

Мрежи

Thursday, August 27, 2020 4:39 PM

Всеки компютър или устройство свързано с мрежа има собствен IP адрес. Той може да бъде IPv4 или IPv6.

IPv4 има 32битов адресно пространство. Което води до факта, че има 4 милиарда адреса.

18 милиона резервирали за лични мрежи.

270 милиона резервиран за мултикаст -> изпращате (данни) през компютърна мрежа до няколко потребители едновременно.

Всеки IPv4 адреси съдържат 4ри пунктирани четворни октети.

Наричат се октети, защото са направени от 8битови числа. 4ри октета правят 32бита.

Разделени са от точки.

Стойността на един октет може да бъде от 0 до 255.

Има няколко обхвата на лични IP адреси.

- 192.168.0.0-192.168.255.255 включват 65 000 IP адреса.
- 172.16.0.0-172.31.255.255 включват 1 милион адреса.
- 10.0.0.0-10.255.255.255 включват към 16 милиона адреса.

Това са лични адреси и не могат да бъдат маршрутизириани през интернет.

Означава също, че тези адреси са безплатни за ползване докато са преведени към публични адреси преди данните да стигнат публичния интернет.

Обикновено правим това със употребата на NAT (Network address translation)

Има още някой блокове които не можем да ползваме за публично мрежарство и те са:

- 169.254.0.0-169.254.255.255 ползва се само за локално свързване, което означава, че се ползват само когато IP адреса трябва да бъде посочен ръчно.

Всеки host има също не маршрутизиращи се адреси наречени "loopback" (обратна връзка)

- 127.0.0.0-127.255.255.255 Тези адреси са само ползвани от нашите компютри и никога маршрутизираны по мрежа.

Всеки адрес пристигнал по loopback трябва да падне (drop), защото хостовете не трябва да получават пакети извън тяхната мрежа.

Адреси завършващи с 0 или 255 са също специални:

- *.0 - дефинират мрежата или подмрежата
- *.255 - за изльчването - Адресът за изльчване позволява на данните да се изпращат до всеки хост в мрежата

В общ линии ние получаваме един IP адрес от нашият интернет доставчик, който бива даден на рутера.

Използваме личните мрежи с цел да се използват от нашите локални машини. Когато един хост иска да комуникира със външният свят неговият адрес бива преведен в публичен. Обратният трафик също бива обратно преведен в личният адрес. Това ни позволява да използваме по малко публични адреси.

С IP версия 4 има четири октета от по 8 бита всеки правейки 32 битово число. В десетично число тези обхвати са от 0.0.0.0 до 255.255.255.255.

0.0.0.0	255.255.255.255
00000000.00000000.00000000.00000000	11111111.11111111.11111111.11111111

Това може да не изглежда по добре, но ще почне да има смисъл когато започнете да разбивате мрежи в подмрежки. Първоначално, мрежовата порция на IP адрес започва с нула като негов бит от по-висок ред и ползва седем бита от първият октет за мрежовият адрес. Това значи, че имаме 7 бита останали за мрежите. Което ни дава 127 мрежи. Останалите мрежови адреси от мрежовото пространство се ползват от хостове. След като имаме 24 бита останали за хостове $32 - (7+1) = 24$. Което сочи че ще имаме около 16 милиона хостове. Това е било окей докато са съществували само няколко компании с много големи мрежи. Но е станало ясно, че ще има нужда от още мрежи и, че те не трябва да са толкова големи.

01111111.11111111.11111111.11111111

01111111.11111111.11111111.11111111

7 bits

01111111.11111111.11111111.11111111

7 bits

01111111.11111111.11111111.11111111


127 networks

24 bits

01111111.11111111.11111111.11111111


16 million hosts

През 1981 адресният обхват е бил разделен в класове. Оригиналната мрежа сега се нарича клас A. Клас В ползва 16 бита от мрежовата порция и 16 бита за хостовете.

10111111.11111111.11111111.11111111

Ползваме 10 като високо редовият бит което ще остави 14 бита за мрежови адреси.

Третият клас е C и ползва 24 битове за мрежовата си порция и 8 бита за хостове.

21 bits

11011111.11111111.11111111.11111111


2,097,152 networks

Ползва 110 за високо редовият си бит което ще остави 21 бита за мрежи и 8 бита за хостове.

Class	Високо редови бит/бинарна стойност	Мрежови бит	Хост бит	Мрежки	Хостове
A	0	8	24	128	16,777,216
B	10	16	16	16,384	65,536
C	110	21	8	2,097,152	256
D	1110	N/A	N/A	N/A	N/A
E	1111	N/A	N/A	N/A	N/A

Имайте в предвид, че най-ниският и най-високият брой в мрежата е с цел излъчване. За това от всички цифри за хостове трябва да извадите 2.

Class D мрежите са резервирани за мулти кастинг. Class E мрежите са дефинирани като експериментални.

Имаме и друга мрежа която се ползва с цел да дефинира колко трябва от адресното пространство е за мрежи и колко е за хостове.

За клас A е 8 бита, ако сложим обаче само единици в тази секция маската ни ще бъде

8 bits

00000000.00000000.00000000.00000000

8 bits

11111111.00000000.00000000.00000000

Или 255.0.0.0

За В ще бъде:

11111111.11111111.00000000.00000000
255 . 255 . 0 . 0

За C:

11111111.11111111.11111111.00000000
255 . 255 . 255 . 0

В някой ситуации няма да искате да държите хостовете си в една и съща мрежа, но имате само една мрежа в тази ситуация може да събнетнете или разделите мрежата си. Това значи, че може да вземете един бит от хост секцията и Да го добавите в мрежовата секция, за да увеличите по две броя на мрежите и да разделите броя на мрежите на две. Изглежда в бинарен изглед ето така:

1 bit 7 bits

11111111.11111111.11111111.10000000

И ще завършва ето така:

255 . 255 . 255 . 128

В тази ситуация ще имаме нова две мрежи която ще е:

192.168.2.0 и ще свършва 192.168.2.127 което значи, че адреса за излъчване е 127 и ще може да ползваме от 1 до 126 за хостове. Втората мрежа ще е от 128 до 255 където 128 ще е мрежовият адрес, а 255 с цел излъчване. Използвамеите IP адреси ще са от 129 до 254.

Подмрежовата маска може да бъде написана в два различни формата: като четири точкова нотация или CIDR.

CIDR е безкласове вътрешен маршрутизиране.

С точковата нотация казваме на мрежата да бъде примерно 255.255.255.0 ако бинарната стойност е 11111111 11111111 11111111 00000000. Всеички хостови имена и портове ще са вътре в тази мрежа. Трябва да има адрес 192.168.2.1 да може нашите устройства да попадат вътре в тази мрежа. Това е първият хост от където се нарича сървър. Нашата система ще има портове на които някой сървъри слуизпълждащо 192.168.100/25 ако обаче добавим 26 накрая ще имаме 4 при мрежи с по 64 хоста. Ако имаме 192.168.2.100 и на него има уеб сървър то трафика след като бъде преведен ще стигне до 192.168.2.100 след това ще оти на 192.168.2.100:80 . :80 е гнездото.

Какво ви трябва за TCP/IP пакети:

- Адрес на източника
- Източник порт
- Адрес на дестинацията
- Порта на дестинацията
- Орган за присвояване на интернет IANA
- 65000 порта

Портове:

- 0-1023 често срещани комуникационни портове.
- 20 и 21 се ползват за FTP
- 22 за SSH
- 23 за telnet
- 25 SMTP mail
- 53 за DNS
- 80 за уеб сървари.

Ако ви е трудно да помните портовете може да ги видите в /etc/services файла:

less /etc/services

```
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, officially ports have two entries
# even if the protocol doesn't support UDP operations.
#
# Updated from https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml .
#
# New ports will be added on request if they have been officially assigned
# by IANA and used in the real-world or are needed by a debian package.
# If you need a huge list of used numbers please install the nmap package.

tcpmux      1/tcp          # TCP port service multiplexer
echo        7/tcp          #
telnet     23/tcp          #
discard    9/tcp          sink null
discard    9/udp          sink null
sysstat    11/tcp          users
daytime    13/tcp          #
datagram   13/udp          #
netstat    15/tcp          #
qotd      17/tcp          quote
chargen   19/tcp          ttymonitor source
chargen   19/udp          ttymonitor source
ftp-data   20/tcp          #
ftp       21/tcp          #
ftps      21/udp          fspd
ssh       22/tcp          # SSH Remote Login Protocol
telnet    23/tcp          #
smtp      25/tcp          mail
time      37/tcp          timeserver
time      37/udp          timeserver
whois     43/tcp          nickname
tacacs    49/tcp          # Login Host Protocol (TACACS)
tacacs    49/udp          #
domain    53/tcp          # Domain Name Server
domain    53/udp          #
bootps   68/udp          #
bootpc   68/tcp          #
tftp      69/udp          #
gopher    70/tcp          # Internet Gopher
finger    79/tcp          #
http     80/tcp          www           # WorldwideWeb HTTP
kerberos  88/tcp          kerberos-sec  # Kerberos v5
kerberos  88/udp          kerberos-sec  # Kerberos v5
iso-tsap  100/tcp         tsap          # ISO TSAP 1000
iso-tpnba 104/tcp         dicom         # digital Imag. & Comm. 300
pop3     110/tcp          pop-3        # POP version 3
sunrpc   111/tcp          portmapper  # RPC 4.0 portmapper
sunrpc   111/udp         portmapper  #
auth     113/tcp          authentication 128 ident
ntp      119/tcp          readnews untp # USENET News Transfer Protocol
ntp      123/udp          # Network Time Protocol
snmp     161/tcp          loc-srv      # DCE/OSF Location
snmp     161/udp          #
netbios-ns 137/tcp         loc-srv      # NETBIOS Name Service
netbios-ns 137/udp         loc-srv      #
netbios-dgm 138/tcp         # NETBIOS Datagram Service
netbios-dgm 138/udp         #
netbios-ssn 139/tcp         # NETBIOS session service
netbios-ssn 139/udp         #
imap2    143/tcp          imap          # Interim Mail Access P 2 and 4
imap2    143/udp          #
smtp     25/tcp          # Simple Mail Protocol
```

TCP

Протоколът за контрол на предаването, или TCP. TCP е част от първоначалната програма за мрежово предаване, която в крайна сметка отстъпва място на Интернет протокола, използван в съвременните мрежи.

TCP се използва широко заради своята надеждност, подреден характер (пакетите се обработват във фиксирана последователност, само при пристигането им) и корекция на грешки. TCP се използва за много неща, като имейл, прехвърляне на файлове и всяка друга операция, при която наредените данни без грешки са по-важни от чистата скорост.

UDP

Протоколът за потребителски дейтаграми или UDP е малко по-различен от това, което можете да очаквате от транспортен протокол. Разлика от TCP, UDP е метод за комуникация без връзка. Това означава, че UDP дейтаграмите могат да се изпращат, без да се установява връзка между две устройства, което им позволява да се изпращат, без да се обмисля скорост или последователност.

За UDP основният фокус е скоростта. Тъй като UDP дейтаграмите се координират от приложението, а не от протокола, те могат да бъдат получени и обработени, когато идват. Това е от решаващо значение за неща като видео потоци или VOIP, където обработката на информация възможно най-бързо е по-важно от повторното сглобяване на данните в перфектен ред.

ICMP

Протоколът за съобщения за контрол на интернет или ICMP има съвсем различна функция от TCP и UDP. За разлика от тези типове, не е традиционен протокол за пакети данни. ICMP е специален тип пакет, използван за комуникация между устройства, носещ всички инструкции за пренасочване до времеви марки за синхронизация между устройствата.

Това, за което ICMP е може би най-известен обаче, е echo исканията. Това е почти така, както звучи. Едно устройство изпраща ICMP до друго, като казва на получателя да изпрати отговор, потвърждаващ, че е получил заявката. След това получателят отговаря с нов ICMP пакет, отговорът за echo, потвърждавайки заявката.

IPv6

По-ефективно маршрутизиране

IPv6 намалява размера на маршрутни таблици и прави маршрутизирането по-ефективно и йерархично. IPv6 позволява на доставчици на интернет услуги да обединяват префиксите на мрежите на своите клиенти в един префикс и да обявяват този един префикс в IPv6 интернет. Освен това в мрежите IPv6 фрагментацията се обработва от устройството източник, а не от рутера, като се използва протокол за откриване на максималната единица за предаване на пътя (MTU).

По-ефективна обработка на пакети

Опростената заглавка на пакета на IPv6 прави обработката на пакети по-ефективна. В сравнение с IPv4, IPv6 не съдържа контролна сума на ниво IP, така че контролната сума не трябва да се преизчислява при всеки хоп на рутера. Да се отървем от контролната сума на IP беше възможно, тъй като повечето технологии на ниво връзка вече съдържат контролна сума и възможности за контрол на грешки. В допълнение, повечето транспортни слоеве, които обработват свързване от край до край, имат контролна сума, която позволява откриване на грешки.

Насочени потоци от данни

IPv6 поддържа мултиicast, а не излъчване. Multicast позволява пакетните потоци с интензивна честотна лента (като мултимедийни потоци) да бъдат изпращани едновременно до множество дестинации, спестявайки мрежовата честотна лента. Незainteresовани хостове вече не трябва да обработват пакети за излъчване. В допълнение, заглавката на IPv6 има ново поле, наречено Flow Label, което може да идентифицира пакети, принадлежащи към същия поток.

Опростена мрежова конфигурация

Автоматичната конфигурация на адресите (присвояване на адреси) е вградена в IPv6. Рутерът ще изпрати префикса на локалната връзка в своите реклами на рутера. Хостът може да генерира свой собствен IP адрес, като добавя своя адрес на ниво връзка (MAC), преобразуван в 64-битов формат Extended Universal Identifier (EUI), към 64 бита на префикса на локалната връзка.

Поддръжка за нови услуги

Чрез премахване на преобразуването на мрежови адреси (NAT) се възстановява истинската свързаност от край до край на IP слоя, дава възможност за нови и ценни услуги. Мрежите peer-to-peer са по-лесни за създаване и поддръжка, а услуги като VoIP и Качество на услугата (QoS) стават по-стабилни.

Сигурност

IPSec, който осигурява поверителност, удостоверяване и целостта на данните, е включен в IPv6. Поради техния потенциал да прена злонамерен софтуер, IPv4 ICMP пакетите често се блокират от корпоративни защитни стени, но ICMPv6, внедряването на протокола интернет съобщения за контрол за IPv6, може да бъде разрешено, тъй като IPSec може да бъде приложен към ICMPv6 пакетите.

Ifconfig

ip addr

ip addr show <eth>

Имаме 3 типа hostname:

Static: **/etc/hostname**

Tasient: динамично ползвано от ядрото може да бъде променено от DHCP или MDNS

Pretty: Текстовият формат показан на потребителя

Настройване на мрежовата карта:

Чете се от **/etc/sysconfig/network-scripts/ifcfg-name**

DHCP:

```
DEVICE=name
BOOTPROTO=dhcp
ONBOOT=yes
```

STATIC:

```
DEVICE=name
BOOTPROTO=dhcp
ONBOOT=yes
PREFIX=24
IPADDR=10.0.1.27
IPV6ADDR=address
HWADDR=MAC-address
```

====

IPV6INIT=answer дали да бъде пуснато

====

DNS config == **/etc/resolv.conf**

DNS{1,2}=address

PEERDNS=answer (yes,no)

=====

Каквато и да е промяна да бъде направена в мрежовите настройки за да се запази трябва да направите: nmcli с reload.

Може да променяте настройките с команда `nmcli`:



nmcli

Списък с наличните Wi-Fi AP точки

nmcli device wifi list

Показване на обща информация и свойства за Wi-Fi интерфейс:

nmcli -p -f general,wifi-properties device show wlan0

Изброяване на разрешения за полкит на NetworkManager

nmcli general permissions

Изброяване на ниво журнали и домейни на NetworkManager

nmcli general logging

man nmcli-examples

Можем да управляваме мрежите със systemd също:

Конфигурационните файлове са в:

/usr/lib/systemd/network

/run/systemd/network

/etc/systemd/network

Завършват с:

.network

Инсталира се с:

sudo yum install -y systemd-networkd

Основно отстраняване на неизправности в мрежата:

Маршрутизацията е процес, който позволява на една система да намери мрежовия път към друга.

Маршрутизацията често се обработва от устройства в мрежата,
посветен на маршрутизиране на трафика, наречени рутери.

Въпреки това всеки компютър или програмираме мрежово устройство може да бъде в състояние да действа като рутер.

Обикновено хостът не трябва да има дефинирана сложна маршрутизация, тъй като има правила за това къде трафикът отива в зависимост от това откъде идва.

Сценарий първи, ако хостовете източник и дестинация са в една и съща физическа мрежа, трафикът се препраща директно към дестинацията, обикновено от мрежовия превключвател.

Сценарий втори, ако хостовете източник и местоназначение не са в една и съща физическа мрежа, всички дефинирани в таблицата маршрутизиране се опитват един след друг.

Тези маршрути могат да бъдат динамични маршрути, добавени от нашия DHCP сървър, или могат да бъдат статични маршрути, които сме добавили ръчно.

Ако бъде намерен правилен маршрут, пакетът се препраща.

Сценарий три, ако не бъде намерен подходящ маршрут, тогава пакетът се изпраща към шлюза по подразбиране, който ще бъде конфигуриран или динамично чрез DHCP, или ръчно чрез системния администратор.

След това шлюзът по подразбиране се опитва да намери подходящ маршрут за пакета и го изпраща нататък. Ако не може, ще го изпрати до шлюза по подразбиране

За да прегледате вашата таблица за маршрутизация, можете да влезете във вашата VM. В терминала въведете **ip route** и натиснете enter.

```
default via 192.168.10.1 dev enp8s0 proto dhcp metric 100
169.254.0.0/16 dev enp8s0 scope link metric 1000
192.168.10.0/24 dev enp8s0 proto kernel scope link src 192.168.10.219 metric 100
```

Нека видим резултата. Лявата колона е целевият мрежов адрес с маска на подмрежата.

Думата по подразбиране означава адреса на шлюза по подразбиране. Къде се изпраща трафик, ако няма свободен маршрут.

Вторият раздел обхваща две думи, **dev** и името на устройството за мрежата.

Това може да е физическо или виртуално мрежово устройство.

Третият раздел също обхваща две думи,

proto, последвано от протокола. Протоколът на ядрото казва, че ядрото е инсталирало маршрута по време на автоконфигуриране.

Тези протоколи са дефинирани във файла **/etc/iproute2/rt_protos**.

Ако този раздел казва „прото статичен“, това означава, че маршрутът е зададен от администратора и отменя динамичното маршрутизиране.

Следващият раздел определя обхвата на дестинацията. Файлът **/etc/iproute2/rt_scopes** ги определя.

Следващият раздел е за адреса на източника. За много редове това ще бъде последната част от данните. Възможно е обаче да има друг раздел, показващ метриката, това би показало разходите за използване на маршрута или броя на скачанията. Локалните мрежи биха били един хоп, а отдалечените мрежи повече.

Можем да добавяме статични маршрути с помощта на командата **ip route**.

Можем също да добавяме статични маршрути, като използваме старата команда **route**, въпреки че първо трябва да бъде инсталиран. Ако обаче искате вашите статични маршрути да оцелеят при рестартиране, ще трябва да ги добавите към конфигурационни файлове **/etc/sysconfig/network-scripts**.

Когато създаваме конфигурационен файл на мрежовия интерфейс, ние го именуваме **ifcfg-име** на интерфейс, като **ifcfg-eth0**. За стап маршрути ние именуваме името на интерфейса на маршрута на конфигурационния файл **route-eth0**.

Например, ако искахме да добавим статичен маршрут за интерфейса eth0, щяхме да създадем файл, наречен route-eth0. За да доби маршрут ръчно с помощта на команда ip бихме въвели **ip route add 192.168.100.0/24**, който е маската на подмрежата, **192.168.10** който е шлюзът, **dev team0**, който е интерфейсът.

```
ip route add 192.168.100.0/24 192.168.100.1 dev team0
```

Ако въведете **ip route**, ще видите резултатите от този ред.

```
default via 192.168.122.1 dev eth0 proto dhcp metric 100
192.168.100.0/24 dev team0 proto kernel scope link src 192.168.100.100 metric 350
192.168.122.0/24 dev eth0 proto kernel scope link src 192.168.122.112 metric 100
192.168.124.0/24 dev virbr0 proto kernel scope link src 192.168.124.1
```

Предшественият тип на тази команда е малко по-дълъг.

Ще напишем **route add -net 192.168.100.0 netmask 255.255.255.0 gw 192.168.100.1 dev team0**

Ако обаче рестартирате, след като зададете статичния маршрут с някоя от тези команди, той ще изчезне. За да го направим постоянно ще искаме да го добавим към конфигурационен файл. Като такива бихме създали файл **route-team0** в **/etc/sysconfig/network-scripts**. Като такива бихме създали файл **route-team0** в **/etc/sysconfig/network-scripts**. Вътре в този файл бихме добавили:

```
ADDRESS0=192.168.100.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.100.1
```

Името на устройството се извлича от самото име на файла.

Това би било еквивалентно на пример за ръчен IP маршрут по-рано, но ще оцелее при рестартиране

Много често се отстраняват проблеми с мрежата. Можем да конфигурираме мрежата според информацията, която имаме, и ако тя работи, трябва да разберем защо.

ethtool enp8s0

```
nick@xmachine:/etc/netplan$ ethtool enp8s0
Settings for enp8s0:
  Supported ports: [ TP MII ]
  Supported link modes:  10baseT/Half 10baseT/Full
                         100baseT/Half 100baseT/Full
                         1000baseT/Full
  Supported pause frame use: Symmetric Receive-only
  Supports auto-negotiation: Yes
  Supported FEC modes: Not reported
  Advertised link modes:  10baseT/Half 10baseT/Full
                         100baseT/Half 100baseT/Full
                         1000baseT/Full
  Advertised pause frame use: Symmetric Receive-only
  Advertised auto-negotiation: Yes
  Advertised FEC modes: Not reported
  Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                         100baseT/Half 100baseT/Full
                                         1000baseT/Full
  Link partner advertised pause frame use: No
  Link partner advertised auto-negotiation: Yes
  Link partner advertised FEC modes: Not reported
  Speed: 1000Mb/s
  Duplex: Full
  Port: MII
  PHYAD: 0
  Transceiver: internal
  Auto-negotiation: on
Cannot get wake-on-lan settings: Operation not permitted
  Current message level: 0x00000033 (51)
                           drv probe ifdown ifup
  Link detected: yes
```

Ако скороста е налична и duplex е фул всичко е наред.

```
ip addr show
ifconfig
ping 192.168.10.219
```

arp ще ни покаже MAC адреса и корелацията на IP адреса в нашата мрежа

```
nick@xmachine:/etc/netplan$ arp
Address      HWtype  HWaddress          Flags Mask     Iface
192.168.10.52   ether   3c:2a:f4:ad:eb:dc  C        enp8s0
192.168.10.254  ether   3c:2a:f4:e7:ca:92  C        enp8s0
192.168.10.235  ether   70:85:c2:d0:f0:0e  C        enp8s0
192.168.10.69   ether   08:00:27:55:c8:76  C        enp8s0
_gateway       ether   74:4d:28:9b:8e:4a  C        enp8s0
192.168.10.62   ether   (incomplete)           C        enp8s0
nick@xmachine:/etc/netplan$ arp -n
Address      HWtype  HWaddress          Flags Mask     Iface
192.168.10.52   ether   3c:2a:f4:ad:eb:dc  C        enp8s0
192.168.10.254  ether   3c:2a:f4:e7:ca:92  C        enp8s0
192.168.10.235  ether   70:85:c2:d0:f0:0e  C        enp8s0
192.168.10.69   ether   08:00:27:55:c8:76  C        enp8s0
192.168.10.1    ether   74:4d:28:9b:8e:4a  C        enp8s0
192.168.10.62   ether   (incomplete)           C        enp8s0
nick@xmachine:/etc/netplan$ ip route
default via 192.168.10.1 dev enp8s0 proto dhcp metric 100
169.254.0.0/16 dev enp8s0 scope link metric 1000
192.168.10.0/24 dev enp8s0 proto kernel scope link src 192.168.10.219 metric 100
nick@xmachine:/etc/netplan$ route -n
Kernel IP routing table
Destination  Gateway      Genmask      Flags Metric Ref  Use Iface
0.0.0.0      192.168.10.1  0.0.0.0      UG 100    0      0 enp8s0
169.254.0.0  0.0.0.0      255.255.0.0  U 1000   0      0 enp8s0
192.168.10.0 0.0.0.0      255.255.255.0 U 100    0      0 enp8s0
nick@xmachine:/etc/netplan$ netstat -r
Kernel IP routing table
Destination  Gateway      Genmask      Flags  MSS Window irtt Iface
default      _gateway    0.0.0.0      UG     0 0          0 enp8s0
link-local   0.0.0.0      255.255.0.0  U      0 0          0 enp8s0
192.168.10.0 0.0.0.0      255.255.255.0 U      0 0          0 enp8s0
```

ping <default_gateway>

ping 8.8.8.8

Traceroute е инструмент, който измерва транзитни закъснения на пакети през IP мрежа

```
nick@xmachine:/etc/netplan$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
1  _gateway (192.168.10.1)  0.214 ms  0.202 ms  0.214 ms
2  78.83.185.233 (78.83.185.233)  0.315 ms  0.310 ms  0.388 ms
3  * * *
4  92.247.143.226 (92.247.143.226)  0.588 ms  7.240 ms  0.623 ms
5  108.170.250.161 (108.170.250.161)  0.613 ms  0.628 ms  0.548 ms
6  72.14.237.137 (72.14.237.137)  1.339 ms  74.125.251.185 (74.125.251.185)  1.596 ms  108.170.238.171 (108.170.238.171)  0.621 ms
7  dns.google (8.8.8.8)  0.500 ms  0.520 ms  0.507 ms
```

Можем да видим че има 7 скока между нашата мрежа и проследената.

cat /etc/resolv.conf

dig @IPPERSONALDNS domainname Насилваме да се установи връзка със даден уеб сървър.

```
nick@xmachine:/etc/netplan$ dig @127.0.0.53 linkedin.com
; <>> Dig 9.16.1-Ubuntu <>> @127.0.0.53 linkedin.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<< opcode: QUERY, status: NOERROR, id: 44877
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
linkedin.com.           IN      A
;; ANSWER SECTION:
linkedin.com.          3237   IN      A      108.174.10.10

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sat Oct 03 08:56:09 EEST 2020
;; MSG SIZE  rcvd: 57
```

dig показва IP на домейна

```
nick@xmachine:/etc/netplan$ dig redhat.com
; <>> DiG 9.16.1-Ubuntu <>> redhat.com
; global options: +cmd
; Got answer:
; >>>HEADER<<- opcode: QUERY, status: NOERROR, id: 33728
; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
; QUESTION SECTION:
;redhat.com.           IN      A
;
; ANSWER SECTION:
redhat.com.      3350    IN      A      209.132.183.105
;
; Query time: 24 msec
; SERVER: 127.0.0.53#53(127.0.0.53)
; WHEN: Sat Oct 03 08:57:31 EEST 2020
; MSG SIZE rcvd: 55

nick@xmachine:/etc/netplan$ dig redhat.com +nocomments +noauthority +noadditional +nostats
; <>> DiG 9.16.1-Ubuntu <>> redhat.com +nocomments +noauthority +noadditional +nostats
; global options: +cmd
;redhat.com.          IN      A
redhat.com.      3272    IN      A      209.132.183.105
```

Опция без излишен текст

```
nick@xmachine:/etc/netplan$ dig redhat.com +noall +answer
redhat.com.      3195    IN      A      209.132.183.105
```

Взимане на мейл сървър IP

```
nick@xmachine:/etc/netplan$ dig redhat.com MX +noall +answer
redhat.com.      597    IN      MX      10 us-smtp-inbound-2.mimecast.com.
redhat.com.      597    IN      MX      10 us-smtp-inbound-1.mimecast.com.
```

Всички IPта

```
nick@xmachine:/etc/netplan$ dig redhat.com ANY +noall +answer
redhat.com.      3599    IN      NS      a9-65.akam.net.
redhat.com.      3599    IN      NS      a13-66.akam.net.
redhat.com.      3599    IN      NS      a1-68.akam.net.
redhat.com.      3599    IN      NS      a16-67.akam.net.
redhat.com.      3599    IN      NS      a28-64.akam.net.
redhat.com.      3599    IN      NS      a10-65.akam.net.
redhat.com.      3599    IN      TXT     "docusign=fd355fc-11f9-4eaf-8ecf-64433ef46173"
redhat.com.      3599    IN      TXT     "amazonses:ablaZDac37yeQuCZAZjbFqrELxUCC+8pBdvhFEPtSly="
redhat.com.      3599    IN      TXT     "v=spf1 ip4:103.23.64.2 ip4:103.23.65.2 ip4:103.23.66.26 ip4:103.23.67.26 ip4:107.21.15.141 ip4:108.17.8.0/21 ip4:13.111.0.0/16 ip4:136.147.128.0/20 ip4:136.147.176.0/20 ip4:148.105.8.0/21 ip4:148.139.0.2 ip4:148.139.1.2 ip4:148.139.2.2 ip4:148.139.28.2 i" ip4:148.139.29.2 ip4:148.139.3.2 ip4:149.72.0.0/16 ip4:149.96.1.26 ip4:149.96.13.2 include:spf1.redhat.com -all"
redhat.com.      3599    IN      TXT     "adobe-idp-site-verification=10154eb7d4abc67e9e45621e46476febbec28a97a4610d7c043c42c667aa18d4"
redhat.com.      3599    IN      TXT     "docusign=c6aa79da-fde1-4b7e-8874-28eeb223ca63"
redhat.com.      3599    IN      TXT     "MS=m88428189"
redhat.com.      3599    IN      TXT     "status-page-domain-verification=hyls0f05cd87"
redhat.com.      3599    IN      TXT     "status-page-domain-verification=dfx5rbys1ts5"
redhat.com.      3599    IN      TXT     "atlassian-domain-verification=fHiTv781WbOHg16U1McyXa9JUSS05B0ECvgSzZJ9+b4q8wv0Tf4iI75xdcyoC00"
redhat.com.      3599    IN      TXT     "miro-verification=@bc02d4257d450b9f9034363a58f88b40904dc22"
redhat.com.      3599    IN      SOA     a1-68.akam.net. noc.redhat.com. 2020100200 300 180 604800 14400
redhat.com.      599    IN      MX      10 us-smtp-inbound-2.mimecast.com.
redhat.com.      599    IN      MX      10 us-smtp-inbound-1.mimecast.com.
redhat.com.      3599    IN      A      209.132.183.105
```

Накратко

```
nick@xmachine:/etc/netplan$ dig redhat.com +short
209.132.183.105
```

С DNS сървър с цел грабване на nameservers

```
nick@xmachine:/etc/netplan$ dig @ns1.redhat.com redhat.com
; <>> DiG 9.16.1-Ubuntu <>> @ns1.redhat.com redhat.com
; (1 server found)
; global options: +cmd
; Got answer:
; >>>HEADER<<- opcode: QUERY, status: NOERROR, id: 8340
; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 6, ADDITIONAL: 1
; WARNING: recursion requested but not available
;
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; QUESTION SECTION:
;redhat.com.           IN      A
;
; ANSWER SECTION:
redhat.com.      3600    IN      A      209.132.183.105
;
; AUTHORITY SECTION:
redhat.com.      3600    IN      NS      a13-66.akam.net.
redhat.com.      3600    IN      NS      a16-67.akam.net.
redhat.com.      3600    IN      NS      a28-64.akam.net.
redhat.com.      3600    IN      NS      a1-68.akam.net.
redhat.com.      3600    IN      NS      a10-65.akam.net.
redhat.com.      3600    IN      NS      a9-65.akam.net.
;
; Query time: 40 msec
; SERVER: 209.132.186.218#53(209.132.186.218)
; WHEN: Sat Oct 03 09:06:16 EEST 2020
```

```
;; MSG SIZE  rcvd: 187
```

```
nick@xmachine:/etc/netplan$ dig -x 209.132.183.105 +short
redirect.redhat.com.
```

Работа с файл който съдържа домейни.

```
nick@xmachine:~$ dig -f dnsnames.txt +noall +answer
centos.com.      3599   IN    A     217.160.0.230
redhat.com.      3202   IN    A     209.132.183.105
ford.com.        1305   IN    A     136.2.65.37
ford.com.        1305   IN    A     136.2.49.37
google.com.       238    IN    A     172.217.169.174
```

Има два файла за разрешаване на имена на клиент, с които трябва да се занимаваме в Enterprise Linux. **resolv.conf** и **nsswitch.conf**.

Целта на **resolv.conf** е да предостави списък с глобален DNS сървър, който нашият клиент да запитва при разрешаване на DNS имена.

Има две колони:

Първият ще има или #search, #domain, #nameserver, #sortlist, #options.

/etc/resolv.conf

```
search localnet.com ford.com apple.com
domain otherdomain.com
nameserver 4.2.2.2
nameserver 8.8.8.8
nameserver 192.168.1.21
sortlist 130.155.160.0/255.255.240.0
options rotate
options timeout:5
options attempts:2
```

Ако в първата колона е посочен домейн, той ще го закрепи към името на кратко име. Обикновено това е нашето местно име на дом.

Можем да го променим.

Ключовата дума, която използваме най-много, е сървър за имена.

Сървърът на имена е последван от IP адрес на DNS сървър за заявка.

Например, сървър за имена 4.2.2.2. Можете да имате до три реда за сървъри на имена.

Ключовата дума sortlist ни позволява да сортираме IP адреси в определени мрежи.

Например, ако имахме списък за сортиране като този, той ще позволи сортирането на адреси в тази мрежа.

Може да има до десет двойки мрежа и подмрежа.

Последната ключова дума е опции, тя ни позволява да зададем опции като:

отправя запитване към имената на неговия кръг, за да разпредели товара,

изчакване или брой опити, които клиентът ще направи запитване към DNS сървъра преди да се откаже. Имайте предвид, че ако иска да зададете DNS сървъри за имена на мрежов интерфейс, можете, като ги добавите в специфичните за интерфейса конфигурационни файлове. В настройката също трябва да сложите **peer DNS** за да се прехвърлят конфигурациите в **resolv.conf**.

/etc/sysconfig/network-scripts/ifcfg-eth0

```
BOOTPROTO=dhcp
DEFROUTE=yes
NAME=enp0s3
DEVICE=enp0s3
ONBOOT=yes
DNS1=4.2.2.2
DNS2=8.8.8.8
PEERDNS=yes
```

Файловете за превключване на имена или файловете nsswitch се използват от приложенията за получаване на информация за услуги за имена.

Това не е само за поръчка за разрешаване на имена на домейни, а по-скоро за множество категории.

Всяка категория информация се идентифицира с име на база данни.

Първата колона на файла е името на базата данни. Останалите колони посочват реда, в който да се търсят данните.

Разбрани са следните бази данни.

/etc/nsswitch.conf

```
# Generated by authselect on Sat Oct 3 03:58:29 2020
# Do not modify this file manually.

# If you want to make changes to nsswitch.conf please modify
# /etc/authselect/user-nsswitch.conf and run 'authselect apply-changes'.

# Note that your changes may not be applied as they may be
# overwritten by selected profile. Maps set in the authselect
# profile will always take precedence over the same maps
# set in the user file. Only maps that are not set by the profile
# are applied from the user file.

# For example, if the profile sets:
#   passwd: sss files
# and /etc/authselect/user-nsswitch.conf contains:
#   passwd: files
# hosts: files dns
# the resulting generated nsswitch.conf will be:
#   passwd: files # from profile
#   hosts: files dns # from user file

passwd: sss files system
group: sss files system
netgroup: sss files
automount: sss files
services: sss files

# Include from /etc/authselect/user-nsswitch.conf

# /etc/nsswitch.conf

# Name Service Switch config file. This file should be
# sorted with the most-used services at the beginning.

# Valid databases are: aliases, ethers, group, gshadow, hosts,
# initgroups, netgroup, networks, passwd, protocols, publickey,
# rpc, services, and shadow.

# Valid service provider entries include (in alphabetical order):

compat          Use /etc files plus +_compat pseudo-db
db              Use the pre-processed /var/db files
dns              Use DNS (Domain Name Service)
files            Use the local files in /etc
hesiod           Use Hesiod (DNS) for user lookups
nis              Use NIS (NIS version 2), also called YP
nisplus          Use NIS+ (NIS version 3)

See 'info libc "NSS Basics"' for more information.

# Commonly used alternative service providers (may need installation):

ldap             Use LDAP directory server
myhostname        Use system host names
machines          Use systemd machine names
mdns*, mdns*_minimal Use Avahi mDNS/DNS-SD
resolve           Use system-resolved resolver
sss              Use System Security Services Daemon (sssd)
systemd           Use systemd for dynamic user option
winbind           Use Samba winbind support
wsl              Use WSL host support
wrapper          Use wrapper module for testing

# Notes:
```

```
# Valid service provider entries include (in alphabetical order):

compat          Use /etc files plus +_compat pseudo-db
db              Use the pre-processed /var/db files
dns              Use DNS (Domain Name Service)
files            Use the local files in /etc
hesiod           Use Hesiod (DNS) for user lookups
nis              Use NIS (NIS version 2), also called YP
nisplus          Use NIS+ (NIS version 3)

See 'info libc "NSS Basics"' for more information.

# Commonly used alternative service providers (may need installation):

ldap             Use LDAP directory server
myhostname        Use system host names
machines          Use systemd machine names
mdns*, mdns*_minimal Use Avahi mDNS/DNS-SD
resolve           Use system-resolved resolver
sss              Use System Security Services Daemon (sssd)
systemd           Use systemd for dynamic user option
winbind           Use Samba winbind support
wsl              Use WSL host support
wrapper          Use wrapper module for testing

# Notes:

# 'sssd' performs its own 'files'-based caching, so it should generally
# come before 'files'.

# WARNING: Running nscd with a secondary caching service like sssd may
# lead to unexpected behaviour, especially with how long
# entries are cached.

# Installation instructions:

# To use 'db', install the appropriate package(s) (provide 'makedb' and
# 'libnss_db.so.'), and place the 'db' in front of 'files' for entries
# you want to be looked up first in the databases, like this:

passwd:  db files
shadow:  db files
group:   db files

# In order of likelihood of use to accelerate lookup.
shadow:  files sss
hosts:   files dns myhostname

aliases:   files
ethers:    files
gshadow:   files
# Allow initgroups to default to the setting for group.
# initgroups: files
networks:  files dns
protocols: files
publickey: files
rpc:       files

# Notes:
```

Псевдоними за пощенски псевдоними, етери за Ethernet номера

група за групи потребители, публикации за имена и номера на хостове,

initgroups, за допълнителни списъци с групи,

netgroup за мрежови групи, мрежки за мрежови имена и номера, парола за пароли, протоколи за мрежови протоколи, publickey, за секретни клавиши, използвани от NFS и NIS plus,

rpc за номера за отдалечени процедури,

услуги за мрежови услуги и накрая

shadow, за пароли в сянка.

Ние се интересуваме основно от ключовата дума hosts,

възможни опции биха били

- файлове,
- dns,
- nis,
- nisplus.

Последните две биха използвали

- sons,
- nis
- nisplus,

за резолюция на хостинг.

Първите две биха били за по-чести ситуации.

Опцията за файлове посочва разрешаването на hostnames, използва се в локалния файл за разрешаване на имена **/etc/hosts**.

DNS ще инструктира клиента да реши чрез DNS.

Можем да използваме и двете и след това ще ги изprobваме по ред.

Отляво надясно, за случаите е обичайно да се използват хостове

двоеточие пространство файл, пространство DNS,

```
hosts:    files dns
```

по този начин той ще се консултира

първо файлът **/etc/hosts** и след това се опитайте да разрешите DNS втори.

Това ни позволява да заменим DNS с локален запис на **/etc/hosts**.

След всяка опция можем да имаме незадължителен статус и действие в квадратни скоби.

```
hosts:      dns [ !UNAVAIL=return ] files
```

Състоянието може да бъде успешно, когато не е възникнала грешка, не е намерена там, където търсенето е успешно, но записът не е намерен, недостъпен, когато услугите са постоянно недостъпни, или опитайте отново, когато услугата временно не е налична.

Това може да означава, че файлът е заключен към сървъра с този процес на тази мини заявка - временно.

Действието може да бъде едно от връщането, което се връща
результатът веднага, продължете
който преминава към следващия метод в списъка,
или обединяване, което позволява на системата да премине към следващата
метод, след резултат и след това обединява резултатите.

- return

- continue

- merge

Можем да използваме удивителен знак, за да отречем състоянието
да съответства на всичко, освен на това, което е в квадратните скоби.

В нашия пример по-рано щеше да се върне ако DNS не е бил наличен.

Което е същото като да бъдеш на разположение и след това да не преминеш към разрешаване чрез файла /etc/hosts.

```
hosts:      dns [ !UNAVAIL=return ] files
```

Systemd-networkd is manage е новият стек за управление на мрежовите конфигурационни файлове на linux.

Заедно с това идва systemd, който е решен.

Разрешено осигурява разделителна способност за конкретно име.

В момента управляваме глобалната конфигурация за разрешаване на имена с:

/etc/resolv.conf и е спрямо интерфейс устройство в:

/etc/sysconfig/network-scripts/ifcfg_*

Настоящата система е доста твърда и не е много динамична.

Не е необично в дневно време да бъдете свързани към повече от една динамична мрежа наведнъж и всеки да има различни сървъри за разрешаване на имена.

Например може да имаме VPN връзка преминаваща през wifi обществена мрежа.

DNS сървърите, свързани с тези връзки, ще идват и си отиват, както връзките.

Ако DNS не се обработва динамично за екземпляр, тогава може да има стари DNS записи, които са без значение към сегашната ни среда.

Например, ако прекъснем връзката с VPN, но DNS записите отразяват устройства в тези мрежи.

Ние избираме кои форми за разрешаване на имена да използваме и в кой ред във файла /etc/nsswitch.conf

cat /etc/nsswitch.conf ако се налага скролнете до hosts реда това е редът, в който системата ви разрешава имената на хостове.

hosts: files dns myhostname

Моят казва файлове и след това името на хоста ми.

Това означава, че първо се консулира с локалния /etc/host файл, и след това иска от DNS да разреши името.

И накрая NSS - модулът за плъгин за името ми на хост за локално конфигурираното име на хост на системата.

Като се връща от системното повикване get host name.

Ако искате да включите разрешената от системата услуга (systemd – resolved) в действието, ще добавим думата resolv към този ред трябва да решим къде искаме да го включим, за да определим реда.

Можем също да преместим файла /etc/resolv.conf и след това да свържем /run/systemd/resolve/resolv.conf.

```
sudo ln -s /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

```
sudo systemctl start systemd-resolved
```

```
sudo systemctl enable systemd-resolved
```