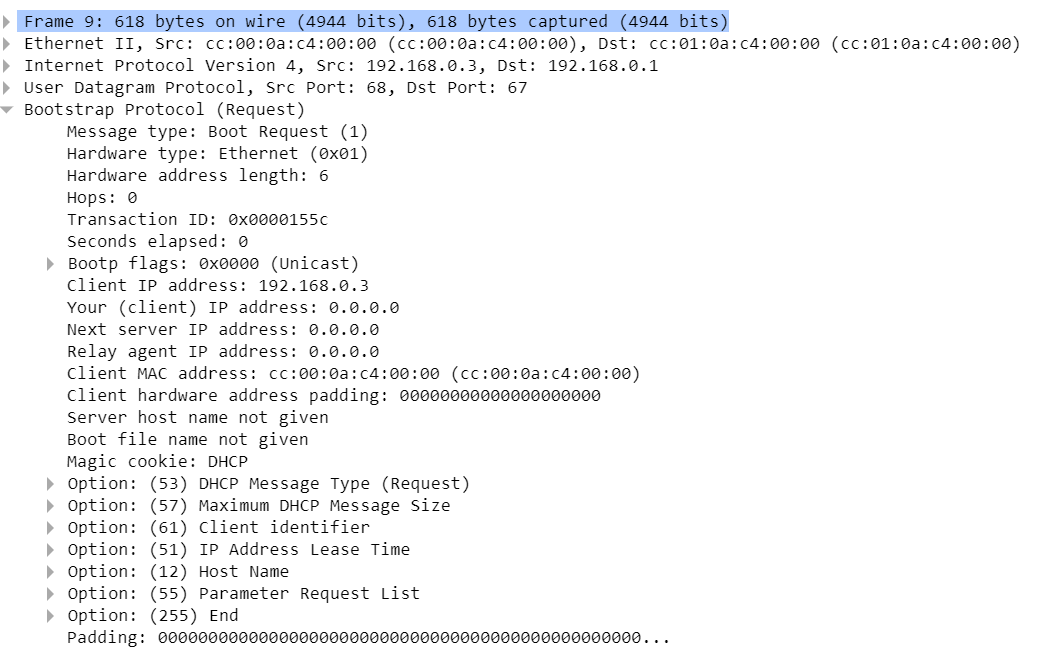


Если клиента что-либо не устроит в предложенном ему IP-адресе (например, клиент определяет, что такой адрес в сети используется), то клиент отправит серверу сообщение DHCPDECLINE и начнёт процедуру инициализации заново. Сообщение DHCPDECLINE отправляется бродкастом, поскольку клиент отвергает предложенный ему IP-адрес. DHCP-сервер, получив сообщение DHCPDECLINE, должен пометить IP-адрес как недоступный.

Если клиент хочет продолжить пользоваться IP-адресом, выданным ему DHCP-сервером, процесс происходит за меньшее количество сообщений. В этом случае клиент передает сообщение DHCPREQUEST, указывая в сообщении свой адрес. DHCP-cервер, получивший запрос, проверяет корректность сети и адреса и в случае успеха посылает клиенту подтверждение DHCPACK. Клиент получает подтверждение и применяет настройки.

Если DHCP-сервер обнаруживает, что клиент находится в неподходящей сети, он отвечает отказом DHCPNACK. Если сеть корректна, проверяется наличие записи для этого клиента и доступность запрошенного адреса. Если адрес по какой-либо причине не подходит (например занят), то сервер отвечает отказом DHCPNACK. Получив отказ, клиент больше не может пользоваться сохраненным сетевым адресом и должен запросить новый, начав полную процедуру инициализации.

Если же на сервере нет записи клиента, он считает, что адрес был выдан другим DHCP-сервером и оставляет запрос без ответа. Такое поведение позволяет нескольким независимым DHCP-серверам находиться в одной сети.



[Рассмотрите взаимодействие подробнее](https://www.cloudshark.org/captures/0009d5398f37).

IP-адрес выдается клиенту на определенное время, которое называется временем аренды (lease time). Время аренды зависит от настроек сервера и может варьироваться от нескольких минут до недель и даже месяцев. По прошествии половины срока клиент пробует обновить аренду. Если это не удаётся сделать сразу, клиент будет пытаться сделать это снова вплоть до окончания срока. Если все попытки окажутся неудачными, по окончании срока клиент будет искать другой DHCP-сервер.

В процессе обновления клиент проходит два состояния — обновление адреса (RENEWING) и обновление конфигурации (REBINDING). Первое состояние наступает примерно на половине срока аренды адреса (T1), второе — по истечении 87.5% (или 7/8) полного срока аренды (T2). Для предотвращения синхронизации разных клиентов при расчете значений T1 и T2 к ним добавляется случайное отклонение.

### Обновление адреса (RENEWING)

Так как IP-адрес выдается клиенту в аренду на некоторое (настраиваемое) время, то в какой-то момент времени, клиент захочет эту аренду продлить. Для этого клиент посылает запрос DHCPREQUEST, но не широковещательный, а адресованный своему DHCP-серверу. Сервер получает запрос, после чего возможно два варианта:

1. Сервер соглашается продлить аренду. Для этого он отправляет клиенту сообщение DHCPACK с указанием нового срока аренды и тех параметров, которые могли измениться с момента создания или последнего продления аренды.
2. Сервер по какой-то причине не хочет продлевать аренду. В таком случае он отправляет клиенту сообщение об отказе DHCPNACK.

В зависимости от полученного ответа клиент:

1. В случае положительного ответа DHCPACK отмечает новый срок истечения аренды и все измененные параметры, полученные от сервера, сбрасывает таймеры T1 и T2 и переходит в нормальное (BOUND) состояние.
2. Получив отрицательный ответ, DHCPNACK немедленно переходит в состояние инициализации (INIT) и начинает процедуру получения аренды заново.

### Обновление конфигурации (REBINDING)

Если клиент не получает сразу ответ от сервера на запрос обновления аренды, то ожидает его в течение времени (T2 — t)/2 сек (но не меньше 60 сек), где t — время отправки последнего сообщения DHCPREQUEST, затем отправляет сообщение повторно. Пока сервер не ответит, клиент остается в состоянии RENEWING и регулярно шлет запрос DHCPREQUEST на сервер. В течение этого времени он сохраняет свой текущий адрес и продолжает нормально работать.

Если ответ от сервера не поступил к моменту T2, клиент переходит в состояние REBINDING и передает широковещательное сообщение DHCPREQUEST со своим текущим адресом. Срок повтора запросов DHCPREQUEST рассчитывается аналогично предыдущему случаю, только вместо T2 используется полное время окончания срока аренды.

Если срок аренды завершается до получения ответа от сервера, клиент должен прекратить все сетевые операции и перейти в состояние инициализации (INIT). Если DHCP-сервер все-таки ответит после завершения аренды, клиент может возобновить работу с прежним адресом.

### Освобождение адреса

Клиент может явно отказаться от аренды сетевого адреса, передав серверу сообщение DHCPRELEASE. При получении этого сообщения сервер помечает адрес как свободный, но сохраняет запись с параметрами клиента в базе на случай, если клиент захочет использовать адрес повторно. Стоит уточнить, что клиент не освобождает аренду при обычном выключении, все настройки сохраняются локально. Клиент передает DHCPRELEASE только при явной необходимости отказаться от аренды, например при перемещении в другую подсеть. Также освободить аренду можно вручную, например с помощью команды ipconfig /release.

По умолчанию сообщения DHCP ограничены текущей подсетью. Для работы DHCP использует широковещание (broadcast), а маршрутизаторы не пропускают широковещательный трафик за пределы домена. Чтобы обойти это ограничение, маршрутизаторы должны уметь перенаправлять полученные броадкаст-DHCP-сообщения в unicast-пакеты и отправлять их напрямую на DHCP-сервер. Такая функция называется dhcp-relay/dhcp-helper/bootp-helper.

## Установка и настройка DHCP

1. Устанавливаем нужный пакет:

|  |
| --- |
| [root@dhcp-server ~]# **yum install dhcp**  Dependencies Resolved  ==================================================================================================================  Package Arch Version Repository Size  ==================================================================================================================  Installing:  dhcp x86\_64 12:4.2.5-77.el7.centos base 514 k  Installing for dependencies:  bind-export-libs x86\_64 32:9.11.4-9.P2.el7 base 1.1 M  Updating for dependencies:  dhclient x86\_64 12:4.2.5-77.el7.centos base 285 k  dhcp-common x86\_64 12:4.2.5-77.el7.centos base 176 k  dhcp-libs x86\_64 12:4.2.5-77.el7.centos base 133 k  Transaction Summary |

1. Копируем пример настройки файла в /etc/dhcp/dhcpd.conf:

|  |
| --- |
| [root@dhcp-server ~]# **cp /usr/share/doc/dhcp-4.2.5/dhcpd.conf.example /etc/dhcp/dhcpd.conf**  cp: overwrite ‘/etc/dhcp/dhcpd.conf’? y  [root@dhcp-server ~]# |

1. Закомментируем все строчки конфига, чтобы потом включить только нужные нам опции:

|  |
| --- |
| [root@dhcp-server ~]# **sed s/^/#/ /etc/dhcp/dhcpd.conf | tee /etc/dhcp/dhcpd.conf** |

1. Воспользовавшись примером, настроим DHCP-подсеть для 192.168.200.0/24. В нашем случае диапазон выдаваемых адресов ограничен 192.168.200.100-192.168.200.199:

|  |
| --- |
| [root@dhcp-server ~]# **vim /etc/dhcp/dhcpd.conf**  subnet 192.168.200.0 netmask 255.255.255.0 {  range 192.168.200.100 192.168.200.199;  option domain-name-servers 8.8.8.8;  option routers 192.168.200.1;  option broadcast-address 192.168.200.255;  default-lease-time 600;  max-lease-time 7200;  } |

1. Включаем dhcpd при загрузке системы, а также запускаем службу:

|  |
| --- |
| [root@dhcp-server ~]# **systemctl enable dhcpd**  Created symlink from /etc/systemd/system/multi-user.target.wants/dhcpd.service to /usr/lib/systemd/system/dhcpd.service.  [root@dhcp-server ~]# **systemctl start dhcpd**  [root@dhcp-server ~]# **systemctl status dhcpd**  ● dhcpd.service - DHCPv4 Server Daemon  Loaded: loaded (/usr/lib/systemd/system/dhcpd.service; enabled; vendor preset: disabled)  Active: active (running) since Sun 2019-09-22 17:04:20 MSK; 2min 36s ago  Docs: man:dhcpd(8)  man:dhcpd.conf(5)  Main PID: 2186 (dhcpd)  Status: "Dispatching packets..."  CGroup: /system.slice/dhcpd.service  └─2186 /usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user dhcpd -group dhcpd --no-pid  Sep 22 17:04:20 dhcp-server dhcpd[2186]: Sending on LPF/ens37/00:0c:29:2f:32:39/192.168.200.0/24 |

1. Проверяем получение IP-адреса клиентом:

|  |
| --- |
| [root@dhcp-client ~]# **ip a s ens37**  3: ens37: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000  link/ether 00:0c:29:4b:c3:62 brd ff:ff:ff:ff:ff:ff  inet 192.168.200.100/24 brd 192.168.200.255 scope global noprefixroute dynamic ens37  valid\_lft 541sec preferred\_lft 541sec  inet6 fe80::2a12:de2d:f218:499e/64 scope link noprefixroute  valid\_lft forever preferred\_lft forever |

Как видно, клиент получил IP-адрес в соответствующей подсети на ближайшие 541 секунду.

# NIC Bonding

В некоторых случаях серверы подключаются к сети более чем одним интерфейсом. В таких ситуациях для предотвращения закольцовок на втором уровне модели OSI, а также для увеличения скорости сходимости сети, требуется эти несколько физических интерфейсов объединить в один логический. В терминах различных операционных систем такая конфигурация может называться по-разному: channel bonding, Ethernet bonding, port trunking, channel teaming, NIC teaming, link aggregation. В CentOS 7 в состав дистрибутива включены 2 различных драйвера для этого — bonding и teaming. Последний — более современный, поддерживает больше функций, например балансировку для LACP, поддержку D-Bus и Unix Domain Sockets и так далее.



Для настройки teaming вам необходимо как минимум 2 сетевых интерфейса на сервере, подключенных в один коммутатор, либо в 2 разных коммутатора, умеющих stack либо multi-chassis etherchannel (MEC). Мы будем использовать 2 сервера, подключенные друг к другу.

1. Устанавливаем необходимый пакет на обоих серверах:

|  |
| --- |
| [root@team-01 ~]# **yum install teamd**  Dependencies Resolved  ==================================================================================================================  Package Arch Version Repository Size  ==================================================================================================================  Updating:  teamd x86\_64 1.27-9.el7 base 113 k  Updating for dependencies:  libteam x86\_64 1.27-9.el7 base 49 k  Transaction Summary  ================================================================================================================== |

1. Настраиваем интерфейсы. Мы будем использовать интерфейсы ens37 и ens38:

|  |
| --- |
| [root@team-01 ~]# **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 1500 qdisc pfifo\_fast state UP mode DEFAULT group default qlen 1000  link/ether 00:0c:29:2f:32:2f brd ff:ff:ff:ff:ff:ff  3: ens37: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP mode DEFAULT group default qlen 1000  link/ether 00:0c:29:2f:32:39 brd ff:ff:ff:ff:ff:ff  4: ens38: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP mode DEFAULT group default qlen 1000  link/ether 00:0c:29:2f:32:43 brd ff:ff:ff:ff:ff:ff |

1. Создаем логический интерфейс team0, который в дальнейшем будет представлять агрегированные физические порты. Повесим на этот интерфейс IP-адрес.

|  |
| --- |
| [root@team-01 network-scripts]# **pwd**  /etc/sysconfig/network-scripts  [root@team-01 network-scripts]# **cat ifcfg-team0**  DEVICE=nm-team  DEVICETYPE=Team  BOOTPROTO=static  DEFROUTE=NO  PEERDNS=yes  PEERROUTES=yes  IPV4\_FAILURE\_FATAL=no  IPV6INIT=yes  IPV6\_AUTOCONF=yes  IPV6\_DEFROUTE=yes  IPV6\_PEERDNS=yes  IPV6\_PEERROUTES=yes  IPV6\_FAILURE\_FATAL=no  NAME=team0  UUID=c794ce57-2879-4426-9632-50cf05f8d5b5  ONBOOT=yes  IPADDR=192.168.200.1  NETMASK=255.255.255.0 |

1. Указываем интерфейсам ens37 и ens38, что они — члены team0-интерфейса:

|  |
| --- |
| [root@team-01 network-scripts]# **cat ifcfg-team-slave-ens37**  NAME=team-slave-ens37  UUID=9b5d1511-43ee-4184-b20d-540c2820bb6a  DEVICE=ens37  ONBOOT=yes  TEAM\_MASTER=c794ce57-2879-4426-9632-50cf05f8d5b5  DEVICETYPE=TeamPort  [root@team-01 network-scripts]# **cat ifcfg-team-slave-ens38**  NAME=team-slave-ens38  UUID=9b5d1511-43ee-4184-b20d-540c2820bb6b  DEVICE=ens38  ONBOOT=yes  TEAM\_MASTER=c794ce57-2879-4426-9632-50cf05f8d5b5  DEVICETYPE=TeamPort |

1. Аналогичные настройки, но с другим IP-адресом делаем на втором сервере и перезапускаем службу network:

|  |
| --- |
| [root@team-01 network-scripts]# **systemctl restart network**  [root@team-01 network-scripts]# **ip a s**  1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  inet 127.0.0.1/8 scope host lo  valid\_lft forever preferred\_lft forever  inet6 ::1/128 scope host  valid\_lft forever preferred\_lft forever  2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000  link/ether 00:0c:29:2f:32:2f brd ff:ff:ff:ff:ff:ff  inet 192.168.1.41/24 brd 192.168.1.255 scope global noprefixroute dynamic ens33  valid\_lft 25198sec preferred\_lft 25198sec  inet6 fe80::fdc8:d699:3e8d:5c3b/64 scope link tentative noprefixroute dadfailed  valid\_lft forever preferred\_lft forever  inet6 fe80::d473:6bc3:2555:aa17/64 scope link noprefixroute  valid\_lft forever preferred\_lft forever  3: ens37: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast master nm-team state UP group default qlen 1000  link/ether 00:0c:29:2f:32:39 brd ff:ff:ff:ff:ff:ff  4: ens38: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast master nm-team state UP group default qlen 1000  link/ether 00:0c:29:2f:32:39 brd ff:ff:ff:ff:ff:ff  5: nm-team: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000  link/ether 00:0c:29:2f:32:39 brd ff:ff:ff:ff:ff:ff  inet 192.168.200.1/24 brd 192.168.200.255 scope global noprefixroute nm-team  valid\_lft forever preferred\_lft forever  inet6 fe80::20c:29ff:fe2f:3239/64 scope link noprefixroute  valid\_lft forever preferred\_lft forever |

1. Появился интерфейс с именем nm-team и выбранным нами IP-адресом. Проверяем связность со вторым хостом и видим, что пинг проходим, а ARP выучен с нашего team-интерфейса:

|  |
| --- |
| [root@team-01 network-scripts]# **ping 192.168.200.2**  PING 192.168.200.2 (192.168.200.2) 56(84) bytes of data.  64 bytes from 192.168.200.2: icmp\_seq=1 ttl=64 time=0.297 ms  --- 192.168.200.2 ping statistics ---  1 packets transmitted, 1 received, 0% packet loss, time 0ms  [root@team-01 network-scripts]# ip nei show dev nm-team  192.168.200.2 lladdr 00:0c:29:4b:c3:62 STALE |

Так как физически интерфейсов два, а логически — один, сервер должен выбирать, в какой из физических интерфейсов отправлять данные. Делать он это может разными способами в зависимости от настроек с обеих сторон team-интерфейса, а также вашего желания.

Настройка по умолчанию — round-robin, её можно изменить, если вы этого хотите.

|  |
| --- |
| [root@team-01 ~]# **teamdctl nm-team state**  setup:  runner: roundrobin  ports:  ens37  link watches:  link summary: up  instance[link\_watch\_0]:  name: ethtool  link: up  down count: 0  ens38  link watches:  link summary: up  instance[link\_watch\_0]:  name: ethtool  link: up  down count: 0 |

Выбрать вы можете из:

* **Roundrobin** — режим по умолчанию. По очереди посылает пакеты по всем интерфейсам.
* **Broadcast** отправляет пакеты из всех интерфейсов.
* **Activebackup** — один интерфейс активный, второй — запасной. В случае проблем с первым трафик переключается на запасной интерфейс.
* **Loadbalance** использует механизм балансировки для выбора исходящего интерфейса.

Изменить настройки вы можете, дописав новый режим в файл /etc/sysconfig/network-scripts/ifcfg-team0.

|  |
| --- |
| [root@team-01 ~]# **cat /etc/sysconfig/network-scripts/ifcfg-team0**  DEVICE=nm-team  DEVICETYPE=Team  BOOTPROTO=static  DEFROUTE=NO  PEERDNS=yes  PEERROUTES=yes  IPV4\_FAILURE\_FATAL=no  IPV6INIT=yes  IPV6\_AUTOCONF=yes  IPV6\_DEFROUTE=yes  IPV6\_PEERDNS=yes  IPV6\_PEERROUTES=yes  IPV6\_FAILURE\_FATAL=no  NAME=team0  UUID=c794ce57-2879-4426-9632-50cf05f8d5b5  ONBOOT=yes  IPADDR=192.168.200.1  NETMASK=255.255.255.0  TEAM\_CONFIG='{"runner": {"name": "activebackup"}, "link\_watch": {"name": "ethtool"}}'  [root@team-01 ~]# **systemctl restart network**  [root@team-01 ~]# **teamdctl nm-team state**  setup:  runner: activebackup  ports:  ens37  link watches:  link summary: up  instance[link\_watch\_0]:  name: ethtool  link: up  down count: 0  ens38  link watches:  link summary: up  instance[link\_watch\_0]:  name: ethtool  link: up  down count: 0  runner:  active port: ens37 |

Больше информации о возможных режимах и проверках состояния канала:

|  |
| --- |
| man 5 nmcli-examples  teamnl nm-team options |

# SSH

Как правило, удаленное администрирование выполняется с применением SSH. Протокол SSH широко используется, стандартно работает, используя TCP-порт 22, и включает такие сервисы, как scp и sftp.

|  |
| --- |
| [root@iscsi-target ~]# **ss -tulpan | grep 22**  tcp LISTEN 0 128 \*:22 \*:\* users:(("SSHd",pid=1295,fd=3)) |

Сервис стандартно стартует на 0.0.0.0, что разрешает подключение к любому IP-адресу сервера. Проверить настройки и текущее состояние можно так:

|  |
| --- |
| [root@iscsi-target ~]# **systemctl status SSHd**  ● SSHd.service - OpenSSH server daemon  Loaded: loaded (/usr/lib/systemd/system/SSHd.service; enabled; vendor preset: enabled)  Active: active (running) since Sat 2019-10-19 13:55:15 MSK; 1min 58s ago  Docs: man:SSHd(8)  man:SSHd\_config(5)  Main PID: 1295 (SSHd)  CGroup: /system.slice/SSHd.service  └─1295 /usr/sbin/SSHd -D  Oct 19 13:55:15 iscsi-target systemd[1]: Starting OpenSSH server daemon...  Oct 19 13:55:15 iscsi-target SSHd[1295]: Server listening on 0.0.0.0 port 22.  Oct 19 13:55:15 iscsi-target SSHd[1295]: Server listening on :: port 22.  Oct 19 13:55:15 iscsi-target systemd[1]: Started OpenSSH server daemon.  Oct 19 13:55:38 iscsi-target SSHd[11250]: Accepted password for root from 192.168.1.40 port 49496 SSH2 |

Настройки демона находятся в файле **/etc/SSH/SSHd\_config** и по умолчанию выглядят так:

|  |
| --- |
| [root@iscsi-target ~]# **cat /etc/SSH/SSHd\_config**  # $OpenBSD: SSHd\_config,v 1.100 2016/08/15 12:32:04 naddy Exp $  # This is the SSHd server system-wide configuration file. See  # SSHd\_config(5) for more information.  # This SSHd was compiled with PATH=/usr/local/bin:/usr/bin  # The strategy used for options in the default SSHd\_config shipped with  # OpenSSH is to specify options with their default value where  # possible, but leave them commented. Uncommented options override the  # default value.  # If you want to change the port on a SELinux system, you have to tell  # SELinux about this change.  # semanage port -a -t SSH\_port\_t -p tcp #PORTNUMBER  #  #Port 22  #AddressFamily any  #ListenAddress 0.0.0.0  #ListenAddress ::  HostKey /etc/SSH/SSH\_host\_rsa\_key  #HostKey /etc/SSH/SSH\_host\_dsa\_key  HostKey /etc/SSH/SSH\_host\_ecdsa\_key  HostKey /etc/SSH/SSH\_host\_ed25519\_key  # Ciphers and keying  #RekeyLimit default none  # Logging  #SyslogFacility AUTH  SyslogFacility AUTHPRIV  #LogLevel INFO  # Authentication:  #LoginGraceTime 2m  #PermitRootLogin yes  #StrictModes yes  #MaxAuthTries 6  #MaxSessions 10  #PubkeyAuthentication yes  # The default is to check both .SSH/authorized\_keys and .SSH/authorized\_keys2  # but this is overridden so installations will only check .SSH/authorized\_keys  AuthorizedKeysFile .SSH/authorized\_keys  #AuthorizedPrincipalsFile none  #AuthorizedKeysCommand none  #AuthorizedKeysCommandUser nobody  # For this to work you will also need host keys in /etc/SSH/SSH\_known\_hosts  #HostbasedAuthentication no  # Change to yes if you don't trust ~/.SSH/known\_hosts for  # HostbasedAuthentication  #IgnoreUserKnownHosts no  # Don't read the user's ~/.rhosts and ~/.shosts files  #IgnoreRhosts yes  # To disable tunneled clear text passwords, change to no here!  #PasswordAuthentication yes  #PermitEmptyPasswords no  PasswordAuthentication yes  # Change to no to disable s/key passwords  #ChallengeResponseAuthentication yes  ChallengeResponseAuthentication no  # Kerberos options  #KerberosAuthentication no  #KerberosOrLocalPasswd yes  #KerberosTicketCleanup yes  #KerberosGetAFSToken no  #KerberosUseKuserok yes  # GSSAPI options  GSSAPIAuthentication yes  GSSAPICleanupCredentials no  #GSSAPIStrictAcceptorCheck yes  #GSSAPIKeyExchange no  #GSSAPIEnablek5users no  # Set this to 'yes' to enable PAM authentication, account processing,  # and session processing. If this is enabled, PAM authentication will  # be allowed through the ChallengeResponseAuthentication and  # PasswordAuthentication. Depending on your PAM configuration,  # PAM authentication via ChallengeResponseAuthentication may bypass  # the setting of "PermitRootLogin without-password".  # If you just want the PAM account and session checks to run without  # PAM authentication, then enable this but set PasswordAuthentication  # and ChallengeResponseAuthentication to 'no'.  # WARNING: 'UsePAM no' is not supported in Red Hat Enterprise Linux and may cause several  # problems.  UsePAM yes  #AllowAgentForwarding yes  #AllowTcpForwarding yes  #GatewayPorts no  X11Forwarding yes  #X11DisplayOffset 10  #X11UseLocalhost yes  #PermitTTY yes  #PrintMotd yes  #PrintLastLog yes  #TCPKeepAlive yes  #UseLogin no  #UsePrivilegeSeparation sandbox  #PermitUserEnvironment no  #Compression delayed  #ClientAliveInterval 0  #ClientAliveCountMax 3  #ShowPatchLevel no  #UseDNS yes  #PidFile /var/run/SSHd.pid  #MaxStartups 10:30:100  #PermitTunnel no  #ChrootDirectory none  #VersionAddendum none  # no default banner path  #Banner none  # Accept locale-related environment variables  AcceptEnv LANG LC\_CTYPE LC\_NUMERIC LC\_TIME LC\_COLLATE LC\_MONETARY LC\_MESSAGES  AcceptEnv LC\_PAPER LC\_NAME LC\_ADDRESS LC\_TELEPHONE LC\_MEASUREMENT  AcceptEnv LC\_IDENTIFICATION LC\_ALL LANGUAGE  AcceptEnv XMODIFIERS  # override default of no subsystems  Subsystem sftp /usr/libexec/openSSH/sftp-server  # Example of overriding settings on a per-user basis  #Match User anoncvs  # X11Forwarding no  # AllowTcpForwarding no  # PermitTTY no  # ForceCommand cvs server |

* Port 22 имеет смысл заменить на нестандартный, если сервер выставлен в интернет. Это не повышает безопасность, но уменьшит количество попыток зайти к вам по SSH ботами, которые настроены именно на порт 22. То есть вы просто будете в логах видеть меньше попыток залогиниться извне.
* ListenAdress — можно указать только один из доступных адресов для прослушивания.
* PermitRootLogin — имеет смысл заменить на no (убедитесь, что есть пользователи в группе sudo). Это запретит доступ по SSH пользователю root.
* PasswordAuthentication — по умолчанию yes, но лучше поставить no и использовать аутентификацию по ключам.
* X11Forwarding — возможность запуска X11-приложений. Если такой необходимости нет, лучше поставить no.

### Passwordless SSH

Кроме аутентификации пользователя по паролю, SSH позволяет делать это при помощи асимметричной криптографии с парой публичного и приватного ключей.

Можно воспользоваться утилитой SSH-keygen, указав алгоритм ключа. По умолчанию это будет RSA, но, если все ваши сервера поддерживают криптографию на эллиптических кривых, для минимизации нагрузки следует использовать ed25519 или ecdsa.

|  |
| --- |
| [root@iscsi-target ~]# **SSH-keygen -t ed25519**  Generating public/private ed25519 key pair.  Enter file in which to save the key (/root/.SSH/id\_ed25519):  Created directory '/root/.SSH'.  Enter passphrase (empty for no passphrase):  Enter same passphrase again:  Your identification has been saved in /root/.SSH/id\_ed25519.  Your public key has been saved in /root/.SSH/id\_ed25519.pub.  The key fingerprint is:  SHA256:XBUR8Cc7RV2kttLulb1etHqFJ9gSH2oXXsiEUlJc6kY root@iscsi-target  The key's randomart image is:  +--[ED25519 256]--+  | o=B\*oo+|  | .+oo...|  | ..Eo=. |  | . . o.B=..|  | S \*Bo=.|  | .=+B.\*|  | . o.==|  | ..oo|  | .+o |  +----[SHA256]-----+ |

Потребуется ввести пароль, но не к SSH или пользователю, а к приватному ключу. Каждый раз, когда планируется использовать приватную часть ключа, вам придется вводить этот пароль; если вы его забудете, ключ станет бесполезным, использовать его больше не получится. Если ключ применяется для скриптов, пароль использовать не следует.

В директории .SSH внутри вашего домашнего каталога будут созданы файлы id\_<тип\_ключа> и id\_<тип\_ключа>.pub. Первый (id\_<тип\_ключа>) остаётся на вашей машине и никогда не должен покидать её — это приватный ключ. Второй (id\_<тип\_ключа>.pub) — публичный ключ, его можно загрузить на сервер или опубликовать в интернете. Ценности без приватной части он не имеет. Машины, на которые вы загрузили публичный ключ, позволят подключиться только тем клиентам, у которых есть приватная часть ключа. Загрузите публичный ключ id\_<тип\_ключа>.pub в домашнюю директорию на удаленной Linux-машине, которую будете администрировать.

В нашем случае, так как выбранный алгоритм — ed25519, приватная и публичные части ключа выглядят и называются так:

|  |
| --- |
| [root@iscsi-target ~]# **ls ~/.SSH/**  id\_ed25519 id\_ed25519.pub |

В случае использования RSA:

|  |
| --- |
| [root@iscsi-target ~]# **ls ~/.SSH/**  id\_rsa id\_rsa.pub |

Понять, какая часть приватная, а какая — публичная, можно исходя из контента самого файла. В приватном ключе будет написано -----BEGIN PRIVATE KEY----- и -----END PRIVATE KEY-----.

|  |
| --- |
| [root@iscsi-target ~]# **cat ~/.SSH/id\_ed25519**  -----BEGIN OPENSSH PRIVATE KEY-----  b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW  QyNTUxOQAAACCFan6B7XSQ/xFJThqf7HPuJ9grMpUo277VSyZGuqF/fAAAAJjz6qBk8+qg  ZAAAAAtzc2gtZWQyNTUxOQAAACCFan6B7XSQ/xFJThqf7HPuJ9grMpUo277VSyZGuqF/fA  AAAED3KYuO+NUuDFR3NWBtzNAtVmwLSGO7luwsTqACQ/jzVIVqfoHtdJD/EUlOGp/sc+4n  2CsylSjbvtVLJka6oX98AAAAEXJvb3RAaXNjc2ktdGFyZ2V0AQIDBA==  -----END OPENSSH PRIVATE KEY-----  [root@iscsi-target ~]# **cat ~/.SSH/id\_ed25519.pub**  SSH-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIIVqfoHtdJD/EUlOGp/sc+4n2CsylSjbvtVLJka6oX98  [root@iscsi-target ~]# |

Клиентские открытые ключи нужно сохранить на SSH-сервере в файле **~/.SSH/authorized\_keys** (~ — домашняя директория того пользователя, которым будете логиниться), каждый на отдельной строке. Чтобы не делать это вручную, на каждом клиенте можно воспользоваться командой SSH-copy-id (в нашем случае — от пользователя root):

|  |
| --- |
| [root@iscsi-target ~]# **SSH-copy-id root@192.168.1.42**  /usr/bin/SSH-copy-id: INFO: Source of key(s) to be installed: "/root/.SSH/id\_ed25519.pub"  The authenticity of host '192.168.1.42 (192.168.1.42)' can't be established.  ECDSA key fingerprint is SHA256:Ro2rSQq+/bZVFP/5FLLpHMK1qvUbrXPVH9B/W2m5WFI.  ECDSA key fingerprint is MD5:47:62:08:27:1e:b2:83:f8:6f:b1:40:49:6c:9f:04:a0.  Are you sure you want to continue connecting (yes/no)? yes  /usr/bin/SSH-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  /usr/bin/SSH-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys  root@192.168.1.42's password:  Number of key(s) added: 1  Now try logging into the machine, with: "SSH 'root@192.168.1.42'"  and check to make sure that only the key(s) you wanted were added.  [root@iscsi-target ~]# |

Для начала нам придется ввести пароль пользователя на удаленном сервере. Это необходимо, чтобы залогиниться на сервер по паролю и скопировать на него публичный ключ. После можно будет подключаться без ввода пароля.

|  |
| --- |
| [root@iscsi-target ~]# **SSH root@192.168.1.42**  Last login: Sun Oct 13 18:46:58 2019 from 192.168.1.40  [root@iscis-initiator ~]# |

Как только мы проверили, что клиент может попасть на сервер при помощи SSH-ключа, следует отключить возможность доступа по паролю на сервере. Для этого в файле /etc/SSH/SSHd\_config установите следующие значения и перезапустите SSH-демон:

|  |
| --- |
| [root@iscis-initiator ~]# **vim /etc/SSH/SSHd\_config**  PubkeyAuthentication yes  PasswordAuthentication no  [root@iscis-initiator ~]# **systemctl restart SSHd**  [root@iscis-initiator ~]# |

При работе в Windows c PuTTY можно сгенерировать ключ с помощью puttygen. Загрузить публичный ключ .pub на сервер можно через любой sftp-клиент. В настройках аутентификации PuTTY SSH укажите путь к приватному ключу (обычно с расширением .ppk).

Обратите внимание, что форматы ключей отличаются: в authorized\_keys нужно вставить ключ в формате OpenSSH (начинается с SSH-rsa AAAA…).

Рекомендуем статью о [генерации SSH-ключей](https://lode.in/administration/generatsiya-ssh-klyuchej-s-putty-i-nastrojka-avtorizatsii-po-zakry-tomu-klyuchu-ssh.html).

## SOCKS

Протокол SOCKS v5 одобрен организацией IETF (Internet Engineering Task Force) в качестве стандарта Internet и включен в RFC 1928 (http://ypn.ru/366/). Согласно источнику http://ypn.ru/366/, протокол SOCKS относится к сеансовому уровню.

Аутентификация в SOCKS5 основана на SSH-ключах или паролях.

Современные браузеры могут использовать SOCKS5-прокси. В качестве адреса отправителя будет виден адрес SOCKS5-сервера, но при этом DNS-запросы (в отличие от классического HTTP-прокси, где DNS-запросы выполняет прокси-сервер) будут выполняться с вашей машины.

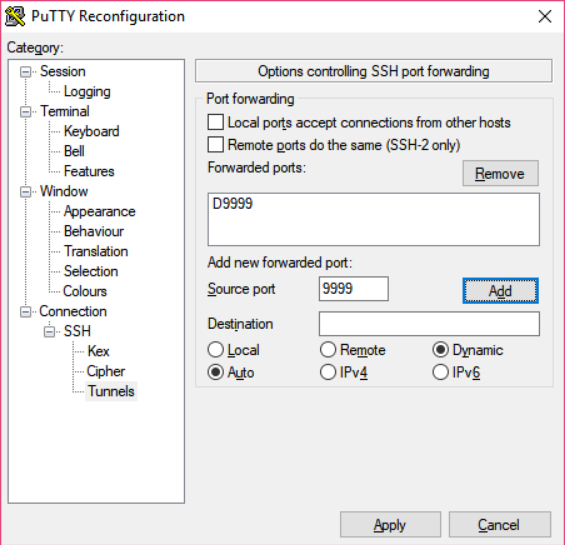
В качестве SOCKS5-прокси можно использовать openSSH-сервер на Linux-машине и PuTTY на клиентской (или SSH для Linux и MAC OS X-машин).

Разберем, что нужно для настройки SOCKS5-прокси. На сервере должен быть установлен OpenSSH-сервер (а также разрешен соответствующий доступ в файерволе), позволяющий выполнять подключения по протоколу SSH (secure shell) SFTP и предоставляющий множество возможностей туннелирования.

### Для подключения из Windows

Подключаемся к серверу по SSH с помощью Putty (если не меняли порт на 22). Затем заходим в настройки и выбираем Connection — SSH — Tunnels. Ставим тип порта Dynamic. В Source port вписываем номер необходимого TCP-порта (например, 9999). Нажимаем Add (иначе не сохранится). Применяем сохраненные данные в Apply.

В самой консоли делать ничего не надо, достаточно, чтобы прошла аутентификация. После этого можно использовать SOCKS5-прокси.



Если использовать не PuTTY, а линуксовый SSH-клиент, то же самое можно сделать при подключении к SSH-серверу с помощью такой команды:

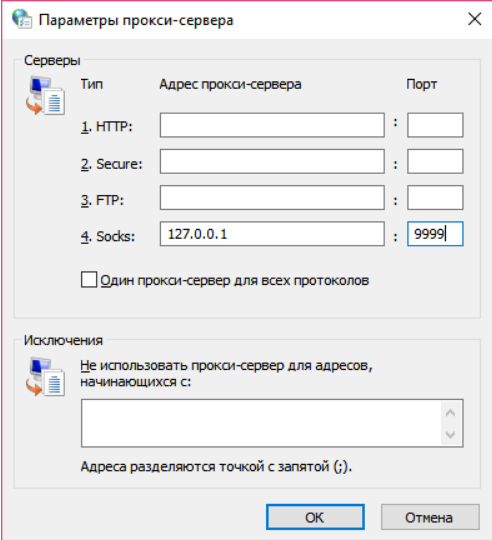
|  |
| --- |
| [root@iscsi-target ~]# **ssh -f -C2qTnN -D <порт> <удаленный\_пользователь>@<удаленный\_сервер>** |

* -f — запросит переход SSH в фоновый режим только перед выполнением команды.
* -C — включит сжатие всех данных (включая stdin, stdout, stderr и данные для перенаправленных Х11- и TCP/IP-соединений).
* -2 — принудит SSH использовать только протокол версии 2.
* -q — включит тихий режим, который подавляет все предупреждения и диагностические сообщения. Будут отображены только фатальные ошибки.
* -T — отменит переназначение терминала.
* -n — перенаправит стандартный ввод из /dev/null (предотвратит чтение из стандартного ввода).
* -N — укажет, что удаленную команду не надо выполнять.
* -D [локальный IP : ] — порт.

Например, со стороны клиента, чтобы поднять SOCKS-прокси на сервере с адресом my\_linux\_server.com и локальным для клиента портом 9999:

|  |
| --- |
| [root@iscsi-target ~]# **ssh -f -C2qTnN -D 9999 user@my\_linux\_server.com** |

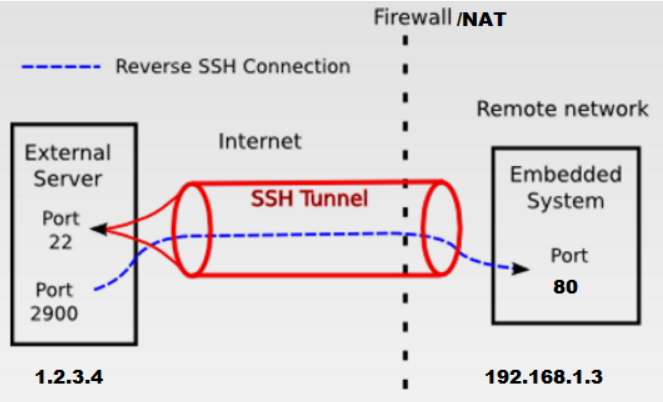
После того, как соединение установилось, мы можем сказать приложению, например браузеру, использовать прокси-сервер. В нашем случае это будет адрес 127.0.0.1, так как мы подключались с этой же машины, можем использовать адрес лупбека. Порт 9999.



### SSH-туннелирование

SSH (secure shell) — протокол, разработанный на смену устаревшим telnet и rlogin. Он фактически вытеснил и FTP, и протоколы удаленного копирования UUCP (unix-to-unix cp), и RCP (использующего в качестве транспорта rsh (remote shell), часть пакета rlogin). Протокол SFTP не является разновидностью FTP, хотя и выполняет аналогичные функции. SFTP использует в качестве транспорта SSH, как и протокол SCP, применяющий SSH для удаленного копирования файлов. В качестве клиента может выступать scp, WinSCP. SSH, SFTP и SCP используют для работы порт 22/TCP.

SSH позволяет туннелировать прикладные протоколы, пробрасывая TCP-порты, обходить NAT и файрволы, пробрасывать порты в обратную сторону и многое другое.



На картинке изображено подключение к машине, находящейся за NAT. Предположим, ее адрес — 192.168.1.3, соответственно, нет возможности принимать входящие соединения из внешней сети. Или у машины есть «белый» IP-адрес, но административно файерволом закрыта возможность передавать данные на порт 80 локальной машины (и другие ее порты, кроме исходящего трафика). В этом случае клиент подключается к внешнему серверу с «белым» IP-адресом (например, 1.2.3.4) по SSH и пробрасывает свой локальный порт 80 на порт сервера 1.2.3.4:2900.

Подключаясь на сервере к 1.2.3.4:2900, внешний клиент получает доступ к 192.168.1.3:80, который без SSH недоступен, но теперь будет доступен через туннель remote forward.

Запуск на клиенте:

|  |
| --- |
| [root@iscsi-target ~]# **SSH -R 127.1:80:1.2.3.4:2900 user@1.2.3.4** |

Обратите внимание, 127.1 — корректная запись для адреса 127.0.0.1.

Также с помощью SSH можно получить доступ к приложению, слушающему только адрес 127.0.0.1 при помощи туннеля local forward.

|  |
| --- |
| [root@iscsi-target ~]# **SSH -L 8600:127.1:3306 user@1.2.3.4** |

После этого, локально подключаясь к 127.0.0.1:8600 MySQL-клиентом, вы будете работать с MySQL на 1.2.3.4, слушающем 127.0.0.1:3306.

Более подробно изучить возможности SSH, научиться генерировать и использовать RSA-сертификаты для него можно по этому материалу. Узнать, как использовать SSH для VPN, можно здесь: https://habrahabr.ru/post/319158/. В разделе «Практика» этого урока есть пример подключения к mysql с использованием PuTTY.

### Удаленное копирование файлов

SCP (Secure CoPy) — утилита для удаленного копирования файлов по сети. По поведению похожа на cp, для транспорта использует SSH. Если на сервере установлен SSH (например, openSSH-server), то на него можно закачивать файлы с помощью SCP, и точно так же можно получать данные с помощью этой утилиты.

Копирование файла на удаленный сервер. Структура аналогична утилите cp: scp <что\_копируем> <куда\_копируем>.

Можно обращаться по IP-адресу или доменному имени.

|  |
| --- |
| [root@iscsi-target ~]# **scp backup.sh user@server.com:/home/user/scripts** |

Или по IP:

|  |
| --- |
| [root@iscsi-target ~]# **scp backup.sh user@10.0.2.8:/home/user/scripts** |

Например:

|  |
| --- |
| [root@iscsi-target ~]# **scp 192.168.1.42:/etc/hosts .**  hosts 100% 158 251.2KB/s 00:00  [root@iscsi-target ~]# ll | grep hosts  -rw-r--r--. 1 root root 158 Oct 19 14:59 hosts |

Следуя той же логике, мы можем загрузить локальный файл на удаленный хост. Например, создадим файл test\_scp:

|  |
| --- |
| [root@iscsi-target ~]# touch test\_scp  [root@iscsi-target ~]# scp test\_scp 192.168.1.42:/root  test\_scp 100% 0 0.0KB/s 00:00  [root@iscsi-target ~]# SSH 192.168.1.42  Last login: Sun Oct 13 21:34:11 2019 from 192.168.1.41  [root@iscis-initiator ~]# ll  total 4  -rw-------. 1 root root 1426 Sep 14 12:46 anaconda-ks.cfg  -rw-r--r--. 1 root root 0 Oct 13 21:56 test\_scp |

# 

# Дополнительные материалы

1. [SSH вместо VPN](https://habrahabr.ru/post/319158/).
2. [SSH](https://ru.wikipedia.org/wiki/SSH).
3. [10 примеров использования команды dig для просмотра параметров DNS (DNS Lookup) в Linux](http://greendail.ru/node/474).
4. [BIND 9](http://help.ubuntu.ru/wiki/bind9).
5. [NetSkills. Видеоуроки. Cisco, Zabbix, Linux](https://www.youtube.com/watch?v=2MLE7TIljlY).
6. [Как сгенерировать открытый/закрытый SSH-ключ в Linux](http://rus-linux.net/MyLDP/sec/generiruem-kluchi-SSH.html).
7. [Генерация SSH-ключей с PuTTY и настройка авторизации по закрытому ключу SSH](https://lode.in/administration/generatsiya-SSH-klyuchej-s-PuTTY-i-nastrojka-avtorizatsii-po-zakry-tomu-klyuchu-SSH.html).

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. [SELinux](https://ru.wikipedia.org/wiki/SELinux).
2. [Настройка и использование ACL в Linux](https://linuxsoid.club/setup-use-acl-linux).
3. [Как сгенерировать открытый/закрытый SSH-ключ в Linux](http://rus-linux.net/MyLDP/sec/generiruem-kluchi-SSH.html).
4. [Установка и настройка SSH](http://a-rm.ru/materials/unix-servera/ustanovka-i-nastroyka-SSH-ubuntu).
5. [Памятка пользователям SSH](https://habrahabr.ru/post/122445/).
6. [WinSCP](https://ru.wikipedia.org/wiki/WinSCP).