нетология

Сеть и сетевые протоколы: **L3-сеть**

Андрей
Вахутинский

Андрей Вахутинский

Заместитель начальника IT-отдела

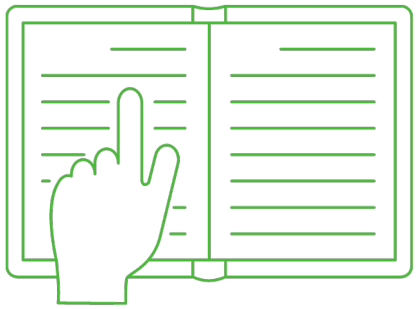
АО “ИНТЕКО”

Андрей Вахутинский

Предисловие

Эта лекция содержит основные понятия, связанные с 3-м уровнем модели OSI (Network / Сетевой уровень).

Также вы познакомитесь с основными командами, которые позволяют получать информацию / вносить изменения в настройки ОС на 3-м уровне модели OSI.



План занятия

1. Предисловие
2. IPv4
3. IPv6
4. Работа маршрутизатора
5. Маршрутизация
6. Популярные сетевые утилиты
7. Итоги
8. Домашнее задание

IPv4

Проблема разграничения сетей

В предыдущей лекции мы говорили о необходимости наличия двух типов адресов:

- **L2 MAC** для адресации внутри локальной сети (шестнадцатеричная запись);
- **L3 IP** для адресации между сетями (десятичная запись).

Мы увидели, что хосты в сети имеют оба типа этих адресов одновременно.

```
vagrant@netology1:~$ ip addr show eth0 | egrep '(ether|inet )'
    link/ether 08:00:27:d2:23:ca brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
```

Проблема разграничения сетей

На уровне L2 все несложно – формат Ethernet фрейма предполагает прямую доступность в локальной сети и адресацию по SRC + DST MAC.

При общении приложений по спецификациям TCP/IP, ОС не оперирует адресами *канального* уровня: TCP соединения устанавливаются между *сетевыми* адресами.

Каким же образом хост «понимает», что адрес сетевого уровня относится или не относится к его собственной сети? И если адрес «чужой», то по какому пути до него можно добраться?

Подобное разграничение L3 сетей достигается дополнительной битовой маской, назначаемой хосту вместе с IP адресом, – маской подсети.

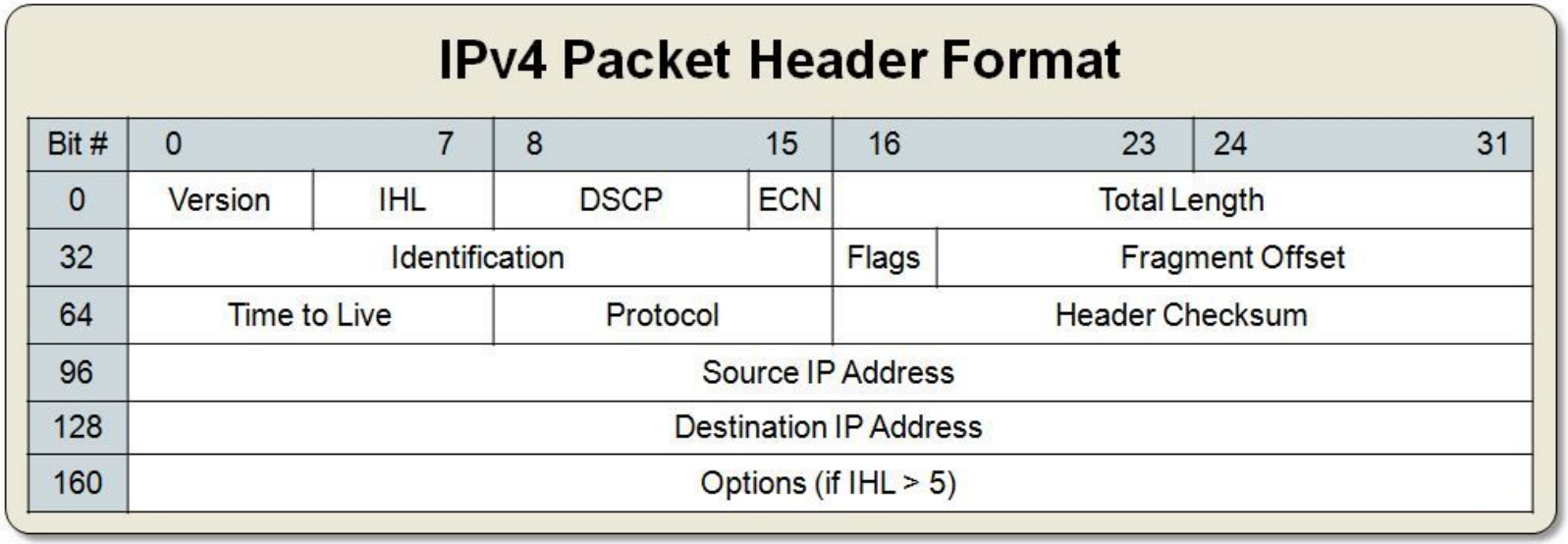
IPv4

Протокол IP (Internet Protocol) создан для использования во взаимосвязанных системах компьютерных сетей с коммутацией пакетов.

Относится к **межсетевому уровню модели TCP/IP**.

Стандарт протокола описан в RFC 791

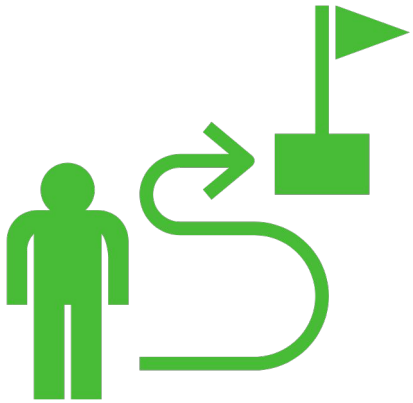
Формат заголовков IPv4



Источник изображения: microchipdeveloper.com

Назначение IPv4

- логическая адресацию хостов на основе IP-адреса;
- инкапсулирует данные вышестоящих протоколов;
- осуществляет маршрутизацию данных между хостами;
- осуществляет фрагментацию IP-пакетов.



Ограничения IPv4

- не устанавливает соединения;
- не гарантирует доставку;
- не обеспечивает обеспечения сохранение последовательности данных при передаче;
- не устраняет возможное дублирования пакетов.

➡️ Решение перечисленных задач возложено на протоколы транспортного уровня. Например, на протокол управления передачей (TCP).

Ограничения адресации

IPv4 использует адреса размером четыре байта (32 бита), которые ограничивают адресное пространство до 4 294 967 296 адресов.

Последний блок IP-адресов был выделен 3 февраля 2011 года.

Специальные адреса подсети

Широковещательный адрес сети – пакеты с широковещательным адресом получат все хосты этой сети.

Вычисляется:

ADDRESS **OR NOT**(MASK)

192.168.0.255

Адрес сети – позволяет определить, что хосты находятся в одной сети.

Вычисляется:

ADDRESS **AND** MASK

192.168.0.0

IP-адрес

IP-адрес представляет из себя 32-битное число.

На практике для более удобного представления адреса записываются в точечно-десятичной нотации, например: 192.168.0.1

IP-адрес состоит из двух частей:

- адрес хоста (его сетевого интерфейса);
- адрес сети («путь» к этому хосту в сети).

Подсети

Для точного указания какая часть IP-адреса является адресом хоста, а какая – адресом сети, существует понятие «маска».

Маска сети – количество бит в адресе, которое отведено под адрес сети.

Маска может указываться двумя эквивалентными способами:

192.168.0.1/24 либо 255.255.255.0

Адресация

Изначально под адрес сети отводился только старший октет и оставшаяся часть представляла из себя адрес хоста:

192.0.0.1

Таким образом, IP-протокол мог адресовать максимум 256 сетей с 16 млн. хостов в каждой.

Адресация

Для более удобного использования адресного пространства была введена **классовая адресация**, при которой старший октет определял тип адреса сети и количество хостов.

Классовая адресация использовалась с 1981 по 1993 годы.

Классовая адресация

класс	первые октет	распределение байт (сеть, хост)	кол-во сетей	кол-во хостов	маска	начальный адрес	конечный адрес
A	0	C.X.X.X	126	16777214	255.0.0.0	1.0.0.0	126.255.255.255
B	10	C.C.X.X	16384	65534	255.255.0.0	128.0.0.0	191.255.255.255
C	110	C.C.C.X	2097152	254	255.255.255.0	192.0.0.0	223.255.255.255
D	1110	multicast (групповой адрес)				224.0.0.0	239.255.255.255
E	11110	резерв				240.0.0.0	255.255.255.255

Бесклассовая адресация

Ввиду быстрого развития сети Интернет, классовая адресация также перестала удовлетворять нуждам пользователей.

В 1993 году появилась бесклассовая адресация или **Classless Inter-Domain Routing** (CIDR).

При этом типе адресации стало возможным применение любых масок подсети к пространству IP-адресов.

IPv4 маска, форматы записи

По формату адреса IP видно, что в нем нет признаков, которые помогли бы с определением границ сети. Для этого существует **маска**.

В случае с IPv4 это такое же 32-битное число, как и сам адрес:

```
vagrant@vagrant:~$ ip -4 addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic eth0
       valid_lft 49201sec preferred_lft 49201sec
```

IPv4 маска, форматы записи

/24 – говорит о том, что из 32 бит адреса 24 бита являются адресом сети, остальные
8 бит – хостовой частью.

Маска /24 CIDR формата в ином варианте записи:
1111 1111.1111 1111.1111 1111.0000 0000 (24 + 8) или 255.255.255.0.

Число адресов в подсети легко прикинуть по CIDR:

$2^8 = 256$.

```
vagrant@vagrant:~$ netmask -s 10.0.2.15/24
       10.0.2.0/255.255.255.0
```

Логическое И

Мы узнали о характеристиках IPv4 адреса, о маске подсети, десятичном и двоичном представлении этих данных.

Далее – нужно собрать всё воедино битовым логическим оператором AND (И), который возвращает:

- **0** – если хотя бы один из битов равен 0;
- **1** – если оба бита равны 1.

Пример:

```
      00110 &
      01011
=     00010
>>> res = 0b00110 & 0b01011;
>>> format(res, '05b')
'00010'
```

Результат применения логического И – адрес сети.

Логическое ИЛИ

Битовый логический оператором OR (ИЛИ) возвращает:

- **1** – если хотя бы один из битов равен 1;
- **0** – если оба бита равны 0.

Пример:

```
      00110 &
      11111
=     11111
>>> res = 0b00110 | 0b11111;
>>> format(res, '05b')
'11111'
```

Результат применения логического И между инвертированной маской сети и адресом хоста – бродкаст адрес.

Простой пример применения маски IPv4

Сетевая часть (n) / хостовая (h)	nnnnnnnn	nnnnnnnn	nnnnnnnn	hhhhhhhh
Адрес десятичный	192	168	0	10
Адрес двоичный	11000000	10101000	00000000	00001010
Маска десятичная	255	255	255	0
Маска двоичная	11111111	11111111	11111111	00000000
Инвертированная маска	00000000	00000000	00000000	11111111
Адрес сети двоичный	11000000	10101000	00000000	00000000
Адрес сети десятичный	192	168	0	0
Бродкаст адрес двоичный	11000000	10101000	00000000	11111111
Бродкаст адрес десятичный	192	168	0	255

Простой пример применения маски IPv4

192.168.0.10/24 – это хост с адресом 192.168.0.10.

Этот хост:

- имеет маску подсети /24 или 255.255.255.0;
- находится в подсети 192.168.0.0.

По стандарту маски должны обеспечивать смежность подсетей, поэтому они всегда будут выглядеть как последовательность из 1 в начале и 0 в конце. Адрес всегда принадлежит какой-то подсети.

Сложный пример применения маски IPv4

Маска десятичная	255	255	224	0
Адрес десятичный	64	233	161	138
Адрес двоичный	01000000	11101001	10100001	10001010
Маска двоичная	11111111	11111111	11100000	00000000
Сетевая часть	nnnnnnnn	nnnnnnnn	nnn	
Хостовая часть			hhhhh	hhhhhhhh
Адрес сети двоичный	01000000	11101001	10100000	00000000
Адрес сети десятичный	64	233	160	0

Сложный пример применения маски IPv4

Еще один пример на базе публичного адреса Google:

```
vagrant@vagrant:~$ google_v4_addr=$(dig +short A google.com | head -n1)
vagrant@vagrant:~$ echo $google_v4_addr; whois $google_v4_addr | grep CIDR
64.233.161.138
CIDR:           64.233.160.0/19
```

Под хостовую часть отделен не просто целый последний октет (/24), а /19, таким образом в подсеть попадают адреса от 64.233.160.0 до 64.233.191.255.

Особенности нумерации IPv4

В первом примере 192.168.0.0 – *специальный зарезервированный адрес*, он называется **адресом подсети**.

Последний адрес подсети, 192.168.0.255, также является зарезервированным, – он предназначен для **широковещательных** сообщений на уровне L3 (по аналогии с адресом ff:ff:ff:ff:ff:ff L2 MAC).

Часто адресом шлюза, через который можно попасть в другие сети, выступает первый доступный адрес сети, в данном случае - 192.168.0.1.

Нетрудно подсчитать, что при маске 255.255.255.0 есть всего 256 адресов в последнем октете (хостовой части), из которых 2 недоступно (адрес подсети и широковещательный адрес в резерве), 1 уже занят нашим хостом 192.168.0.10, а, значит, в данной подсети может находиться еще 253 соседних адреса (или

Особенности нумерации IPv4

Нетрудно подсчитать, что при маске 255.255.255.0 есть всего 256 адресов в последнем октете (хостовой части), из которых 2 недоступно (адрес подсети и широковещательный адрес в резерве), 1 уже занят нашим хостом 192.168.0.10, а, значит, в данной подсети может находиться еще 253 соседних адреса (или даже 252 если считать шлюз).

Автоматизированный калькулятор **ipcalc**:

```
vagrant@vagrant:~$ ipcalc 192.168.0.10/24
Address:   192.168.0.10          11000000.10101000.00000000. 00001010
Netmask:   255.255.255.0 = 24    11111111.11111111.11111111. 00000000
Network:   192.168.0.0/24        11000000.10101000.00000000. 00000000
HostMin:   192.168.0.1           11000000.10101000.00000000. 00000001
HostMax:   192.168.0.254         11000000.10101000.00000000. 11111110
Broadcast: 192.168.0.255         11000000.10101000.00000000. 11111111
```

Пример

Рассмотрим IP-адрес 192.168.0.1/24

Число «24» после косой черты означает количество разрядов в маске подсети, т.е. в нашем случае маска подсети в двоичном представлении будет иметь вид:
11111111 11111111 11111111 00000000

или в десятичном представлении:
255.255.255.0

Важно:

в маске могут подряд быть 1, т.е. не может быть 101010 и т.д.

Пример

Это означает, что первые 24 разряда IP-адреса отводятся на номер сети, а оставшиеся восемь – на адреса хостов этой сети.

Получаем, что записи 192.168.0.0/24 означает диапазон возможных адресов хостов от 192.168.0.0 до 192.168.0.255 (2^8 = 256 от 0 до 255).

Кроме этого, в каждой сети используются два специальных адреса: адрес сети и широковещательный адрес.

Поэтому, диапазон адресов хостов будет от 192.168.0.1 до 192.168.0.254

Особенные диапазоны

В диапазоне IPv4 есть множество адресов, имеющих специальное назначение:

- всем известный 127.0.0.1, который на самом деле лишь 1 стандартный адрес loopback интерфейса из огромной (16+ млн.) подсети 127.0.0.0/8;
- немаршрутизируемые частные сети 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16;
- немаршрутизируемые link-local адреса 169.254.0.0/16, которые могут быть, например, назначены при неуспешной работе DHCP клиента;
- диапазон 224.0.0.0/4 для multicast (широковещательная групповая передача);
- 100.64.0.0/10 для carrier-grade NAT;
- и многие другие, ознакомьтесь с полным списком на Wiki.

IPv6

IPv6 уже здесь

Для системного администратора/DevOps внедрение IPv6 с современными ядрами не является проблемой.

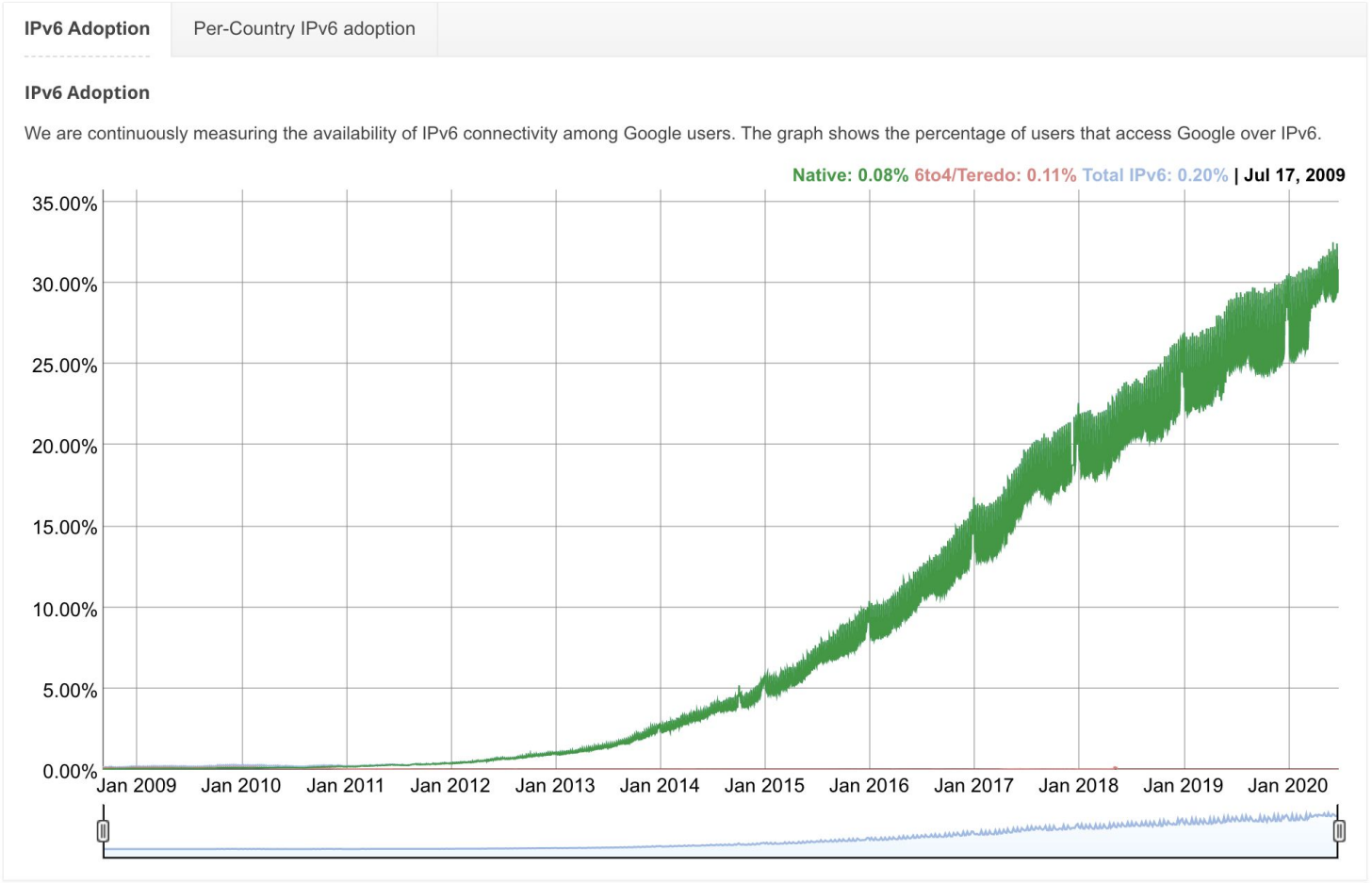
Большинство приложений нормально работают с IPv6, решены проблемы dual-stack систем (с IPv4 и IPv6 одновременно).

Утилиты:

- ping6;
- traceroute6;
- ip -6;
- ...
- /etc/gai.conf для настройки;
- dual-stack.

IPv6 уже здесь

Статистика Google:



IPv6

IPv6 – новый стандарт протокола IP, призванный исправить недостатки IPv4.

Некоторые преимущества IPv6:

- увеличенное адресное пространство (128 бит);
- автоконфигурация (не нужно настраивать адреса вручную);
- Jumbogram (передача до 4 Гб данных в одном пакете).

Адреса IPv6

IPv6-адрес имеет вид:

2001:0:0:0:DB8:800:200C:417A

2001::DB8:800:200C:417A (сокращенный формат)

При использовании в URL, необходимо *заключать в скобки* «[]».

https://[2001:0:0:0:DB8:800:200C:417A]:8080/

Адреса IPv6

IPv6-адрес может быть:

- **Unicast** – обычный адрес хоста;
- **Anycast** – групповой адрес, пакет будет доставлен наиболее близкому хосту с точки зрения протокола маршрутизации;
- **Multicast** – групповой адрес, пакеты будут доставлены каждому хосту в группе;
- **Broadcast** – не реализованы.

Работа маршрутизатора

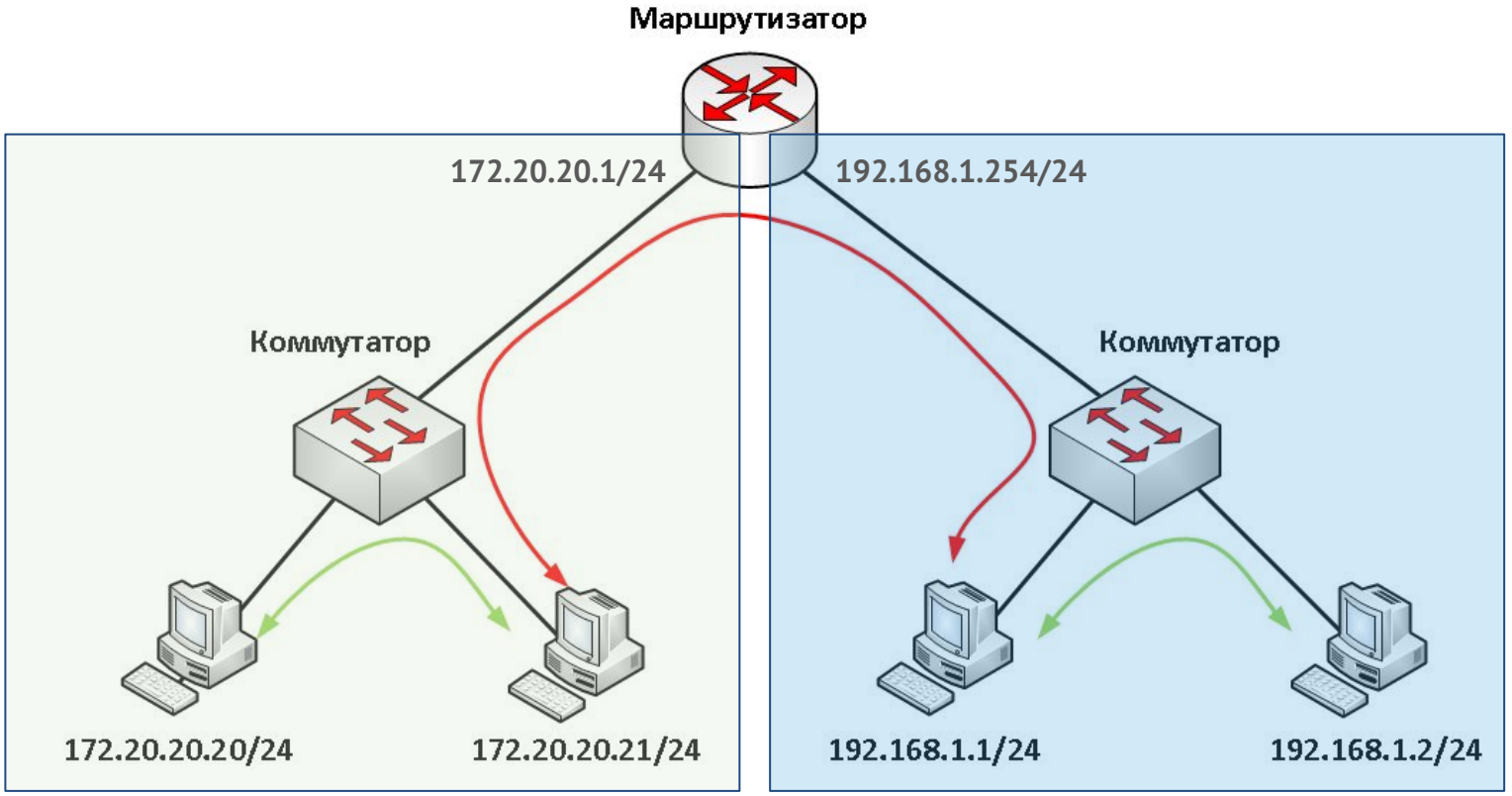
Маршрутизатор

На прошлой лекции мы посмотрели, как происходит взаимодействие внутри сети.

А как оно происходит при коммуникации устройств, находящихся
в разных сетях?

Маршрутизатор

Основная задача маршрутизатора – объединение сетей на сетевом уровне.



Маршрутизатор

Маршрутизатор стоит на границе сети (сегмента сети).

Внутри каждого сегмента сети адресация происходит по аппаратным MAC-адресам.

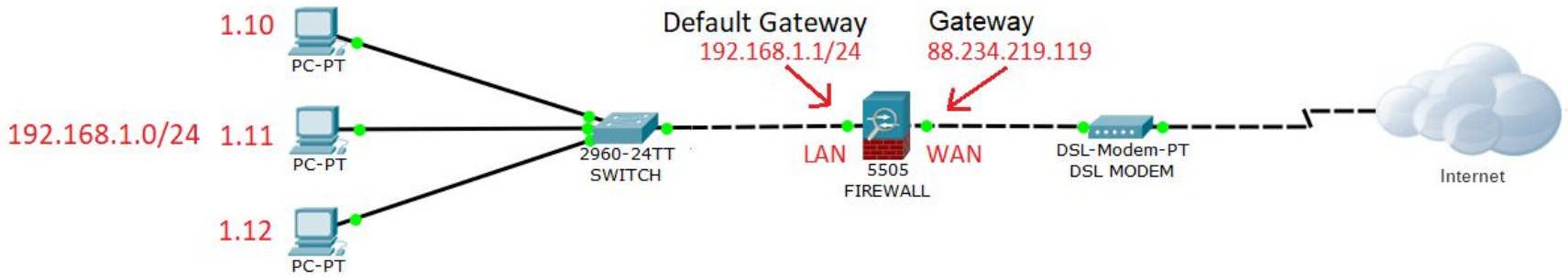
Вне сегмента сети такие адреса недоступны.

Из этого следует, что при своей работе маршрутизатор вынужден подменять MAC-адрес запроса на свой собственный MAC-адрес в другом сегменте (он объединяет несколько сегментов – это его задача).

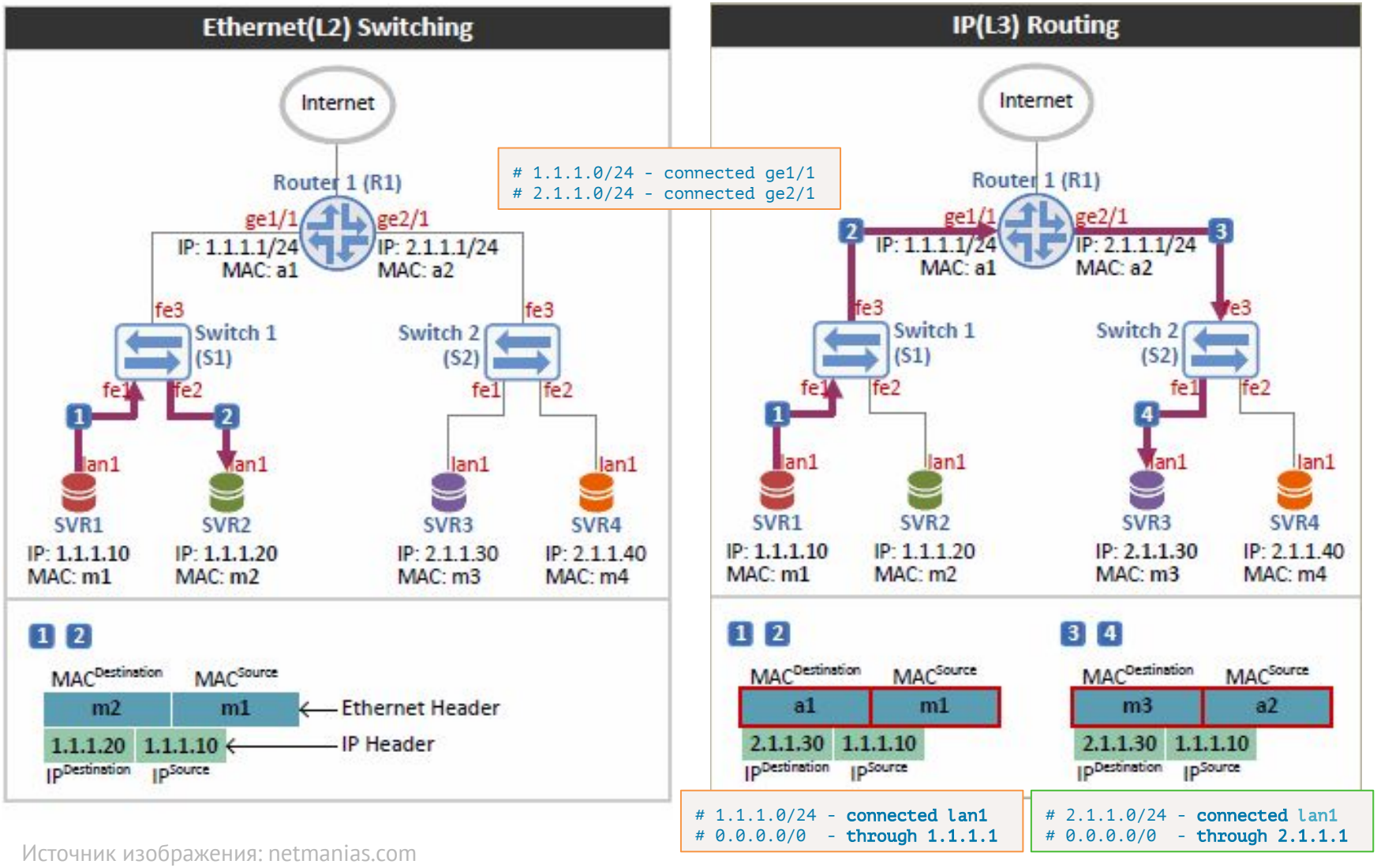
Маршрут по умолчанию

У каждого устройства, которое выходит в интернет, необходимо прописать маршрут по-умолчанию (default gateway).

Без него устройство не будет знать, какому устройству внутри его сегмента сети, перенаправлять пакеты для дальнейшей отправки получателю.



Двухуровневая адресация



Источник изображения: netmanias.com

Маршрутизация

Маршрутизация

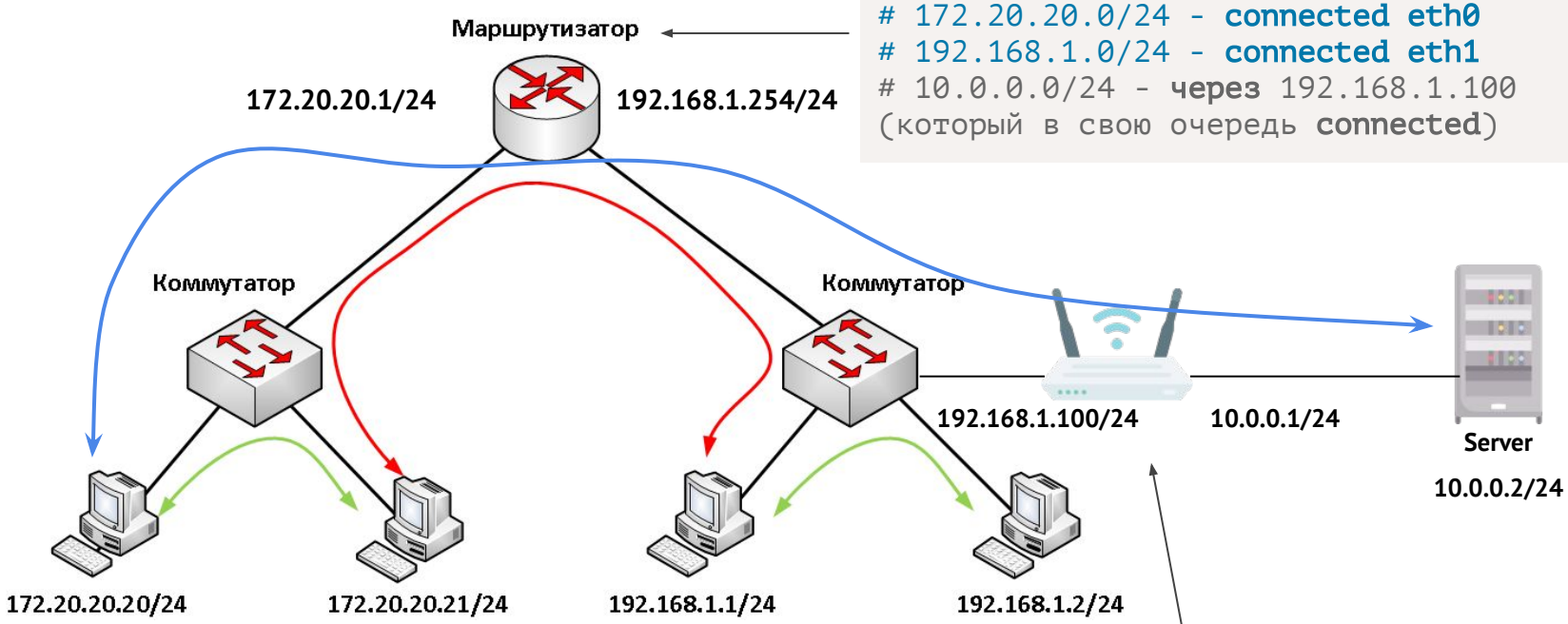
Статическая маршрутизация — вид маршрутизации, при котором маршруты указываются в явном виде при конфигурации маршрутизатора.

Вся маршрутизация при этом происходит без участия каких-либо протоколов маршрутизации.

Динамическая маршрутизация — вид маршрутизации, при котором таблица маршрутизации редактируется программно.

Вы настраиваете протокол маршрутизации, а он вносит изменения в таблицу маршрутизации.

Статическая маршрутизация



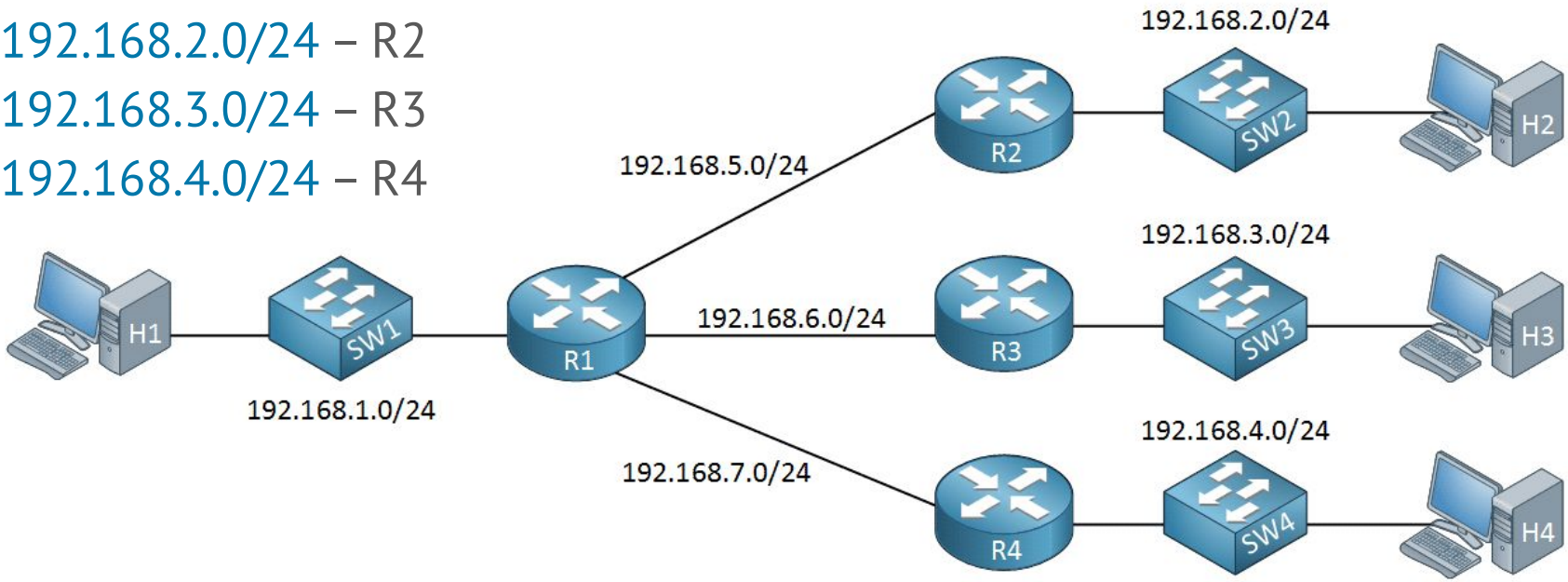

```
# 172.20.20.0/24 - connected eth0
# 192.168.1.0/24 - connected eth1
# 10.0.0.0/24 - через 192.168.1.100
(который в свою очередь connected)
```

```
# 10.0.0.0/24 - connected
# 192.168.1.0/24 - connected
# 172.20.20.0/24 - через 192.168.1.254 (который в свою очередь connected)
# 0.0.0.0/0 (default) - через 192.168.1.254
```

Динамическая маршрутизация

На схеме 4 филиала, 4 основных сети в таблице маршрутизации.
А если филиалов 100? И надо добавить ещё 1 – сколько вносить изменений?

192.168.1.0/24 – R1
192.168.2.0/24 – R2
192.168.3.0/24 – R3
192.168.4.0/24 – R4



Протоколы маршрутизации

По алгоритмам:

- Дистанционно-векторные (Distance Vector) – RIP (зависит от количества хопов);
- Протоколы состояния каналов (Link-state) – OSPF (зависит от метрик линков);
- Усовершенствованные дистанционно-векторные – EIGRP (смесь).

По области применения:

- Междоменная маршрутизация – BGP;
- Внутридоменная маршрутизация – OSPF, RIP, EIGRP.

Популярные сетевые утилиты

TTL и traceroute

Одно из важных полей в заголовках IP – TTL или time to live (время жизни).

При прохождении IP пакета через роутер, счетчик в этом поле уменьшается на единицу.

Пакеты, в которых TTL достиг нуля, уничтожаются.

Когда это происходит, роутер формирует сообщение ICMP Time Exceeded.

TTL и traceroute

Данное свойство используется в traceroute:

- формируется пакет с TTL = 1, первый на пути следования роутер уменьшает TTL до 0 и отвечает Time Exceeded (первый хоп);
- traceroute формирует пакет с TTL = 2, он успешно преодолевает первый роутер, и уже второй отвечает Time Exceeded;
- когда traceroute вместе Time Exceeded получает Connection Refused или случайно выбранный порт совпадает с каким-то слушающим сервисом, искомый хост считается достигнутым.

TTL и traceroute

➡️ Таким образом, используя src IP и метки времени, traceroute узнает и трассу прохождения пакета, и время его прохождения.

По умолчанию traceroute в Linux использует UDP пакеты со случайным портом, однако ключами можно выбрать TCP (-T) или ICMP режим (-I). mtr вместо разового прохождения трассы собирает статистику.

mac, ip

Посмотреть интерфейсы:

```
osboxes@osboxes:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
    link/ether 08:00:27:f4:39:50 brd ff:ff:ff:ff:ff:ff
```

Посмотреть адреса (для IPv4):

```
osboxes@osboxes:~$ ip -4 address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 86328sec preferred_lft 86328sec
```

ping

ping (ICMP echo request + ICMP echo reply) в представлении не нуждается:

```
osboxes@osboxes:~$ ping -c 1 netology.ru
PING netology.ru (104.26.8.143) 56(84) bytes of data.
64 bytes from 104.26.8.143 (104.26.8.143): icmp_seq=1 ttl=63 time=3.88 ms
--- netology.ru ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.877/3.877/3.877/0.000 ms
```

Поэтому попробуем что-то поинтереснее:

```
osboxes@osboxes:~$ ping -M do -s $((2000-28)) -c1 netology.ru
PING netology.ru (172.67.75.22) 1972(2000) bytes of data.
ping: local error: message too long, mtu=1500
--- netology.ru ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

osboxes@osboxes:~$ ping -M do -s $((1500-28)) -c1 netology.ru
PING netology.ru (172.67.75.22) 1472(1500) bytes of data.
1480 bytes from 172.67.75.22 (172.67.75.22): icmp_seq=1 ttl=63 time=10.1 ms
--- netology.ru ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 10.135/10.135/10.135/0.000 ms
```

Где 28 – длина заголовков IP (20) + ICMP (8).

Как tcpdump поможет при проблемах с маской?

```
root@netology1:~# ip -4 a s eth1 | grep inet
    inet 172.28.128.10/24 scope global eth1
root@netology2:~# ip -4 a s eth1 | grep inet
    inet 172.28.128.60/24 scope global eth1
root@netology1:~# ip r get 172.28.128.60
172.28.128.60 dev eth1 src 172.28.128.10 uid 0
```

curl 172.28.128.60 работает.

Ломаем на хосте netology2 сеть, устанавливая заведомо некорректную маску:

```
root@netology2:~# ip addr del 172.28.128.60/24 dev eth1
root@netology2:~# ip addr add 172.28.128.60/30 dev eth1 # вместо /24
root@netology2:~# ip -4 a s eth1 | grep inet
    inet 172.28.128.60/30 scope global eth1
```

Как tcpdump поможет при проблемах с маской?

Если запустить в этот момент tcpdump на обоих хостах, будет видно, что SYN доходит до сервера, но он не отвечает клиенту SYN+ACK, и клиент перепосылает SYN:

```
root@netology1:~# tcpdump -nn -i eth1
10:34:38.393610 IP 172.28.128.10.50700 > 172.28.128.60.80: Flags [S]...
10:34:39.423744 IP 172.28.128.10.50700 > 172.28.128.60.80: Flags [S]...
root@netology2:~# tcpdump -nn -i eth1
10:34:37.621593 IP 172.28.128.10.50700 > 172.28.128.60.80: Flags [S]...
10:34:38.652240 IP 172.28.128.10.50700 > 172.28.128.60.80: Flags [S]...
```

Это только один из примеров, как tcpdump быстро покажет, на каком хосте случились проблемы с сетью (ведь зная как устанавливается TCP соединение мы сразу видим, на какой стадии происходит проблема и на каком хосте).

Итоги

Итоги

Сегодня мы познакомились с:

- базовыми представлениями про IPv4;
- IPv6;
- познакомились с работой маршрутизаторов.

Домашнее задание

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Андрей Вахутинский