**Лабораторная работа №2**

**Основы TCP/IP сетей. Настройка и диагностика сети**

# Теоретическая часть

**Сети на основе стека Internet протоколов**

* Современные компьютерные сети строятся на базе стека протоколов семейства TCP/IP
* Разработка стандартов, известных как Request for Comments (RFC), ведется организацией Инженерный Совет Интернета (Internet Engineering Task Force, IETF). Описания протоколов доступны по адресу https://tools.ietf.org/rfc/index
* Согласно rfc1122 стек Internet протоколов состоит из 4-х уровней:
  + Прикладной уровень (Application Level)
  + Транспортный уровень (Transport Level)
  + Межсетевой уровень (Internet Level)
  + Канальный уровень (Link Layer)

1. Протоколы прикладного уровня разделяются на две категории:
   * пользовательские протоколы (SSH, Telnet, FTP, SMTP);
   * протоколы поддержки (DNS, BOOTP, SNMP).
2. Протоколы транспортного уровня обеспечивают взаимодействие между сетевыми приложениями.
3. Описаны два протокола данного уровня:
   * TCP (Transmission Control Protocol);
   * UDP (User Datagram Protocol).
4. На транспортном уровне определяются адреса сетевых приложений в рамках хоста — порты.
5. На межсетевом уровне данные передаются от одного хоста другому хосту, которые находятся в разных сетях.
6. За маршрутизацию отвечает основной протокол данного уровня Internet Protocol (IP). На этом же уровне определены управляющие протоколы ICMP и IGMP и маршрутизирующие протоколы RIP, OSPF, BGP и др.
7. Канальный уровень в модели TCP/IP подробно не описывается, протоколы в рамках модели не конкретизируются.

**IP адреса**

* Каждому сетевому интерфейсу должен быть присвоен IP адрес
* IP адрес можно назначить статически, либо динамически с помощью служб DHCP или BOOTP
* Существует две версии IP протокола, которым соответствуют разные форматы адресов:
  + IPv4 — четыре байта. Записывается в виде 4-х десятичных чисел, разделенных точками
  + IPv6 — шестнадцать байтов (128 бит).Записывается в виде 16-ти чисел в шестнадцатиричном формате, разделяемых двоеточиями

1. В сетях IPv4 поддерживаются три типа адресов:
   * unicast – адрес хоста;
   * broadcast – адрес всех хостов в сети;
   * multicast – адрес группы хостов.
2. В каждом IP адресе выделяется две части:
   * номер сети;
   * номер хоста в этой сети.
3. Первый адрес в сети – адрес самой сети, последний адрес – адрес всех хостов в данной сети.

**Классы IP адресов**

* IP адреса разделены на 5 классов: A, B, C, D, E:
  + Класс A — для номера сети используется первый байт, первый бит — 0
  + Класс B — для номера сети используются первые два байта, первые 2 бита — 10
  + Класс С — для номера сети используются первые три байта, первые 3 бита — 110
  + Класс D предназначен для поддержки групповой рассылки данных, первые 4 бита — 1110
  + Класс E зарезервирован, первые 4 бита — 1111

1 Согласно RFC 1700 выделены специальные IP адреса:

* <номер сети, 0> — обозначает всю сеть;
* <номер сети, все 1> — широковещательные адреса сети;
* <все 1, все 1> — локальная рассылка (рассылка внутри своей сети);
* <все 0, все 0> — текущая сеть, текущий хост;
* <127, произвольно> — loopback (любой пакет направленный на этот адрес, не достигая передающей среды, передается локальному сетевому приложению).

**Адреса для частных сетей**

* RFC 1597 и 1918 определяют понятие частных (private) сетей
* Приватные сети предназначены для:
  + хостов, которым не нужен доступ к публичной сети Интернет
  + хостов, изолированных от внешней сети шлюзом прикладного уровня (прокси)
* IANA (Internet Assigned Numbers Authority) выделила три блока адресов для частных сетей:
  + 10.0.0.0 – 10.255.255.255 — одна сеть класса A
  + 172.16.0.0 – 172.31.255.255 — 16 сетей класса B
  + 192.168.0.0 – 192.168.255.255 — 256 сетей класса C

**Подсети**

* Для преодоления недостатков схемы адресации, базирующейся на классах было предложено следующее:
  + разбиение на подсети (subnetting) и сетевая маска
  + сетевые маски переменной длины (VLSM, Variable Length Subnet Mask)
  + бесклассовая адресация (CIDR, Classless InterDomain Routing)
* В RFC 917 и 950 введена концепция подсети (subnet), которая позволяет разбивать сети класса A, B и C на более мелкие подсети
* В рамках концепции подсети адрес из любой сети класса A, B или C состоит из трех частей: номера сети, номера подсети, номера хоста

1. Номер подсети и номер хоста содержатся в номере хоста исходной сети.
2. Номер сети и номер подсети образуют вместе расширенный сетевой префикс.
3. Для выделения сетевого префикса и номера хоста используется маска подсети.
4. Маска подсети — это 32-битовое число, биты маски равные 1 – выделяют биты в адресе, используемые в качестве сетевого префикса, биты маски равные 0 — для нумерации хостов.

**Бесклассовая адресация**

* В RFC 1009 определен механизм поддержки нескольких масок в одной сети. Механизм получил название масок подсетей переменной длины (VLSM)

и позволяет разбивать сеть на подсети разных размеров

* В RFC 1517, 1518, 1519, 1520 был описан механизм бесклассовой междоменной маршрутизации CIDR
* Идеология CIDR заключается в:
  + отказе от классовой адресации
  + агрегировании маршрутных данных
  + концепции суперсетей

1. В бесклассовой адресации разделение адреса на номера сетей и хостов происходит исключительно на основе сетевой маски (сетевого префикса).
2. Адреса в бесклассовом виде принято записывать с помощью битовой маски: адрес/кол-во\_битов\_соответствующих\_номеру\_сети.
3. Агрегирование маршрутных данных позволяет уменьшать размеры таблиц маршрутизации, поскольку отпадает необходимость записывать маршруты до каждой сети.
4. Суперсеть — объединение нескольких смежных сетей C в одно адресное пространство.

**Основные TCP/IP протоколы**

* Основные протоколы семейства протоколов TCP/IP:
  + межсетевой протокол IP (Internet Protocol)
  + межсетевой протокол управления сообщениями ICMP (Internet Control Message Protocol)
  + протокол преобразования адресов ARP (Address Resolution Protocol)
  + протокол пользовательских датаграмм UDP (User Datagram Protocol)
  + протокол управления передачей TCP (Transmission Control Protocol)

1 Семейство протоколов TCP/IP состоит из следующих основных протоколов:

* межсетевой протокол IP (Internet Protocol), который обеспечивает доставку с одной машины на другую без дополнительной обработки данных;
* межсетевой протокол управления сообщениями ICMP (Internet Control Message Protocol), который отвечает за низкоуровневую поддержку работы протокола IP такую, как сообщения об ошибках, содействие

в маршрутизации и т.п.;

* протокол преобразования адресов ARP (Address Resolution Protocol), выполняющий трансляцию логических сетевых адресов в аппаратные (MAC адреса);
* протокол пользовательских датаграмм UDP (User Datagram Protocol), который обеспечивает негарантированную пересылку данных между сетевыми процессами;
* протокол управления передачей TCP (Transmission Control Protocol), который отвечает за гарантированную доставку данных между сетевыми процессами.

**Address Resolution Protocol**

* Для автоматического преобразования IP адреса в канальный адрес предназначен протокол ARP (Address Resolution Protocol, RFC 826)
* Информация о соответствии программных адресов аппаратным хранится в памяти ядра в ARP таблице (ARP cache)
* Записи в ARP таблице могут быть статическими и динамическими
* Команды ip neigh и arp позволяют просматривать, добавлять и удалять записи в ARP таблице
* Параметры, управляющие работой ARP, находятся в /proc/sys/net/ipv4/neigh/default
* ARP протокол не является маршрутизируемым (ARP пакеты не выходят за пределы локального сетевого сегмента)

1. Структура ARP пакета имеет следующий вид:
   * Тип оборудования (Ethernet — 1, IEEE 802.x — 6).
   * Тип протокола (IP — 2048, ARP — 2054).
   * Длина аппаратного адреса (для Ethernet – 6 байтов).
   * Длина программного адреса (для IP — 4 байта).
   * Код операции (для запросов — 1, для ответов — 2).
   * Аппаратный адрес отправителя.
   * IP адрес отправителя.
   * Аппаратный адрес получателя.
   * IP адрес получателя.
2. Алгоритм работы протокола ARP:
   * ARP модуль по заданному IP адресу ищет аппаратный адрес в ARP кэше, если в кэше есть нужный адрес, то аппаратный адрес возвращается сетевому модулю.
   * Если запись в кэше отсутствует, то ARP модуль отправляет широковещательный кадр канального уровня с ARP запросом внутри, при этом в запросе указываются оба адреса отправителя и IP адрес получателя
   * Все хосты получают ARP запрос, но отвечает только тот хост, чей IP адрес указан в поле IP адреса получателя.
   * ARP модуль получателя проверяет тип оборудования и тип протокола.
   * Если типы оборудования поддерживаются данным хостом, то хост запоминает адреса отправителя в кэше, подставляет в ARP ответ свой аппаратный адрес, переставляя местами адреса отправителя и получателя.
   * ARP ответ направляется хосту, пославшему запрос, который помещает (или обновляет) полученную пару адресов в свой кэш.
3. Для повышения эффективности работы механизма ARP каждый хост

при включении рассылает широковещательный запрос. В качестве IP адреса получателя подставляется IP адрес отправителя.

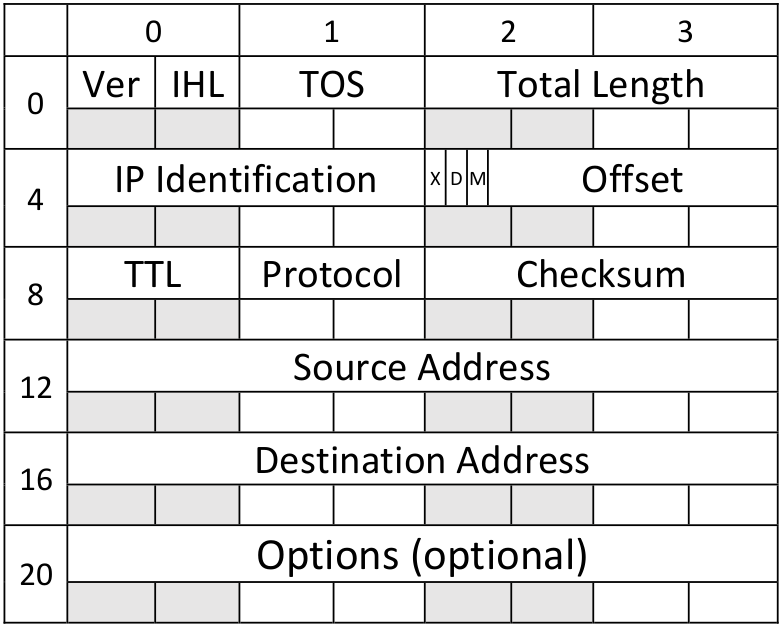
1. Все хосты, получившие такой широковещательный запрос:
   * запоминают адреса отправителя в своем ARP кэше и не отвечают на запрос;
   * в случае, если хост получатель обнаруживает, что его адрес совпадает с адресом хоста отправителя, то хост-отправитель извещается

о дублирующем адресе.

1. Для обновления данных в ARP таблице хост регулярно направляет одноадресный (unicast) запрос нужному хосту.
2. Если хост несколько раз не отвечает на ARP запрос, то соответствующая строка в ARP кэше хоста-отправителя удаляется.
3. Proxy ARP — модифицированный вариант ARP протокола. Используется шлюзами, которые находятся в нескольких сетях одновременно

и заключается в объединении нескольких ARP таблиц в одну.

**Internet Protocol**



Пояснения:

IHL — в 4б

размер от 576б до 64Кб

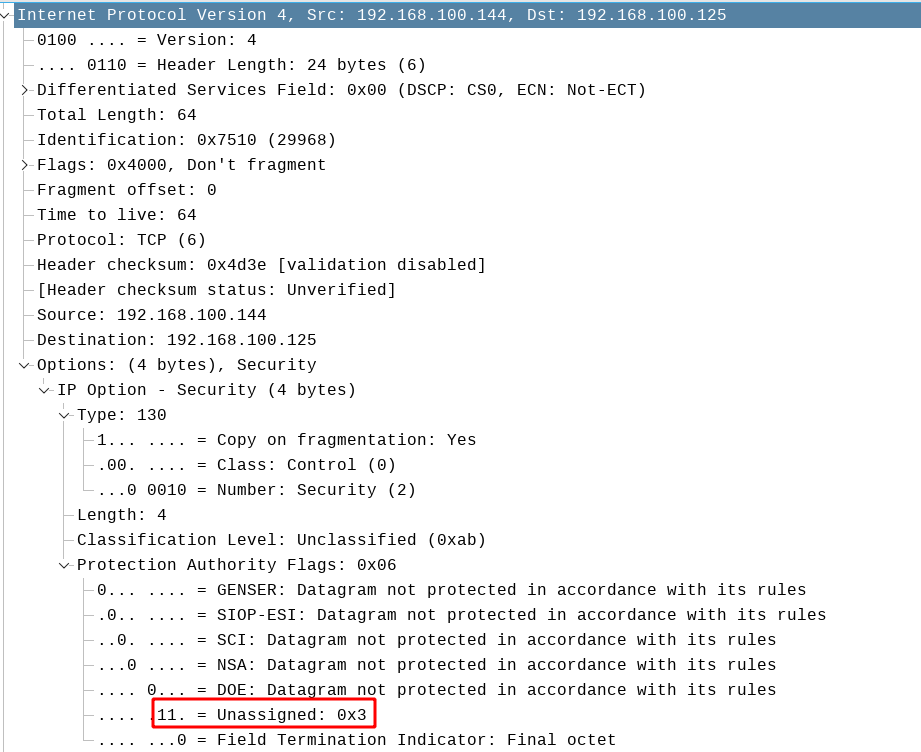
D - запрет фрагментации M - есть фрагменты

TTL — кол-во «прыжков» sysctl net.ipv4.ip\_default\_ttl

/etc/protocols

1. IP дейтаграмма состоит из заголовка и данных. Данные получены от протоколов верхнего уровня.
2. IP заголовок состоит из стандартного заголовка фиксированной длины (20 байт) и из поля опций (options) переменной длины. Макс. длина — 60б.
3. Стандартный IP заголовок содержит следующие поля:
   * Версия (Version, V) — 4 бита.
   * Длина заголовка (header length, IHL ) — 4 бита.
   * Тип службы (Type of Service, ToS) — 1 байт.
   * Общая длина (total length) — 2 байта.
   * Идентификатор (Identification) — 2 байта.
   * Флаги (Flags) — 3 бита.
   * Смещение фрагмента (Fragment offset) — 13 бит.
   * Время жизни (TTL, Time To Live) — 1 байт.
   * Протокол (Protocol) — 1 байт.
   * Контрольная сумма заголовка (Header Checksum) — 2 байта.
   * Адрес отправителя (Source address) — 4 байта.
   * Адрес получателя(Destination address) — 4 байта.

**Метки безопасности в IP-пакетах**



1. Классификационные метки (уровни конфиденциальности и категории) помещаются в дополнительную часть IP заголовка — опцию типа базовая безопасность (Basic Security Option).
2. Общие требования к использованию опций безопасности в IP заголовке изложены в rfc1108 «Security Options for the Internet Protocol».
3. Формат и правила установки классификационных меток в IPv4 сетях, используемых в системах с установленной ОС Astra Linux, описываются в стандарте ГОСТ Р 58256-2018 «Управление потоками информации

в информационной системе. Формат классификационных меток». Примечание: на слайде показан заголовок перехваченного в программе Wireshark IP-пакета, в который помещена классификационная метка со значением уровня конфиденциальности равным 3.

1. В случае отсутствия в заголовке IP-пакета опции безопасности считается, что информация передается с нулевым уровнем конфиденциальности

и без категорий.

**Transmission Control Protocol**

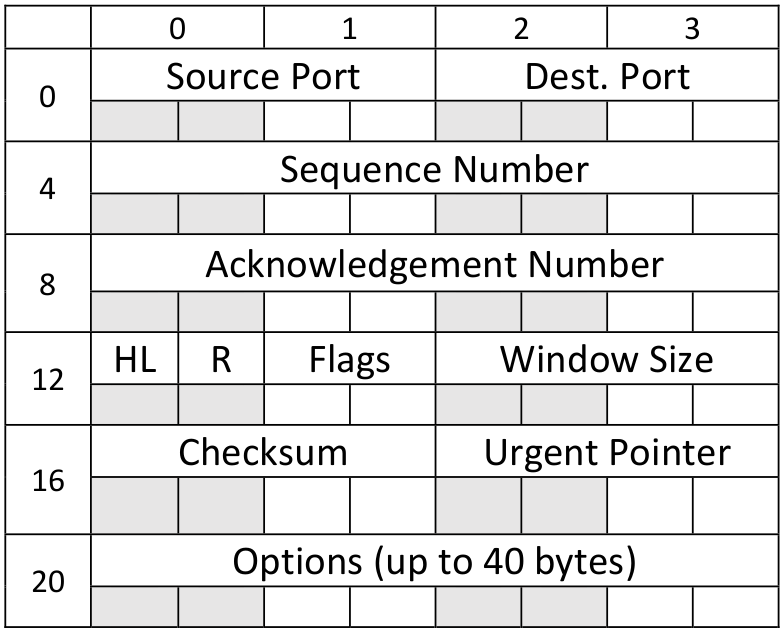


Рисунок TCP заголовок

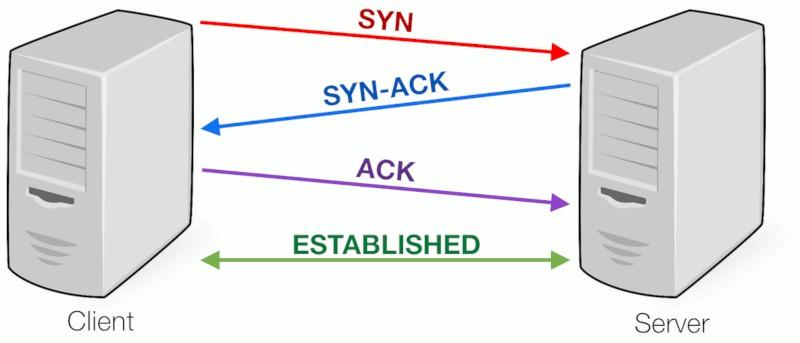


Рисунок 2 Процедура тройного рукопожатия

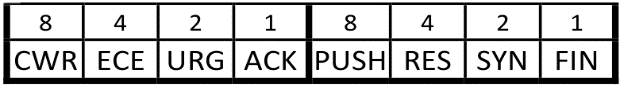


Рисунок 3 TCP флаги

1. Протокол TCP (RFC 793, 1122, 3168, 6093, 6528) ориентирован

на установление соединения и на надежную доставку сегментов данных от одного сетевого процесса другому.

1. Для нумерации сетевых процессов на одном хосте используется дополнительное число – номер порта (rfc6335, общеизвестные (системные) (0—1023), зарегистрированные (или пользовательские, 1024—49151

и динамические (49152—65535), выделяет и регистрирует IANA,

/etc/services).

1. TCP сегмент состоит из заголовка (24 байта) и данных (нет ограничения).
2. Установка TCP соединения (сокетное соединение) происходит путем троекратного обмена сегментами (процедура тройного рукопожатия).
3. Надежность доставки сегментов данных обеспечивается применением схемы повторной передачи с позитивным подтверждением (PAR, Positive Acknowledgment Retransmission).
4. Для повышения эффективности передачи TCP сегментов используется механизм скользящего окна.
5. Модуль TCP может использоваться несколькими прикладными процессами (мультиплексирование).

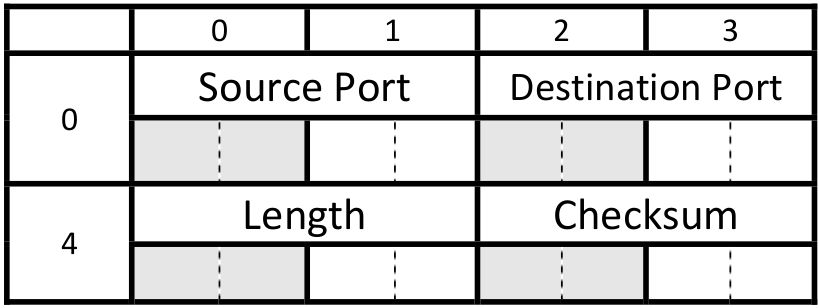
**Состояния TCP сокетов**

* Типовая последовательность состояний TCP соединения со стороны сервера:
  + CLOSED — открываем серверный сокет на прослушивание
  + LISTEN — получаем SYN, отправляем SYNACK
  + SYN\_RCVD — получаем ACK
  + ESTABLISHED — получаем FIN, отправляем ACK, отправляем данные
  + CLOSE\_WAIT — отправляем FIN
  + LAST\_ACK — получаем ACK
  + CLOSED

1 Типовая последовательность состояний TCP соединения со стороны клиента:

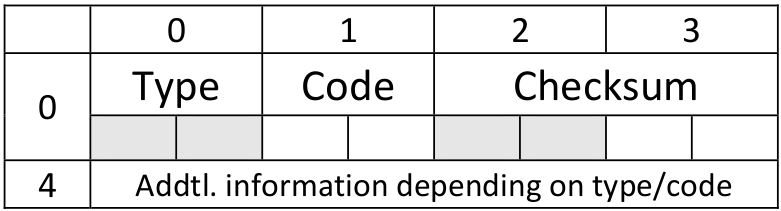
* CLOSED — начинаем соединение, отправляем SYN;
* SYN\_SENT — получаем SYNACK, отправляем подтверждение;
* ESTABLISHED — отправляем FIN, получаем данные;
* FIN\_WAIT\_1 — получаем ACK;
* FIN\_WAIT\_2 — получаем FIN, отправляем ACK;
* TIME\_WAIT — ждем;
* CLOSED.

**User Datagram Protocol**



1. Протокол UDP (User Datagram Protocol, rfc 768) не устанавливает соединение.
2. UDP не пытается упорядочить данные, упорядочивание данных возлагается на приложение.
3. Протокол UDP (также как и TCP) может различать прикладные процессы по номерам портов.
4. UDP поддерживается проверка контрольной суммы данных.
5. UDP быстрее и эффективней, чем TCP, но уступает в надежности и могут возникать проблемы с перегрузкой.

**Internet Control Message Protocol**



1. Протокол ICMP (Internet Control Message Protocol, RFC 792) предназначен для обмена информацией сетевого уровня между хостами, как правило, для передачи сообщений об ошибках и исключительных ситуаций.
2. ICMP работает поверх IP протокола, помещая ICMP сообщения внутрь IP пакета.
3. Наиболее часто используются следующие комбинации типов и кодов:
   * тип=0, код=0 — эхо-ответ на команду ping (echo reply);
   * тип=3, код=0 — сеть-адресат недоступна (network unreachable);
   * тип=3, код=1 — хост-получатель недоступен (host unreachable);
   * тип=3, код=2 — протокол адресата недоступен (protocol unreachable);
   * тип=3, код=3 — порт адресата недоступен (port unreachable);
   * тип=3, код=4 — пакет уничтожен, поскольку требуется фрагментация, но флаг DF=1;
   * тип=3, код=6 — сеть-адресат неизвестна (network unknown);
   * тип=3, код=7 — хост-получатель неизвестен (host unknown);
   * тип=8, код=0 — запрос эха (echo request);
   * тип=11, код=0 — время жизни пакета истекло (time exceeded);
   * тип=12, код=0 — неверный IP заголовок (parameter problem).

**Основные протоколы уровня приложений**

* DNS
* DHCP
* SMTP, POP, IMAP
* SSH
* NTP
* NFS
* CIFS
* FTP, TFTP, Telnet
* IPP

**Именование сетевых интерфейсов**

* Традиционные имена для Ethernet интерфейсов: ethX
* Предсказуемые имена для Ethernet интерфейсов:
  + В Astra Linux нужно убрать параметр net.ifnames=0, передаваемый ядру (/etc/default/grub)
  + Политики именования:
    - BIOS — enoX
    - PCIe — ensX
    - Физическое расположение — enpXsY
    - MAC-адрес — enxXXXXXXXXXXXX
  + Префиксы для других сетевых интерфейсов (ib, sl, wl, ww)

1 Чтобы обеспечить постоянство традиционных имен для интерфейсов, следует определить правило udev типа:

SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?\*",

ATTR{address}=="xx:xx:xx:xx:xx:xx", ATTR{type}=="1", KERNEL=="eth\*", NAME="ethX"

**Настройка сетевых интерфейсов**

* Network Manager:
  + NetworkManager.service
  + nmcli, nmtui, nm-connection-editor
* ifup/ifdown команды:
  + networking.service
  + /etc/network/interfaces
* Команды для ручной настройки сетевых интерфейсов
* Просмотр настроек: ip address show или ifconfig -a
* Изменение настроек: ip address add A.B.C.D/M dev eth0

1. Если установлено графическое окружение, то все сетевые интерфейсы, кроме loopback (lo), находятся по умолчанию под управлением Network Manager:
   * за запуск Network Manager отвечает служба NetworkManager.service;
   * команды Network Manager для настройки соединений:nmcli (текст), nmtui (меню), nm-connection-editor (графика);
   * из консоли изменять сетевые настройки может обычный пользователь, права root не требуются.
2. Если графическое окружение не установлено, то сетевые интерфейсы инициализируются командами ifup/ifdown команды:
   * Команды ifup/ifdown запускаются службой networking.service.

sa@server1-astra:~$ systemctl cat networking.service

...

[Service] Type=oneshot

...

ExecStart=/sbin/ifup -a --read-environment ExecStop=/sbin/ifdown -a --read-environment --exclude=lo

...

* + Основной конфигурационный файл для ifup/ifdown команд -

/etc/network/interfaces.

* + В случае установленного графического окружения ifup/ifdown команды инициализируют только интерфейс loopback (lo).

1. Для настройки сетевых интерфейсов также может использоваться внутренняя подсистема управления сетевыми интерфейсами systemd:
   * по умолчанию служба systemd-networkd.service неактивна;
   * настройка службы systemd-networkd описана в man systemd.network.
2. Для просмотра и ручной временной настройки сетевых интерфейсов могут быть использованы команды ip и ifconfig.
3. Просмотр настроек сетевых интерфейсов осуществляется командами:
   * ip address show
   * ifconfig -a
4. Изменение настроек сетевых интерфейсов производятся так:
   * ip address add A.B.C.D/M dev ethX
   * ifconfig ethX A.B.C.D/M

**Настройка с помощью Network Manager**

* Конфигурационный файл /etc/NetworkManager/NetworkManager.conf
* Основные типы объектов:
  + device и connection
  + объект device является частью объекта connection
  + каждый объект имеет свой набор свойств
* Просмотр настроек: команда show nmcli device **show** имя\_интерфейса nmcli connection **show** имя\_соединения
* Создание/изменение соединений: команды add/modify
* nmcli con **mod** path 1 ip4 A.B.C.D/M ipv4.method manual

1. В конфигурационном файле NetworkManager.conf по умолчанию содержится секция, в которой указывается, что NetWork Manager будет игнорировать интерфейсы, управляемые с помощью ifup/ifdown (изначально, в файле

/etc/network/interfaces есть только lo)

[ifupdown] managed=false

1. Примеры использования команды nmcli приведены в man-странице nmcli- examples

man nmcli-examples

1. В команде nmcli вместо имени соединения можно использовать path #

nmcli con sh path 1

1. Информация о соединениях (connection) хранится в /etc/NetworkManager/system connections
2. Для статического определения IP адреса следует использовать значение manual для параметра ipv4.method (по умолчанию auto, т.е. адрес получается по DHCP).

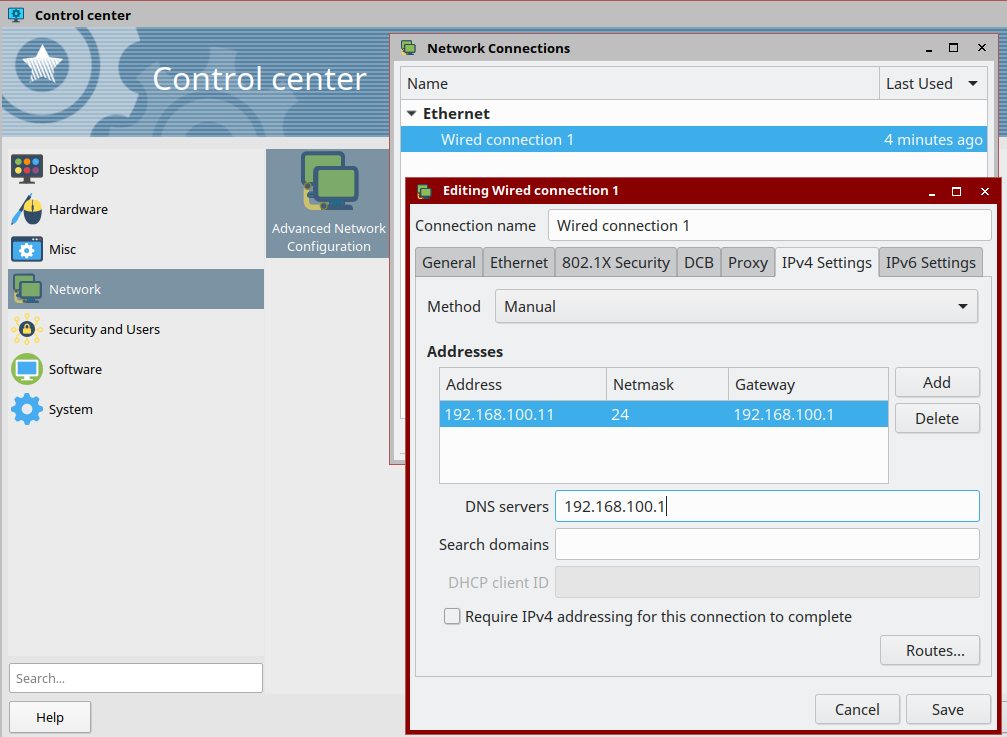
nmcli con mod имя\_соединения ipv4.method manual \ ipv4.addresses A1.B1.C1.D1/M ipv4.gateway A2.B2.C2.D2

1. Изменить настройки соединение можно также с помощью:
   * графической утилиты **Сетевые соединения** (**Advanced Networking Configuration**), которую можно вызвать из **Панели управления** (**Control center**) или командой

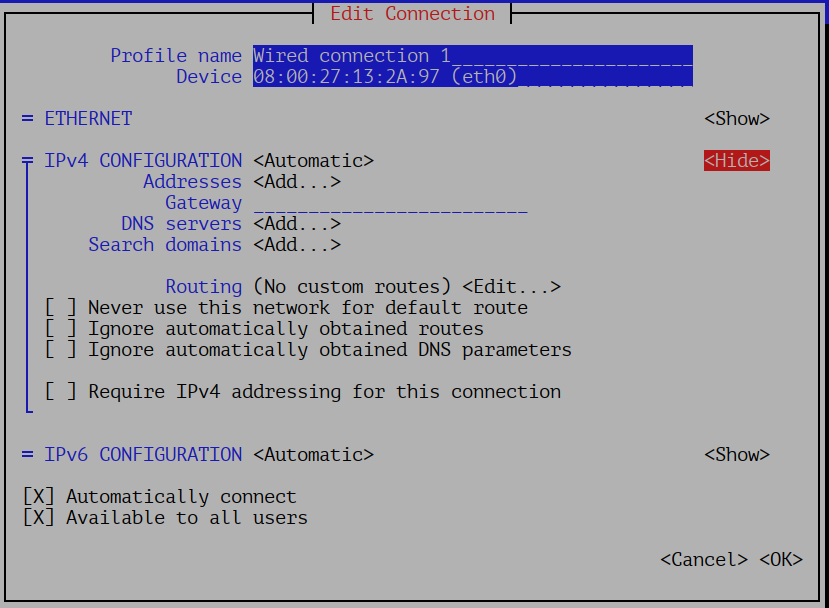
nm-connection-editor;

* + псевдографической утилиты nmtui.

1. Вид графической утилиты **Сетевые соединения** (**Advanced Networking Configuration**):



1. Вид псевдографической утилиты nm-tui:



**Настройка с помощью ifup/ifdown команд**

* Команды: ifup, ifdown, ifquery
* Служба networking.service
* Конфигурационный файл /etc/network/interfaces
  + **auto** ethX — активация интерфейса при загрузке
  + **iface** ethX **inet dhcp**/**static** — метод получение адреса (dhcp или статика)
  + **address** A.B.C.D/M — статическое определение адреса/маски
  + **gateway** А.B.C.D — статическое определение шлюза

1. Подробно о настройках файла /etc/network/interfaces изложено в man- странице interfaces:

man interfaces

1. В случае, если настройка сетевых интерфейсов будет производиться через ifup/ifdown команды, то рекомендуется отключить службу NetworkManager:

systemctl mask NetworkManager

**Агрегирование (объединение) Ethernet интерфейсов**

* Основные режимы агрегирования:
  + balance-rr (0) — циклический (карусельный) — баланс., отказ.
  + active-backup (1) — активный-резервный — отказ.
  + balance-xor (2) — каждый интерфейс работает со своим получателем —

баланс., отказ.

* + balance-tlb (5) — исходящий трафик распределяется исходя

из загруженности каждого интерфейса (transmit load balancing)

* + balance-alb (6) — входящий и исходящий трафики распределяются исходя из загруженности каждого интерфейса (adaptive load balancing)

1. Агрегирование сетевых интерфейсов - это объединение нескольких физических интерфейсов в один логический.
2. Агрегирование сетевых интерфейсов позволяет:
   * увеличить пропускную способность (но не скорость!) каналов передачи данных за счет различных методов балансировки нагрузки;
   * повысить отказоустойчивость.
3. Алгоритм создания bonding с помощью Network Manager (команда nmcli):
   * создаем логический ведущий (master) интерфейс;

nmcli con add type bond ifname имя\_мастера

* + определяем подчиненные (slave) интерфейсы;

nmcli con add type ethernet ifname ethX master имя\_мастера nmcli con add type ethernet ifname ethY master имя\_мастера

* + активируем соединения, соответствующие подчиненным интерфейсам.

nmcli con up bond-slave-ethX nmcli con up bond-slave-ethY

1. Изменение поведения bonding производится с помощью параметра bond.options.

nmcli con mod bond-имя\_мастера bond.options "mode=active- backup"

nmcli con up bond-имя\_мастера. Если требуется задать статический адрес соединению типа bonding, то это делается точно так же, как и для обычного соединения. В качестве имени соединения указываем bond-имя\_мастера.

1. В случае настройки bonding через ifup/ifdown команды в файл

/etc/network/interfaces помещаются настройки вида:

auto имя\_мастера ethX ethY

iface имя\_мастера inet static address A1.B1.C1.D1/M gateway A2.B2.C2.D2

bond-slaves ethX ethY bond-mode active-backup

1. Мониторинг поведения bonding осуществляется путем чтения файла

/proc/net/bonding/имя\_мастера:

watch cat /proc/net/bonding/bond0

**Маршрутизация**

* Маршрутизация позволяет составить путь для доставки сетевых (IP) пакетов от хоста-отправителя, находящегося в одной сети, хосту- получателю, расположенному в другой сети
* Таблица маршрутизации — это структура данных, хранящаяся в ядре и содержащая информацию о доступных путях в сети

(netstat -r, ip route, route)

* Два типа маршрутизации в IP сетях: статическая и динамическая (внутренняя (interior) и внешняя (exterior))
* В корпоративных сетях используют протоколы внутренней динамической маршрутизации: RIP (Routing Information Protocol) и OSPF (Open Shortest Path First) (bird, quagga)

1. Алгоритм построения маршрута:
   * перед отсылкой пакета хост-отправитель определяет по маске сети, в какой сети находится хост-получатель;
   * если хост-получатель находится в той же сети, что и хост-отправитель, то пакет доставляется с использованием механизмов уровней ниже сетевого (IP);
   * в этом случае используется ARP протокол для определения аппаратного (MAC) адреса, соответствующего IP адресу хоста-получателя;
   * если хост-получатель находится в другой сети, то хост-отправитель определяет, на какой хост-маршрутизатор следует отправить пакет;
   * когда маршрутизатор получает пакет, то он по таблице маршрутов (маршрутизации) определяет, в какую сеть данный пакет должен быть направлен;
   * если маршрутизатор не может найти маршрут, по которому следует отправить пакет, то хосту-источнику отправляется ICMP сообщение “Destination Unreachable”, а сам пакет уничтожается (отбрасывается);
   * если определен, так называемый, маршрут по умолчанию (default route), то пакет не отбрасывается, а направляется по этому маршруту.
2. Во время доставки до хоста-получателя пакет может проходить

через несколько маршрутизаторов. Каждый шаг в таком процессе (т.е. прохождение каждого маршрутизатора) называется хопом (hop, прыжок).

1. При статической маршрутизации маршруты устанавливаются и изменяются администратором системы.
2. В случае динамической маршрутизации таблица маршрутизации конструируется и перестраивается динамически. Динамические маршруты вычисляются в режиме “реального времени” в зависимости от сетевых условий.
3. Маршрутизируемый (routed) протокол – протокол, пакеты которого могут пересылаться в другие сети. (Пример – IP протокол).
4. Маршрутизирующий (routing) протокол - протокол, который определяет, куда нужно направить пакеты маршрутизируемого протокола.

**Статическая маршрутизация**

* Путь по умолчанию — статический маршрут, который используется, когда другие правила маршрутизации не срабатывают
* Для обозначения маршрута по умолчанию используются default или 0.0.0.0
* Для временного определения статических маршрутов можно использовать команды ip route или route
* Постоянное определение статических маршрутов:
  + в команде nmcli используются параметры ipv4.gateway для шлюза по умолчанию и ipv4.routes для статических маршрутов
  + в файле /etc/network/interfaces - GATEWAY - для шлюза по умолчанию и post-up/pre-down команда - для установки статических маршрутов

1. В случае наличия в системе нескольких сетевых интерфейсов на уровне ядре ОС должна быть включена внутренняя маршрутизация (за это отвечает параметр ядра ip\_forward):

sudo sysctl -w net.ipv4.ip\_forward=1 - немедленное включение маршрутизации в ядре или

в файл /etc/sysctl.conf добавить

net.ipv4.ip\_forward=1 - для постоянного включения внутренне маршрутизации.

1. Временное определение статических маршрутов выполняются командами ip route или route:

ip route add A.B.C.D/M via A1.B1.C1.D1

route add -net A.B.C.D netmask M1.M2.M3.M4 gw A1.B1.C1.D1

1. Команда nmcli позволяет определить статические маршруты на постоянной основе:
   * шлюз по умолчанию;

nmcli con mod имя\_соединения **ipv4.gateway** A.B.C.D

* + статические маршруты (сеть/шлюз).

nmcli con mod имя\_соединения **+ipv4.routes** \ "A.B.C.D/M A1.B1.C1.D1"

1. Для постоянного задания статических маршрутов через ifup/ifdown команды в файл /etc/network/interfaces добавляются следующие строки:
   * шлюз по умолчанию;

GATEWAY=A.B.C.D

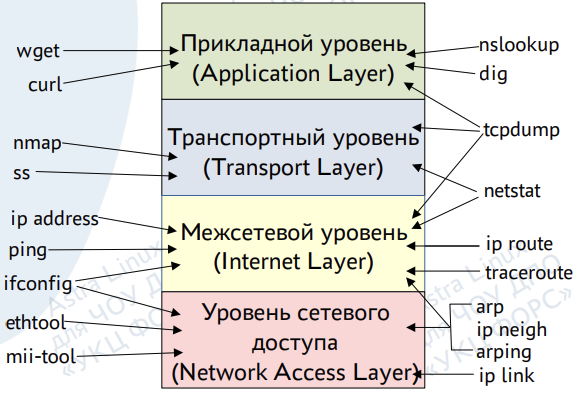
* + статические маршруты (добавление и удаление).

post-up ip route add A.B.C.D/M via A1.B1.C1.D1 pre-down ip route del A.B.C.D/M via A1.B1.C1.D1

**Типовые проблемы, связанные с сетевыми неисправностями**

* Отсутствие сетевого соединения:
  + проблемы с сетевым оборудованием
  + проблемы с сетевыми настройками
* Отсутствие доступа к удаленному хосту:
  + проблемы с маршрутизацией IP пакетов
* Невозможность получения доступа к удаленной службы:
  + проблемы с разрешением имен
  + DoS атаки
  + фильтрация сетевых портов

**Модель TCP/IP и инструменты диагностики**

****

1. Анализ и диагностика сетевых проблем должна происходить на основе понимания модели TCP/IP и используемого стека протоколов.
2. Такой анализ позволяет локализовать причины возникшей проблемы и подобрать соответствующие диагностические инструменты.
3. Анализ сетевых проблем следует производить, начиная с нижнего уровня модели TCP/IP и нижних уровней стека протоколов.
4. Уровень сетевого доступа (др. названия: канальный уровень, уровень сетевого интерфейса) описывает среду передачи данных и передачу данных в локальной сети. На этом уровне определяются канальные (MAC) адреса. Протоколы канального уровня: Ethernet, PPP, L2TP, ARP.
5. Межсетевой уровень (др. названия: сетевой уровень) определяет передачу данных между разными сетями. На этом уровне определяются IP адреса. Протоколы межсетевого уровня: IP, ICMP, IGMP, IPsec.
6. Транспортный уровень определяет передачу данных между сетевыми приложениями. На этом уровне определяются адреса сетевых приложений

— порты.

Протоколы транспортного уровня: TCP, UDP.

1. Прикладной уровень описывает работу сетевых приложений.

Протоколы прикладного уровня: DHCP, HTTP, HTTPS, IMAP, LDAP, NTP, PTP, RPC, SMTP, SNMP, SSH и др.

При проведении изменений в настройках следует:

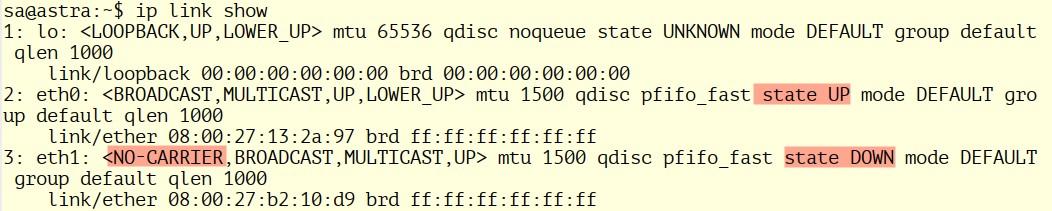
* + сохранять предыдущие конфигурационные файлы;
  + проверять синтаксис проведенных изменений;
  + отслеживать сообщения, поступающие в журнальные файлы, особое внимание уделяя сообщениям об ошибках.

Примечание: для мониторинга поступающих сообщений используются команды:

journalctl -f или tail -f имя\_журнального \_файла

**Диагностика уровня сетевого доступа**

* Проверка сетевого оборудования, сетевых кабелей
* Проверка цвета светодиода на Ethernet сетевой карте
* Выполнить команду ip link show:
  + UP — сетевой интерфейс подсоединен
  + NO-CARRIER — на сетевой интерфейс не приходит сигнал



1. В Astra Linux используется по умолчанию традиционная схема именования сетевых Ethernet интерфейсов: eth0, eth1, … .
2. Команда ip link show имеет следующие полезные варианты использования:
   * ip -br link show — краткая форма представления информации о состоянии интерфейсов;
   * ip -s link show — показывает статистику переданных и полученных пакетов данных для разных интерфейсов.
3. Если команда ip link не показывает никакой информации о всех сетевых интерфейсах (количество установленных сетевых карт больше количества сетевых интерфейсов), то следует:
   * выполнить команду lspci, чтобы определить распознало ли ядро ОС сетевую карту;
   * установить и загрузить драйвер (модуль ядра), соответствующий установленной сетевой карте с помощью команды

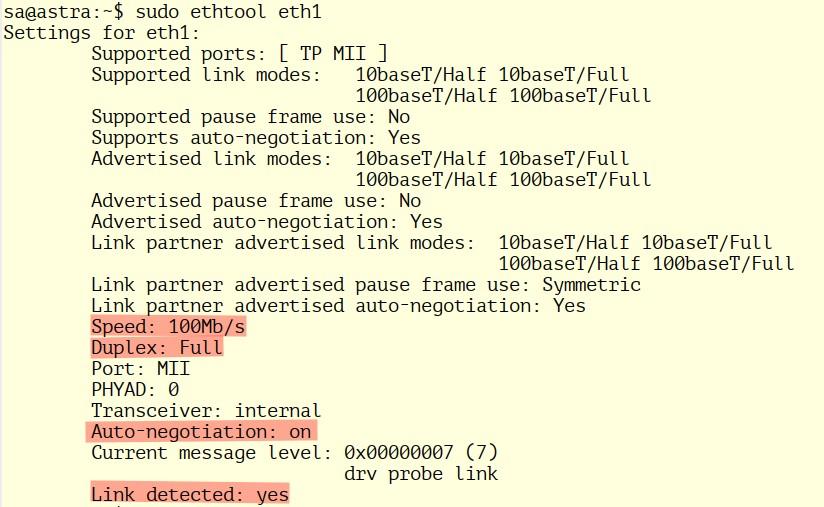
sudo modprobe имя\_драйвера.

1. Сетевые карты Ethernet поддерживают различные режимы передачи данных (скорость передачи данных, полудуплекс/полный дуплекс).
2. Для согласования режимов передачи данных между разными Ethernet устройствами используется механизм самосогласования (autonegotiation).
3. Для просмотра разрешенных и установленных режимов передачи данных сетевых карт Ethernet используются команды ethtool или

mii-tool.

Пример: для сетевого интерфейса eth1 механизм самосогласования включен и выбраны режимы передачи данных: 100Мб/c, полный дуплекс.

1. Утилита ethtool позволяет:



* + получить информацию о драйвере сетевой карты (параметр -i);
  + изменить настройки сетевой карты (параметр -s);
  + получить статистику об использовании сетевого интерфейса, включая информацию об ошибках (параметр -S).

**Сетевые настройки и маршрутизация**

* Проверка сетевых настроек:
  + IP адрес
  + сетевая маска
  + маршрут по умолчанию
  + разрешение имен хостов (DNS)
* Способы настройки сетевых интерфейсов:
  + Network Manager
  + ifup/ifdown команды, файл /etc/network/interfaces
* Динамическое и статическое назначение IP адреса и сетевых настроек

1. С помощью команды

ip address show

следует убедиться, что IP адрес и сетевая маска настроены правильно.

1. Если обнаружены ошибки в значениях IP адреса или сетевой маски, то следует:
   * установить способ настройки сетевых интерфейсов (Network Manager или ifup/ifdown команды);
   * выяснить способ назначения IP адреса (статический или динамический).
2. Если адрес сетевому интерфейсу назначается динамически (через DHCP), то следует проверить доступность сервера DHCP командой:

sudo dhclient -v

1. Для определения способа настройки следует выяснить статус соответствующих служб: NetworkManager.service и networking.service командой:

systemctl status имя\_службы

1. Если активна только служба NetworkManager.service, то исправить сетевые настройки с помощью любой из утилит: nm-connection-editor (графическая утилита), nmtui (псевдографическая утилита), nmcli (утилита командной строки).
2. Если активна только служба networking.service, то исправить сетевые настройки в файле /etc/network/interfaces и проверить содержимое файлов в каталоге /etc/network/interfaces.d.
3. Если активны обе службы, то нужно убедиться, что под управлением Network Manager находятся все интерфейсы кроме интерфейса loopback

Выполните команду

nmcli | grep unmanaged

и обратите внимание, у каких сетевых интерфейсов стоит отметка «без управления» (unmanaged).

1. Если какой-то из сетевых интерфейсов (кроме loopback) находится не под управлением Network Manager, то следует выбрать один из двух вариантов настройки:
   * Отключить службу NetworkManager.service systemctl stop NetworkManager.service systemctl mask NetworkManager.service

и настроить все интерфейсы в файле /etc/network/interfaces.

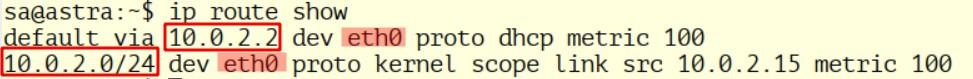
* + Настроить все интерфейсы (кроме loopback) через Network Manager.

1. На любом хосте, подключенном к сети, должен быть настроен маршрут (шлюз) по умолчанию. Проверку параметров такого маршрута (прежде всего, IP адрес шлюза) можно осуществить командой

ip route show.

1. Следует убедиться, что шлюз по умолчанию находится в одной из сетей, непосредственно подключенных к хосту.

Пример: шлюз с адресом 10.0.2.2 находится внутри сети 10.0.2.0/24, к которой хост подключен через интерфейс eth0.



1. Исправить шлюз по умолчанию можно командой:

ip route add default via IP\_адрес\_шлюза

1. Для изменения маршрута по умолчанию следует:
   * использовать утилиты для Network Manager, если интерфейсы управляются этой службой;
   * указать параметр gateway IP\_адрес в файле /etc/network/interfaces.
2. Настройку DNS можно проверить путем просмотра содержимого файла

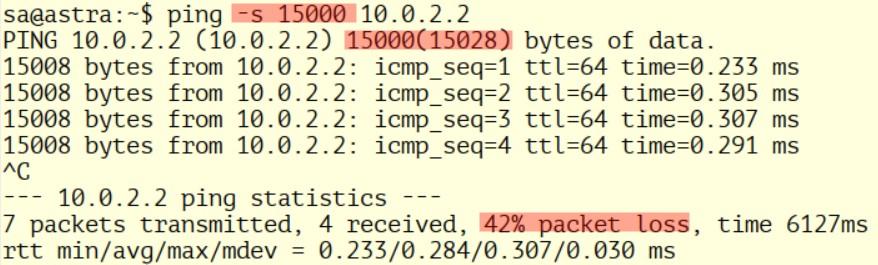
/etc/resolv.conf.

1. Если адреса DNS серверов указаны неверно, то следует:
   * использовать утилиты для Network Manager, если интерфейсы управляются этой службой;
   * исправить адреса серверов DNS в файле /etc/resolv.conf, если интерфейсы настраиваются в /etc/network/interfaces.

**Доступность хостов**

* Проверка доступности хостов по IP адресу или DNS имени:
  + работоспособность TCP/IP стека на локальном хосте
  + доступность шлюза
  + доступность удаленного хоста
* Утилиты ping и traceroute

1. Для проверки доступности хостов используются утилиты ping и traceroute.
2. Утилита ping используется, главным образом, для проверки доступности хостов в локальной сети (сети предприятия), поскольку принято блокировать ICMP сообщения на периметре сети предприятия.
3. Утилита ping может также использоваться для оценки качества линии связи.
4. Порядок использования команды ping:
   * ping localhost или ping 127.0.0.1 - проверка работоспособности локального TCP/IP стека;
   * ping IP\_адреса\_шлюза - проверка доступности хоста в локальной сети и шлюза по умолчанию.
5. Для оценки качества линии связи с помощью команды ping следует использовать ICMP-сообщения большого размера (параметр -s) и обратить внимание на количество потерянных пакетов в сводке, которая приводится в конце вывода команды.



1. Утилита traceroute используется в основном для проверки доступности хостов, находящихся вне локальной сети, и определения возможных проблем с маршрутизацией за пределами сети предприятия.

Изображение выглядит как текст, Шрифт, документ

Автоматически созданное описание

1. Утилита traceroute по умолчанию использует протокол UDP и потому пакеты, отсылаемые данной утилитой, реже отфильтровываются внешними маршрутизаторами.

**Доступность сетевых служб удаленных хостов**

* Протоколы TCP и UDP:
  + порты
  + сокеты
* Сканирование портов удаленных хостов:
  + nc
  + nmap

1. Для адресации сетевых процесс на хосте используются номера портов. Номер порта могут принимать значения от 0 до 65535.
2. Для серверных сетевых процессов порты выделяются статически, а порты для клиентских серверных процессов - динамически из диапазона, определенного в параметре ядра /proc/sys/net/ipv4/ip\_local\_port\_range (по умолчанию 32768-60999).
3. Порты определяются на транспортном уровне TCP/IP модели (протоколы TCP и UDP).
4. Сетевой сокет - это программный интерфейс, предназначенный для обмена данными между процессами и который определяется IP адресом хоста

и портом, соответствующим процессу на данном хосте.

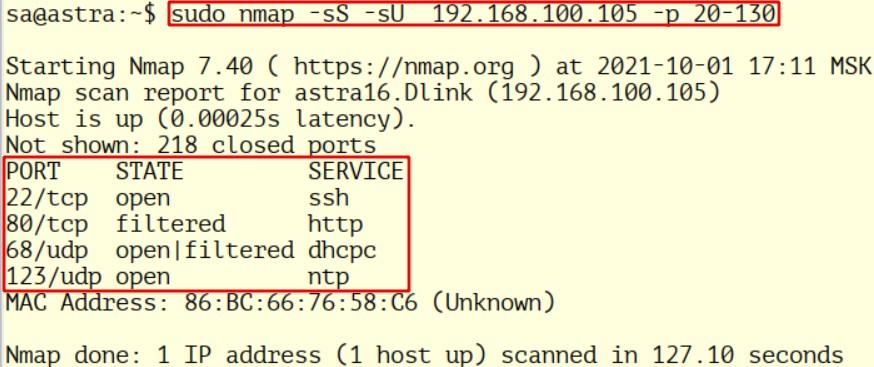
1. Сетевое сокетное соединение - это связь двух сокетов: сокета отправителя (клиента) и сокета получателя (сервера).
2. Серверные сокеты в начале находится в состояние «прослушивание» (listening), ожидая, когда клиентский сокет отправит запрос.
3. Доступность удаленных портов (удаленных сетевых служб) может быть осуществлено командами:
   * nc (программный пакет ncat из smolensk-devel);
   * nmap (программный пакет nmap из smolensk-devel).
4. Утилита nc имеет ограниченную функциональность по сканированию портов:
   * проверка доступности удаленной службы:

nc адрес\_хоста порт

* + сканирование портов:

nc -zv адрес\_хоста диапазон\_портов

1. Утилита nmap предназначена для сканирования портов и поддерживает разные режимы сканирования:
   * -sT - пытается установить TCP соединение (выполнить процедуру тройного рукопожатия);
   * -sS - отправляется пакет с установленным флагом SYN (используется по умолчанию, самый популярный режим сканирования);
   * -sA - отправляется пакет с установленным флагом ACK, не может определить открытые порты, но хорошо умеет определять порты, отфильтрованные межсетевыми экранами.
   * -sU - сканируются UDP порты.

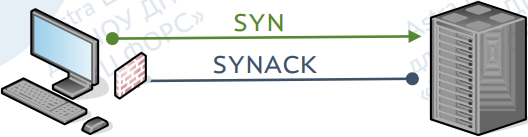


1. Утилита nmap показывает следующие состояния портов:
   * open - процесс «слушает» на этом порту, ожидая запрос от клиента;
   * filtered - порт заблокирован межсетевым экраном;
   * closed - процесс не «слушает» в данный момент, но может возобновить свою работу;
   * unfiltered - nmap не может определить, находится ли порт в состоянии open или closed.

**Обнаружение и устранение неисправностей в работе сетевых служб**

* Проверка работы локальных сетевых служб:
  + netstat -tlnp или ss -tlnp
  + journalctl -u имя\_службы.service
* Обнаружение DoS атак:
  + SYN Flood атака

watch "netstat -tan | grep -i syn | wc -l"

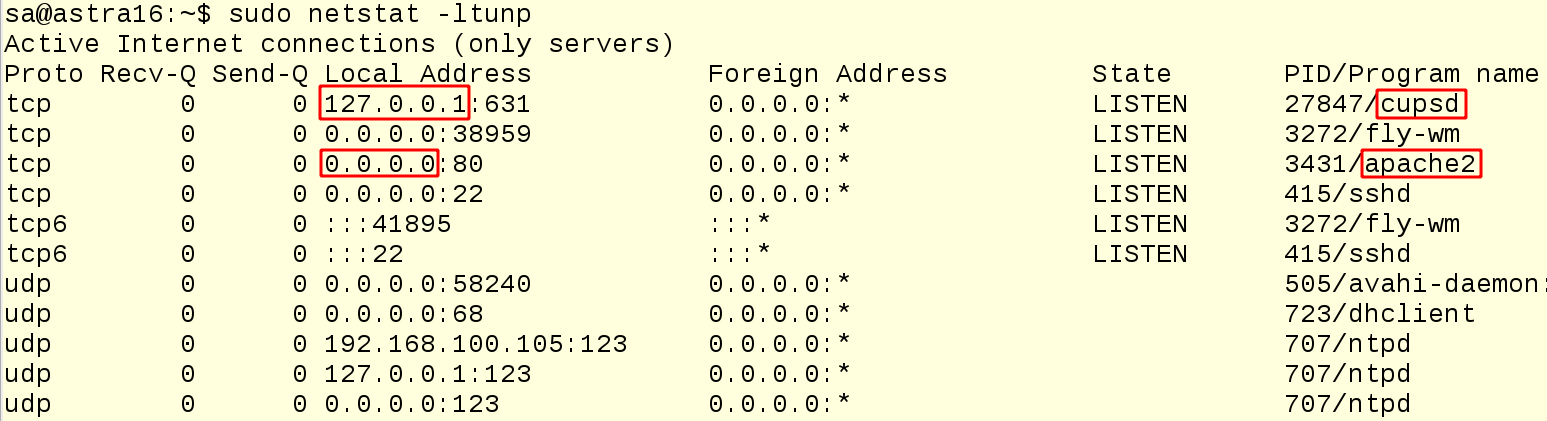


1. После запуска сетевой службы требуется убедиться, что сетевой процесс стартовал и процессу назначен соответствующий порт (будет показана информация о TCP (параметр -t) и UDP (параметр -u) портах и процессах (параметр -p), «слушающих» (параметр -l) на этих портах):

sudo netstat -nltup или sudo ss -nltup

1. При просмотре вывода команды netstat или ss следует обратить внимание, на каком интерфейсе была запущена служба:
   * если указан адрес 127.0.0.1, то служба будет доступна только локально;
   * если указан адрес 0.0.0.0, то служба будет работать на всех сетевых интерфейсах хоста.

Пример: служба печати CUPS - локальная, веб-сервер Apache запущен на всех интерфейсах.



1. В случае, если сетевая служба не стартовала или не использует требуемый сетевой интерфейс, следует проанализировать сообщения из журнала: journalctl -u имя\_юнита\_службы.service
2. Одним из распространенных способов хакерских атак на сетевые службы является, так называемая, DoS атака (Denial of Service, отказ

в обслуживании).

1. DoS атака, в частности, может быть реализована с помощью отправки

с различных поддельных хостов большого количества пакетов, в которых установлен флаг SYN, - начальная стадия установления TCP-соединения (SYN Flood атака).

1. Обработка большого количества SYN-пакетов и существование большого количества полуоткрытых соединений могут существенно нагрузить вычислительные ресурсы сервера (процессор, память), что будет приводить к замедлению работы сервера, вплоть до состояния невозможности дальнейшего обслуживания запросов клиентов.
2. Количество «полуоткрытых» соединений можно посчитать командой:

netstat -tan | grep -i syn | wc -l

Большое количество таких соединений может означать, что идет «SYN Flood» атака.

**Поиск и решение проблем на прикладном уровне**

* Анализ перехваченного сетевого трафика - tcpdump
* Анализ журнальных файлов
* Запуск служб в режиме отладки (debug)
* Диагностика неисправностей с помощью специфических инструментов для разных протоколов прикладного уровня:
  + DNS: nslookup, dig
  + DHCP: dhclient
  + HTTP/HTTPS: curl, wget
  + SMTP: nc адрес\_сервера 25

1. Самым универсальным средством для выявления проблем

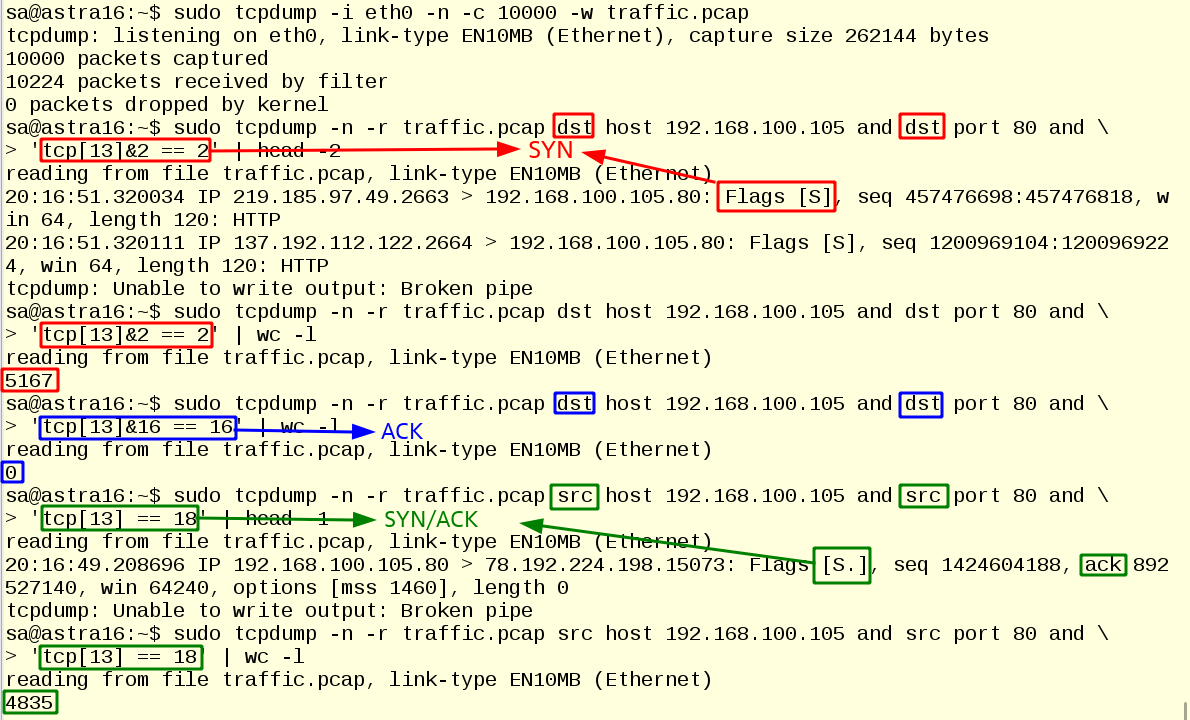
в функционировании сетевых процессов, работающих на прикладном уровне модели TCP/IP является перехват и анализ трафика между клиентом

и сервером.

1. Два наиболее популярных средства для перехвата трафика:
   * tcpdump (программный пакет tcpdump из smolensk-devel);
   * wireshark (программный пакет wireshark из Debian Stretch).
2. Утилита wireshark имеет графический интерфейс и поддерживает интерпретацию данных для огромного количества протоколов (**Analyze→Enabled Protocols**).
3. Основные параметры утилиты tcpdump:
   * -i имя\_интерфейса - указывает интерфейс, на котором будет перехватываться трафик (any - все интерфейсы);
   * -w имя\_файла - сохранение перехваченного трафика в файл;
   * -r имя\_файла - чтение перехваченного трафика из файла;
   * -с число - количество записей о трафике, которые будут перехвачены;
   * -n - не разрешать имена (крайне рекомендуется для постоянного использования);
4. Основные фильтры (BPF, Berkeley Packet Filter) для утилиты tcpdump (man pcap-filter):

* + host - фильтрация по адресу или имени хоста (перед host можно поместить scr или dst, указывая направление трафика);
  + net - фильтрация по номеру сети (перед net можно поместить src или dst);
  + port - фильтрация по номеру или имени порта (имя берется из /etc/services, перед port можно поместить scr или dst);
  + имя протокола: tcp, udp, icmp, arp, ip, ip6;
  + фильтры для заголовков протоколов: имя\_протокола[байт\_заголовка] Примеры: icmp[0] - байт, хранящий тип ICMP сообщения, tcp[13] - флаги TCP.

1. При составлении фильтров можно использовать логические операции and, or, not и группировку условий отбора с помощью круглых скобок.
2. Пример: выполнены следующие действия:
   * перехватывается 10000 записей, которые записываются в файл traffic.pcap;
   * вычисляется количество полученных веб-сервером (192.168.100.105) SYN- пакетов - 5187;
   * вычисляется количество полученных веб-сервером ACK-пакетов - 0;
   * вычисляется количество отправленных веб-сервером SYN/ACK пакетов — 4835.



Анализ перехваченного трафика свидетельствует о SYN Flood атаке на веб- сервер.

**Сетевые утилиты из iproute2 и net-tools**

|  |  |
| --- | --- |
| iproute2 | net-tools |
| ip neigh | arp -na |
| ip link | ifconfig |
| ip addr show | ifconfig -a |
| ip -s link | ifconfig -s или netstat -i |
| ip link set eth0 up/down | ifconfig eth0 up/down |
| Ss | netstat |
| ss -l | netstat -l |
| ip route | netstat -r или route |
| ip route add/del | route add/del |
|  |  |

**Алгоритмы шифрования**

* Симметричные алгоритмы шифрования требуют наличия у абонентов общего секретного ключа
* Асимметричные алгоритмы требуют выпуска каждым абонентом пары ключей, один из которых является приватным и не распространяется

по сети, второй – публичный и может быть передан, в том числе по открытым каналам связи

1. При использовании симметричного алгоритма шифрования применяется общий секретный ключ. С помощью секретного ключа отправителем шифруется исходный текст, который в зашифрованном виде передается получателю, и расшифровывается получателем с использованием этого же секретного ключа.
2. Важнейшая проблема симметричного шифрования: когда один абонент создает секретный ключ, его необходимо передать другому абоненту, так, чтобы он не попал в руки злоумышленника.
3. Для создания общего секретного ключа для симметричного шифрования применяется алгоритм Диффи-Хеллмана.
4. Асимметричный алгоритм шифрования подразумевает использование пару ключей – приватного и публичного. Если текст зашифрован одним ключом пары, расшифровать его можно только с применением второго ключа этой же пары.
5. Приватный ключ никогда не должен покидать того расположения, где он был создан. Публичный ключ может свободно распространяться по открытым каналам связи.
6. Между приватным и публичным ключом существует математическая связь, но подбор приватного ключа по публичному является алгоритмически сложной задачей, время решения которой крайне велико.
7. Асимметричные алгоритмы используются для аутентификации. Абонент, подлинность которого необходимо проверить, создает пару: приватный

и публичный ключ, и передает публичный ключ другому абоненту (по защищенному каналу связи).

1. Когда требуется проверка одного абонента другим, проверяющий абонент создает одноразовую секретную фразу, шифрует ее публичным ключом проверяемого абонента.
2. Если проверяемый абонент смог с помощью приватного ключа расшифровать секретную фразу и предоставить ее проверяющему абоненту, то проверка считается пройденной, а подлинность проверяемого абонента – установленной.

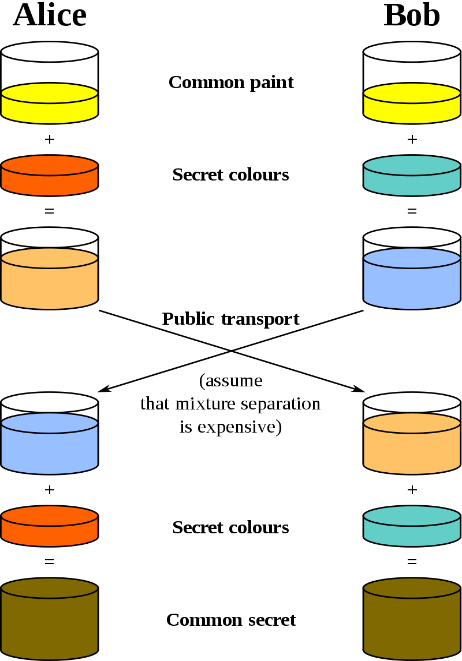
**Алгоритма Диффи-Хеллмана**

* Позволяет при работе по незащищенному от прослушивания каналу связи создать двум участникам общий секретный ключ, не доступный третьей (прослушивающей) стороне, который дальше будет применен

для шифрования с использованием алгоритма симметричного шифрования

* Алгоритм не защищен от атаки «Человек по середине» (man in the middle), поэтому для аутентификации, как правило применяют алгоритм RSA

**Принцип работы алгоритма Диффи-Хеллмана**



1 Проиллюстрируем алгоритм Диффи-Хеллмана с помощью аналогии на красках.

Источник изображения: https://en.wikipedia.org/wiki/Diffie

%E2%80%93Hellman\_key\_exchange#/media/File:Diffie- Hellman\_Key\_Exchange.svg

* Алисе и Бобу известен некий «общий» цвет (желтый);
* Алиса и Боб выбирают каждый свой «секретный» цвет (красный и бирюзовый);
* Алиса и Боб выполняют смешивание «общего» цвета и «секретного» цвета (оранжевый и небесный);
* Алиса и Боб обмениваются результатами вычислений (Алиса получает

небесный цвет, Боб получает оранжевый цвет);

* Алиса и Боб выполняют смешивание полученного цвета и секретного цвета (Алиса – небесный с красным, Боб – оранжевый с бирюзовым);
* Алиса и Боб в результате смешения получают общий секретный цвет – коричневый;
* человек посередине обладая цветами желтым, оранжевым и небесным, смешав их получит другой оттенок цвета, истинный оттенок общего секретного цвета останется неизвестным.

**Служба ssh**

* Предоставляет возможность удаленной работы в командном интерпретаторе, используя аутентификацию и шифрование передаваемых данных
* Позволяет также копировать файлы, используя scp и sftp
* Позволяет осуществлять проброс портов с одной машины на другую
* Наименование службы: **sshd**, наименование клиента: **ssh**
* Запуск/статус/остановка/перезапустить/перечитать конфигурацию/включить/исключить автозагрузку:

systemctl start/status/stop/restart/reload/enable/disable sshd.service

1. Узнать, запущена ли служба sshd: systemctl status sshd.service
2. Запуcк службы ssdh, если она не запущена: systemctl start sshd.service
3. Перечитать конфигурацию службы без ее перезапуска: systemctl reload sshd.service
4. Добавить в автозагрузку (если sshd установлен, но автоматически не стартует после перезагрузки машины):

systemctl enable sshd.service

1. Если служба sshd не была установлена при установке системы, то можно установить с помощью команды:

sudo apt install openssh-server -y

1. Конфигурационный файл для клиента ssh: /etc/ssh/ssh\_config

Справка: man 5 ssh\_config

1. Конфигурационный файл для службы sshd: /etc/ssh/sshd\_config Справка: man 5 sshd\_config
2. ~/.ssh/config – индивидуальные настройки клиента Справка: man 5 ssh\_config

**Некоторые параметры sshd\_config**

* Port 22 – указывает на каком TCP-порту работает служба sshd (по умолчанию 22 порт)
* PasswordAuthentication yes – указывает, что разрешена аутентификация

по паролю. После настройки аутентификации по ключам, следует заменить на no

* X11Forwarding yes – позволяет запускать удаленно графические приложения, отображая графику на клиентских машинах с помощью клиентского X11-сервера. Можно оставить по умолчанию

1. Параметр UsePAM позволяет указать, будет ли служба sshd использовать механизмы аутентификации и поддержки сеансов PAM (по умолчанию используется PAM).
2. Остальные параметры см.: man 5 sshd\_config
3. После внесения изменений в конфигурационный файл необходимо, чтобы служба sshd.service перечитала конфигурационный файл: systemctl reload sshd.service

**Настройка аутентификации по ключам**

### Используется алгоритм асимметричного шифрования RSA

### Пара ключей представлена публичным и приватным ключами

### Ключи на клиенте хранятся в каталоге ~/.ssh в файлах :

### id\_rsa – приватный ключ пользователя

### id\_rsa.pub – публичный ключ пользователя

### known\_hosts – публичные ключи ssh-серверов

### Публичные ключ пользователя c клиента передается на сервер и добавляется в файл ~/.ssh/authorized\_keys

1. Файлы ~/.ssh/authorized\_keys и ~/.ssh/known\_hosts хранят все публичные ключи, которым доверяет либо клиент (known\_hosts) либо сервер (authorized\_keys).
2. Справка по формату authorized\_keys доступна в соответствующем разделе man 8 sshd.
3. Справка по формату known\_hosts содержится в соответствующем разделе

### man 8 sshd.

**Настройка аутентификации по ключам**

### ssh-keyscan – получение публичного ключа сервера, который записывается в known\_hosts

### ssh-keygen – создание пары публичного и приватного ключей

### на клиенте (по умолчанию имя id\_rsa, можно задать иное, парольная фраза – для шифрования приватного ключа)

### ssh-copy-id – копирование публичного ключа клиента на сервер

### ssh-agent – кэширование приватных ключей

### ssh-add – добавление ключей в кэш агента

### Копирование ключа на удаленную машину:

ssh-copy-id username@host

### Запуск ssh-agent:

eval $(ssh-agent)

### После этого выполнить ssh-add, чтобы закэшировать ключ.

### ssh-add попросит вести пароль от ключа. Если будет выдано сообщение

### «Could not open a connection to your authentication agent» значит ssh- agent не запущен, и следует его запустить через eval.

**Основные клиентские команды**

### ssh username@host – для терминального подключения к серверу

### scp somefile username@hostname:/path – копирование файла на удаленную машину

### scp username@hostname:/path localpath – копирование файла на локальную машину

* sshfs username@hostname: ~/mount\_point – монтирование удаленной файловой систем по ssh в локальный (~/mount\_point)
* fusermount -u ~/mount\_point – размонтировать файловую систему, подключенную по ssh

### Если username не указать, будет использовать имя пользователя

### на локальной машине, от имени которого запущена текущая сессия.

### Удаленное подключение ssh по нестандартному порту задается с помощью параметра -p. Например:

ssh -p 55222 username@host

### Для scp нестандартный порт задается с помощью параметра -P. Если порт не задан, то по умолчанию используется TCP-порт 22.

### При необходимости скопировать каталог командой scp следует использовать параметр -r.

### sftp – SSH File Transfer Protocol – протокол, похожий по назначению на ftp, но не являющийся его разновидностью, реализован

### как надстройка над ssh, реализует аутентификацию и шифрование средствами ssh.

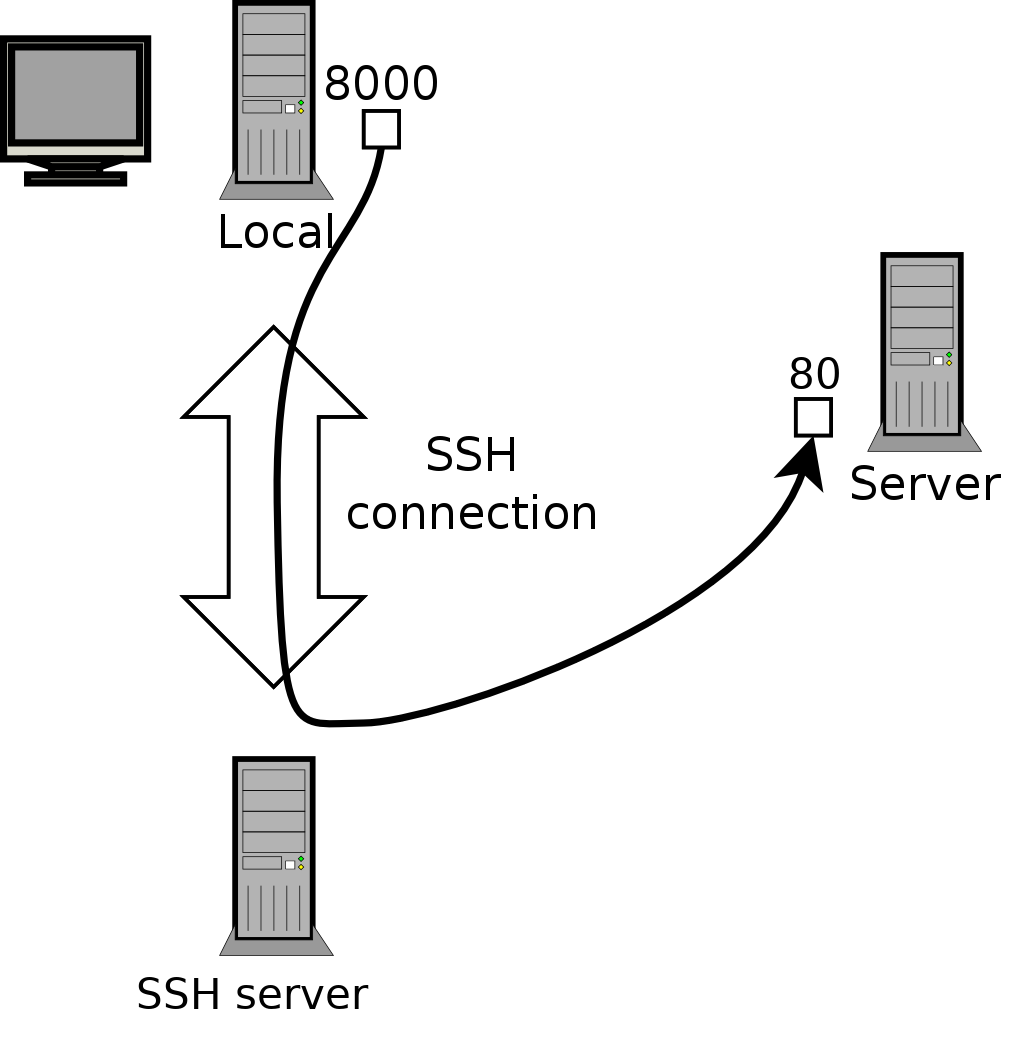
### sftp-сервер находится по адресу /usr/lib/openssh/sftp-server и запускается службой sshd при клиентском подключении по протоколу sftp

### Учитывая, что ftp по умолчанию передает данные в нешифрованном виде, всегда следует использовать scp или sftp.

### Переадресация локального порта

* «Прямой» туннель – проброс порта локального компьютера на порт сервера:

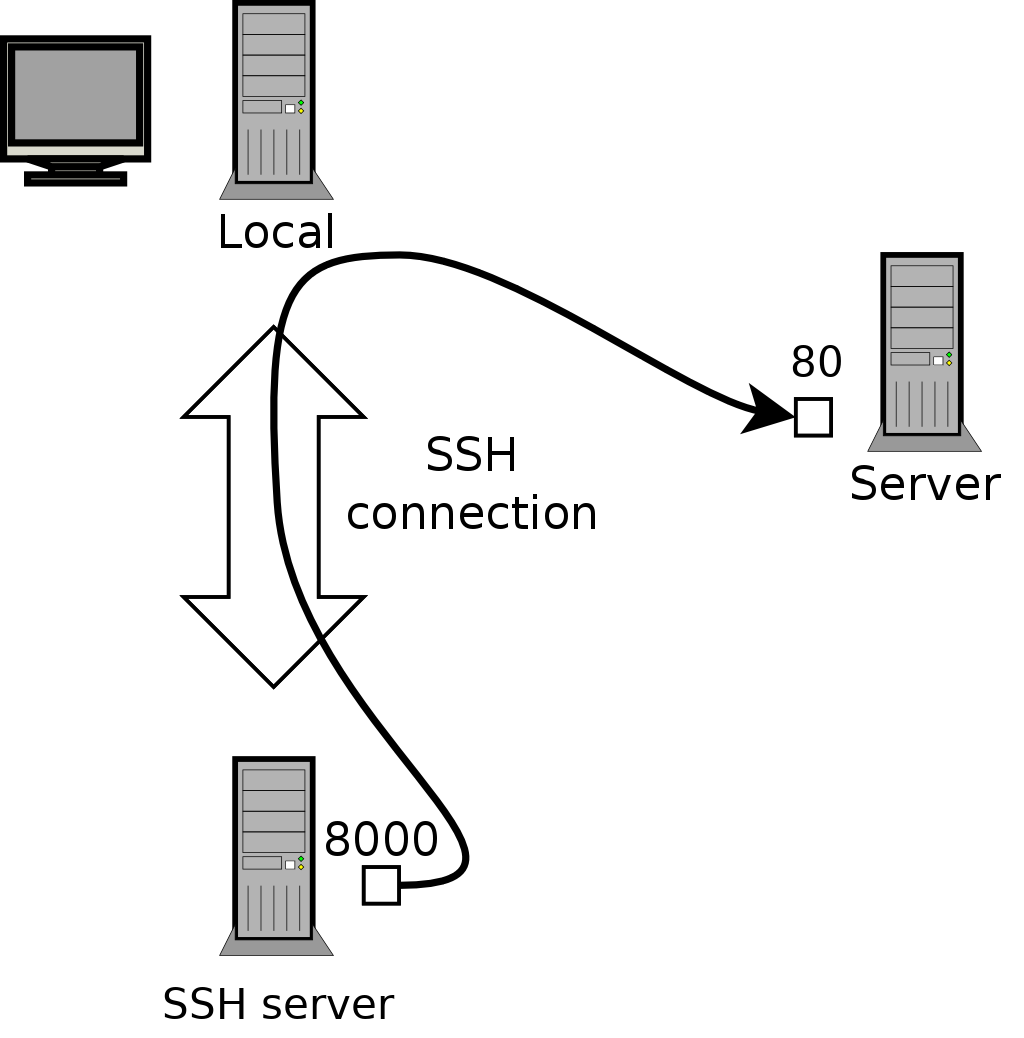
ssh -f -n -L локальный\_порт:адрес\_сервера:порт\_сервера \ user@адрес\_SSH\_сервера



**Переадресация удаленного порта**

* «Обратный» туннель – проброс порта удаленного компьютера на порт сервера:

ssh -f -n -N -R \ удаленный\_порт:адрес\_сервера:порт\_сервера \ user@адрес\_SSH\_сервера



**Туннели в ssh: SOCKS-прокси**

### SSH поддерживает динамическую переадресацию портов, с помощью которой на локальном хосте можно открыть порт прокси-сервера, работающего по протоколам SOCKS4/5

ssh -D localhost:порт user@адрес\_SSH\_сервер

### Пример:

ssh -D 8080 [user@192.168.1.103](mailto:user@192.168.1.103)

### В настройках proxy в операционной системе указать SOCKS-прокси

### 127.0.0.1:8080

**Репозиторий ПО**

Репозиторий содержит:

* данные (deb-файлы) – каталог pool/
* метаданные (описание репозитория и пакетов – каталог dists/)

Репозиторий состоит из компонентов (по типу лицензий):

* main – СПО
* contrib – СПО, но для его работы требуется проприетарное ПО
* non-free – проприетарное ПО

**Настройка доступа к репозиториям**

* Местонахождение репозиториев описывается в файле /etc/apt/sources.list
* Формат записи, описывающий репозиторий:

deb uri\_репо название\_репо названия\_веток

Пример:

deb file:///repo\_dir 1.7\_x86-64 main contrib non-free

(deb – бинарные пакеты, deb-src – пакеты в исходных кодах)

**Подключение стороннего репозитория**

* При подключении стороннего репозитория должен быть установлен публичный ключ, которым подписаны пакеты данного репозитория
* Алгоритм установки публичного ключа:
  + Добавить репозиторий в /etc/apt/sources.list
  + Выполнить команду apt update
  + Если появится сообщение, что ключ не найден, то скачать ключ с публичного сервера и добавить ключ в хранилище ключей:

gpg --keyserver keyserver.ubuntu.com \

--recv-keys идентификатор\_ключа

gpg -a --export идентификатор\_ключа | apt-key add –

**Создание собственного репозитория**

* Для создания репозитория предназначена утилита reprepro (sudo apt install reprepro)
* Алгоритм создания репозитория:
* Создать каталог, в котором будет создан репозиторий
* В подкаталоге conf создать конфигурационный файл distributions
* Создать ключ для репозитория
* Создать структуру репозитория (информация из distributions)

sudo reprepro --ask-passphrase -b имя\_каталога export

* Поместить deb-пакет в репозиторий

sudo reprepro --ask-passphrase -b имя\_каталога \

C имя\_компоненты includedeb кодовое\_имя\_репо имя\_пакета

1. Содержимое файла distributions:

Origin: Debian

Codename: кодовое\_имя\_репо

Suite: stable

Version: 1.0

Architectures: amd64

Components: main contrib non-free

SignWith: yes

1. Процедура создания ключа:

* Выполнить команду sudo gpg --gen-key (на вопрос Real name ответить, например, MyOrg, на вопрос Email address ответить, например, [admin@myorg.test](mailto:admin@myorg.test), задать парольную фразу).
* Для обеспечения нужного количества случайных чисел для генерации ключа на другом терминале выполнить команду sudo dd if=/dev/sda of=/dev/null

**Подключение собственного репозитория**

* На сервере с репозиторием экспортировать публичный ключ репозитория в файл sudo gpg -a --output имя\_файла --export реальное\_имя
* Скопировать ключ на компьютер, на котором будет подключаться репозиторий
* Добавить публичный ключ в хранилище ключей apt sudo apt-key add имя\_файла
* В файл /etc/apt/sources.list добавить строку

deb uri\_репо кодовое\_имя\_репо main contrib non-free

* Обновить кэш репозиториев

sudo apt update

**Практическая работа**

1. Запустите две виртуальные машины – клиент и сервер
2. Проанализируйте файл /etc/network/interfaces. Что содержится в нем?
3. Проверьте соединение между клиентом и сервером
4. Отключите на клиенте службу NetworkManager и настройте сеть, дополнив необходимые конфигурационные файлы. IP адрес машины – 10.10.0.XXX, где XXX – последние 3 цифры в названии машины. Например, у машины ru01wks220 последние цифры – 220. Маска – 255.255.255.0, шлюз по умолчанию – 10.10.0.32
5. Попробуйте отправить ping с сервера на клиент используя доменное имя. Получилось ли это сделать? Если нет, то исправьте это.
6. Подключитесь к ftp серверу
7. Подключите сервер к локальному репозиторию, расположенному по адресу 10.10.0.36
8. Скачайте с локального репозитория утилиту tree и установите ее на ваш сервер.
9. На сервере запустите службу ssh и добавьте ее в автозагрузку
10. На клиенте настройте аутентификацию по ключам с сервером
11. Подключитесь к серверу с машины клиента и создайте в директории /home/study файл с содержимым «Hello world!»
12. Скопируйте с сервера на клиент (командой scp) файл созданный в предыдущем пункте файл.

**Вопросы для проверки**

1. В каких протоколах определяется нумерация (порты) сетевых приложений?
2. В какой протокол встраиваются метки безопасности?
3. Какой диапазон адресов класса A выделен для частных адресов?
4. Удаленный хост доступен, но сетевая служба на удаленном хосте не отвечает. Какое действие следует выполнить, чтобы понять причину, почему нет доступа к удаленной службе?
5. Для сетевых настроек используется служба networking.service.
6. В файл /etc/resolv.conf вносятся изменения, но после перезагрузки системы изменения утеряны. В чем может быть причина?
7. DNS служба не запускается. Какое действие следует выполнить, чтобы понять причину неисправности?
8. Какая команда позволяет узнать настройки сетевых интерфейсов?
9. В каком файле содержатся настройки для команды ifup?
10. Вы изменили настройки сетевого соединения с помощью команды nmcli. Какую команду нужно выполнить, чтобы активировать произведенные изменения?
11. Вы хотите организовать доступ к серверу srv1 с компьютера arm1. Где следует создать пару приватный и публичный ключ?
12. Какой ключ следует скопировать для ssh-доступа на другую машину?
13. Доступ на srv1 осуществляется с 3х arm: arm1, arm2, arm3. Сколько пар ключей должно быть и как их распространять?