



# Сетевой стек Linux

Эксплуатация высоконагруженных систем



# Андрей Трофимов

Системный администратор, ЕНОТ



# Цель занятия:

Изучить

- ключевые концепции и технологии передачи данных
- их практическое применение и реализацию в Linux
- рассмотреть современные подходы и технологии в сетевой инфраструктуре.

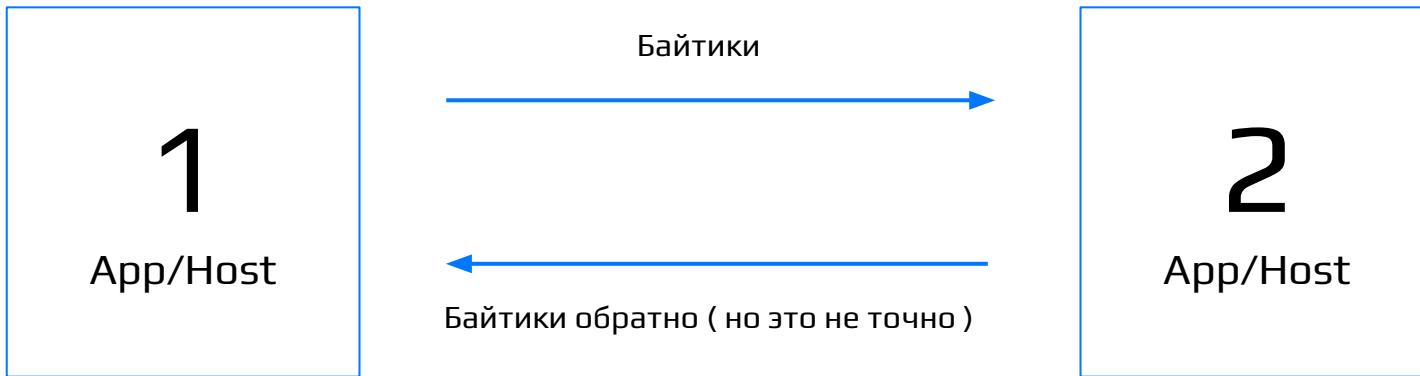
# О чём поговорим?

- Общая теория сетей
  - Как данные передаются внутри сервера
  - Как сервера общаются между собой
- Как это работает на практике
  - Сетевой стек Linux
  - Настройка и диагностика
  - Актуальные сетевые технологии
  - Сеть в контейнерах и оркестраторах

# Что такое передача данных



# Что такое передача данных



# Что такое передача данных



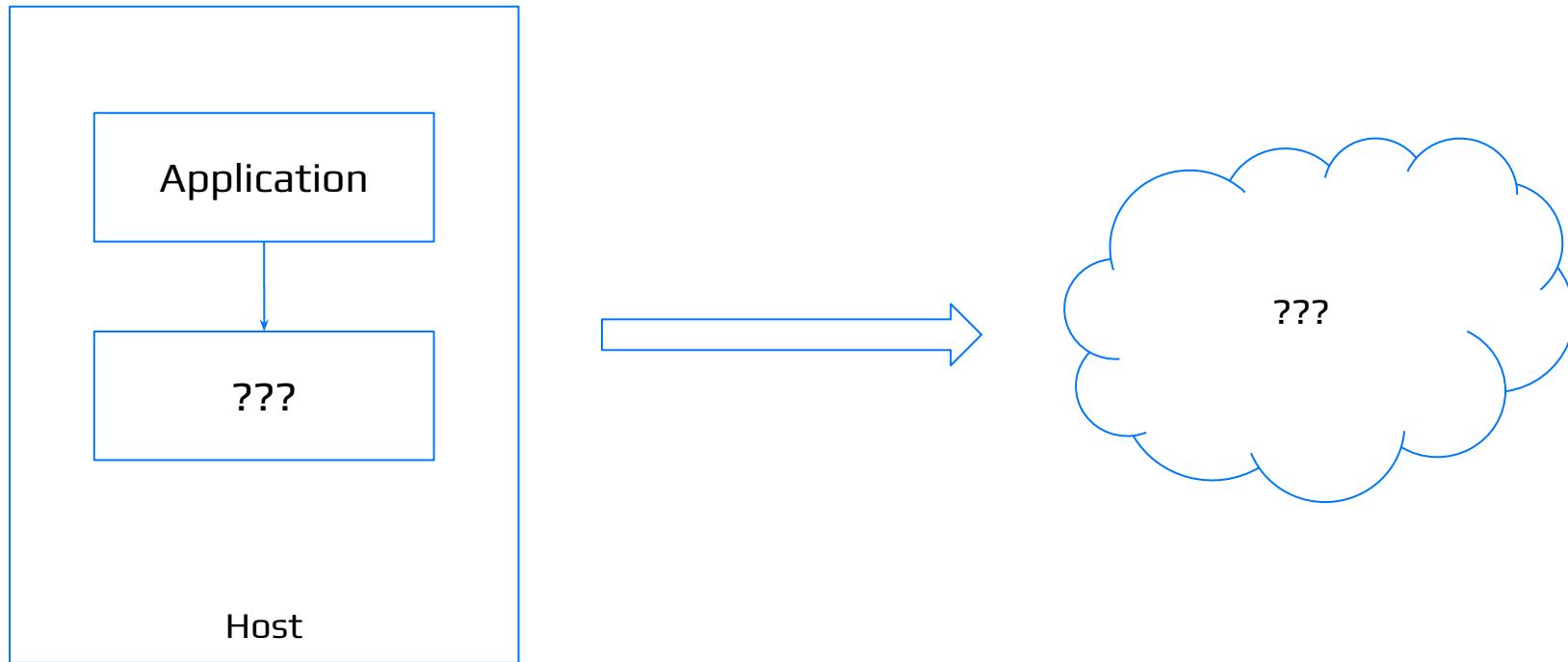
chatgpt советует проверить этот самый сетевой доступ 18:09

ты можешь тут подсказать? 18:09

Если я вообще верно рассуждаю 18:09

А как ты это все смотришь?) 18:24

# Что происходит внутри и снаружи хоста



# Общая теория сетей



# Какой-нибудь реальный запрос



```
→ ~ curl -sv https://corp.vkcdn.ru/media/files/logo_vk.zip -o /tmp/logo_vk.zip
```

*<some kind of magic>*

```
* Connection #0 to host corp.vkcdn.ru left intact
```

```
→ ~ l -h /tmp/logo_vk.zip
```

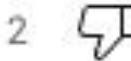
```
-rw-r--r--@ 1 andrey.trofimov wheel 3.0M Sep 16 23:54 /tmp/logo_vk.zip
```

# Здесь должен быть слайд про OSI, но его не будет



@srh\_btk 6 years ago

Модель О Эс Ай правильно говорить. А то ваши Оси ухо режет

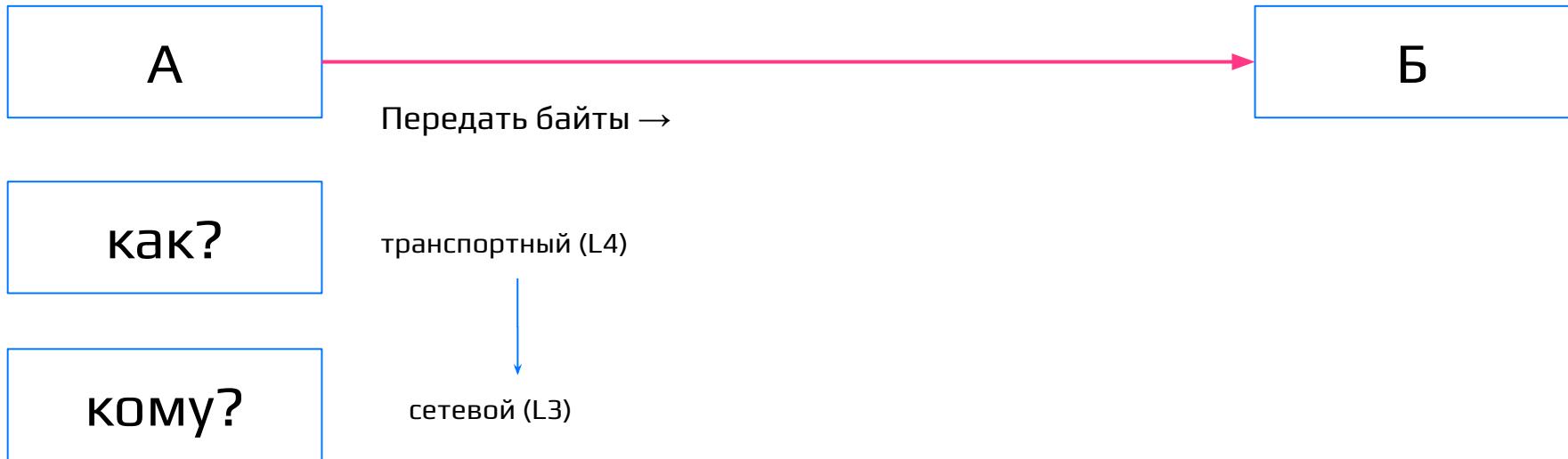


Reply

# Как отправляем?



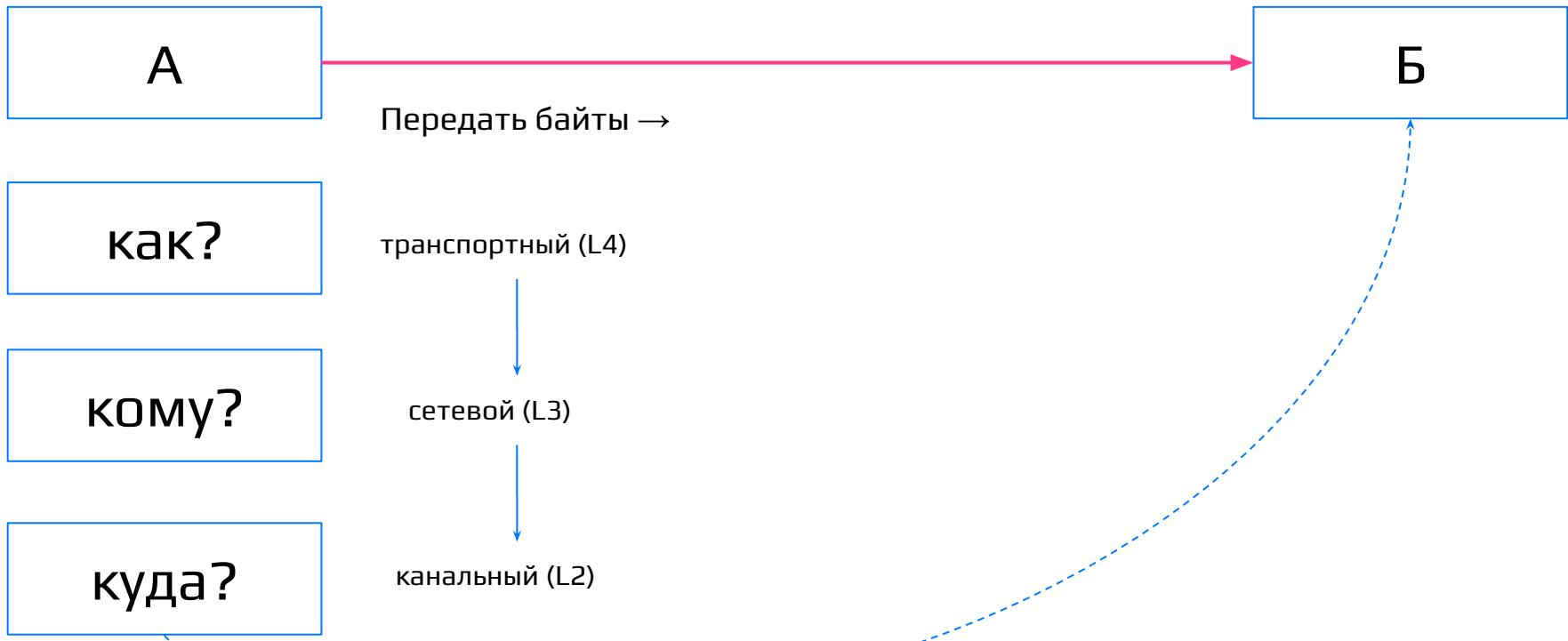
# Кому отправляем?



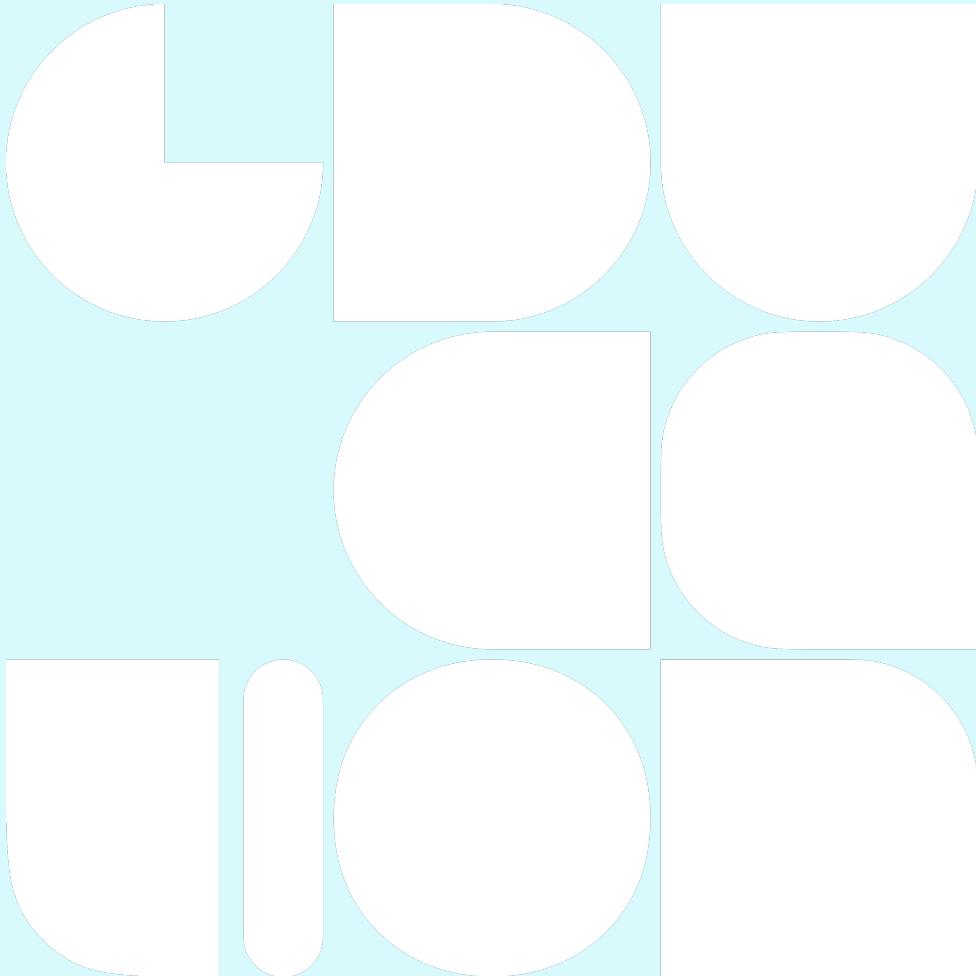
# Куда отправляем?



# Aaaand it's gone



- L4 – транспортный
- L3 – сетевой
- L2 – канальный



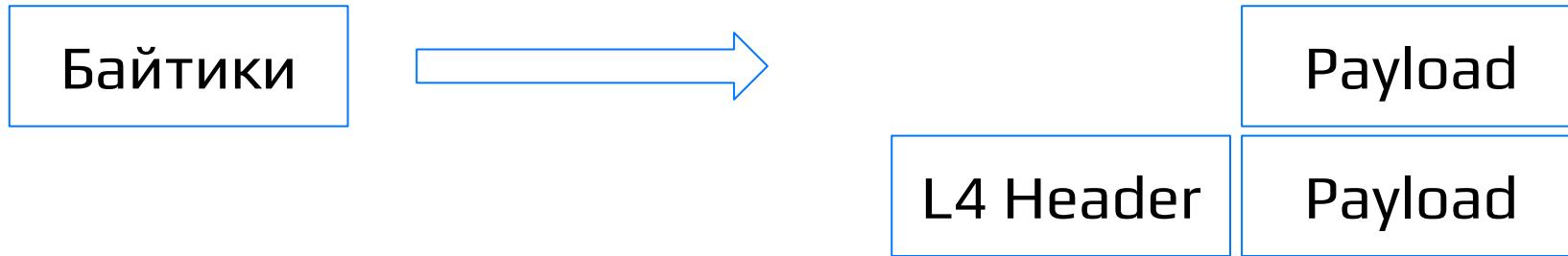
# Как байты становятся пэйлоадом

Байтики



- Нужно добавить некую описательную часть (заголовок)

# Как байты становятся пэйлоадом



- Нужно добавить некую описательную часть (заголовок)
- Просто байты становятся Payload'ом

# UDP

# User Datagram Protocol



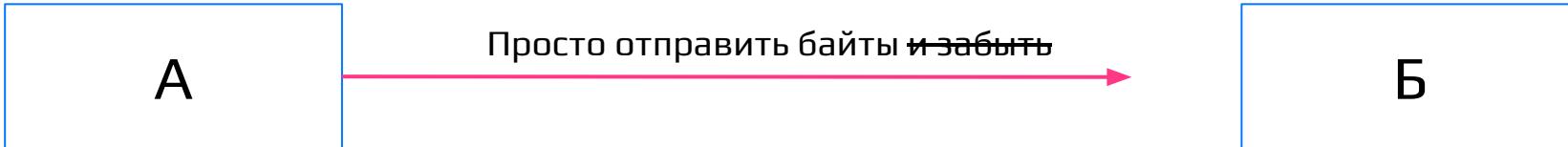
## UDP datagram header

# UDP



Примеры?

# UDP



Примеры?

- всё где важна низкая задержка
- не критичны потери
  - данные неважные
  - данные быстро теряют актуальность

snmp, dns, игры, стриминг, сислог, хартбиты...

# TCP

Transmission Control Protocol



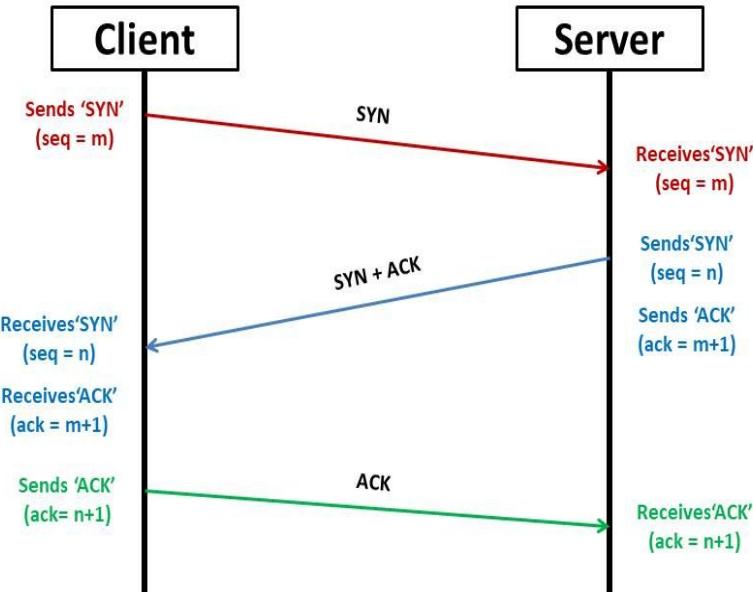
- Создаём соединение (ожидаем ответ) – 3-way handshake
- Управляем потоком – Flow control
- Следим за перегрузками – Congestion control

# TCP

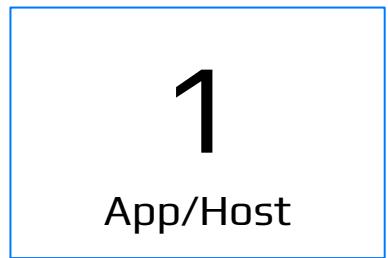
Структура заголовка

Бит	0 – 3	4 – 6	7 – 15	16 – 31
0			Порт источника, <b>Source Port</b>	Порт назначения, <b>Destination Port</b>
32			Порядковый номер, <b>Sequence Number (SN)</b>	
64			Номер подтверждения, <b>Acknowledgment Number (ACK SN)</b>	
96	Длина заголовка, <b>(Data offset)</b>	Зарезервировано	Флаги	Размер Окна, <b>Window size</b>
128		Контрольная сумма, <b>Checksum</b>		Указатель важности, <b>Urgent Pointer</b>
160		Опции (необязательное, но используется практически всегда)		
160/192+		Данные		

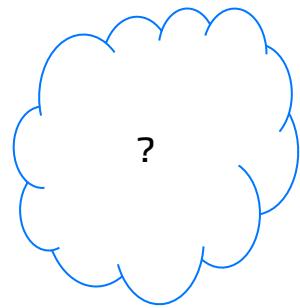
# TCP – 3 way handshake



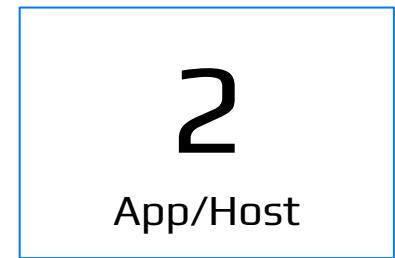
- Тратим время на установление соединения
- **Постоянное подтверждение и контроль потери пакетов**
- Прямое влияние **RTT** и **потерь** на скорость
- Можем подтверждать не каждый пакет, а некоторую последовательность
- Можем передавать вместе с SYN (TFO)



буферы  
и  
очереди

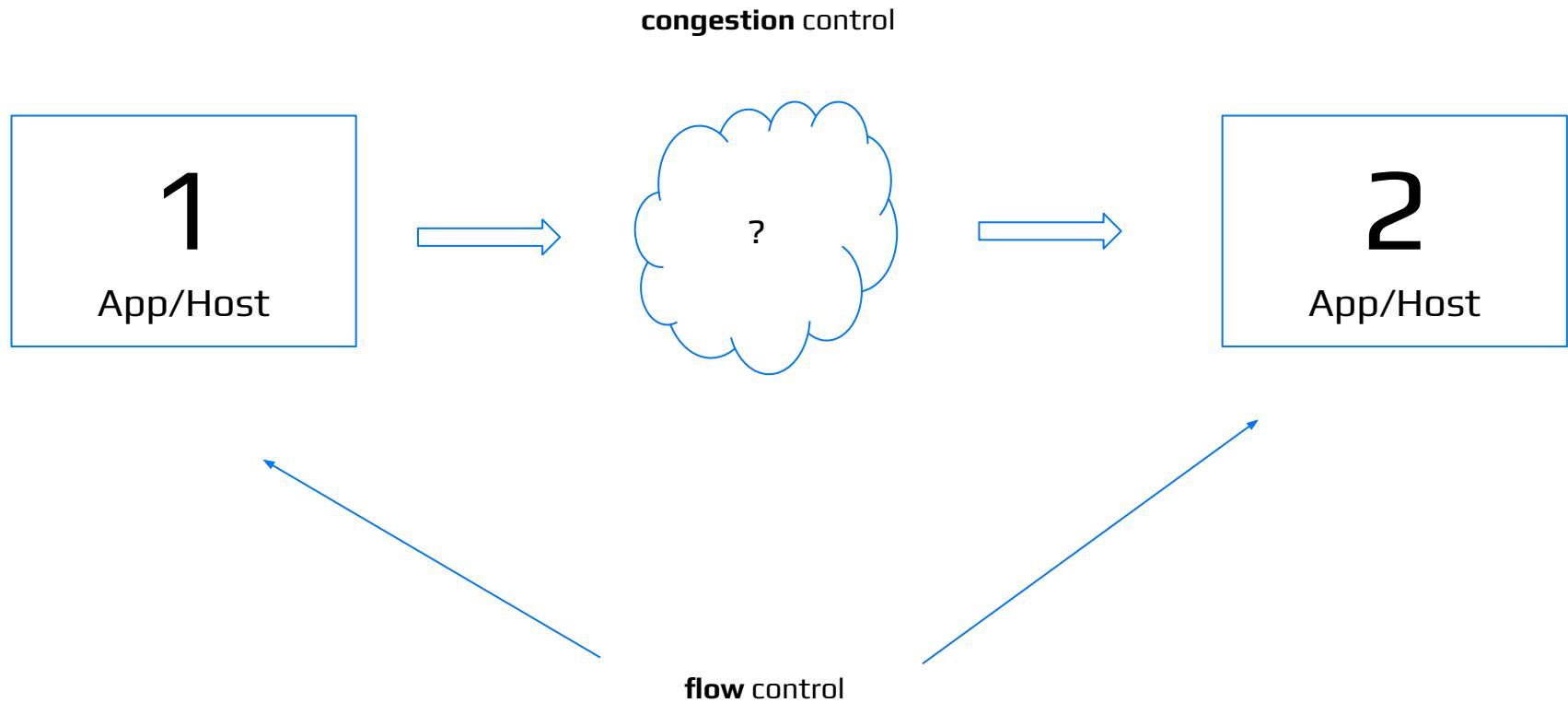


буферы  
и  
очереди



буферы  
и  
очереди

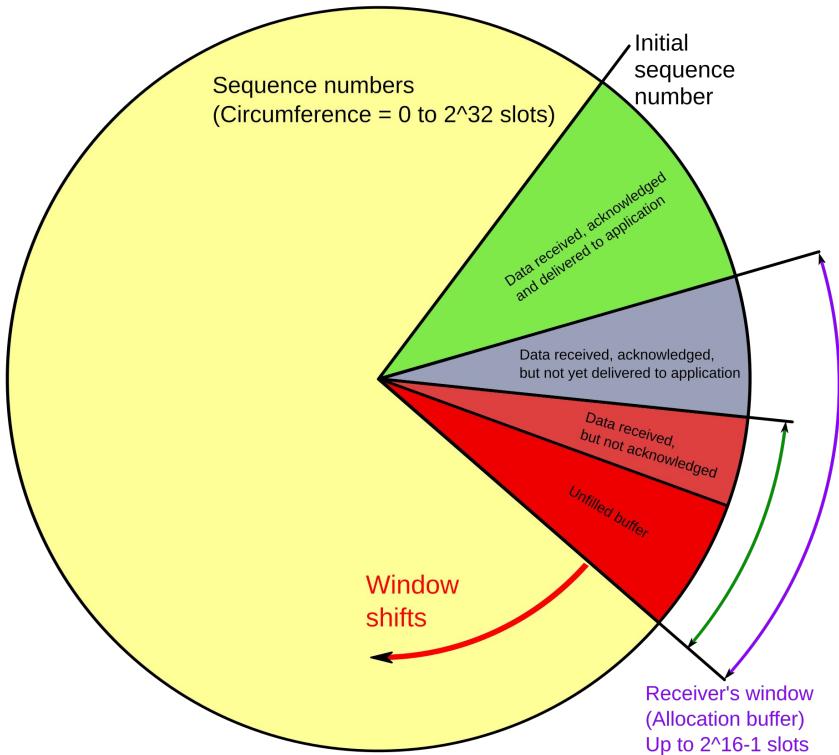




# TCP – Flow Control

- Отправитель и получатель должны работать “синхронно”
- Отправитель и получатель сообщают друг-другу о состоянии своих буферов
- Количество байт, которое может быть передано управляет размером “окна”
- Размер окна динамически меняется

# TCP – Flow Control



Данные подтверждённые (получен ACK) и переданные приложению

Данные подтверждённые (получен ACK) но **НЕ** переданные приложению

Данные полученные, но **НЕ** подтверждённые

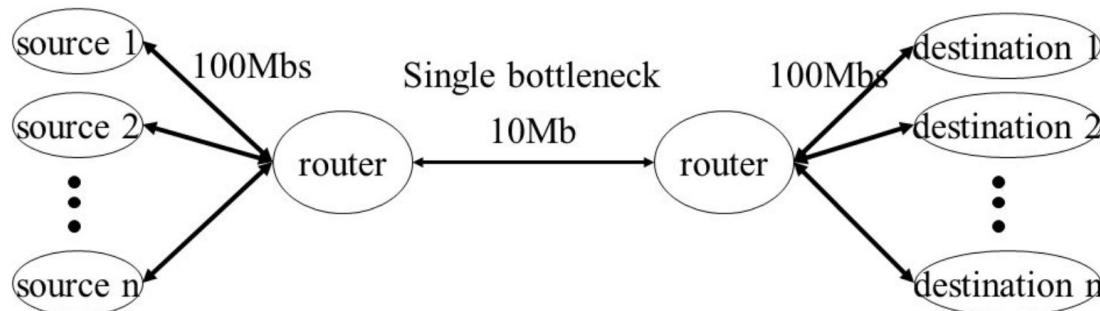
# TCP – Congestion Control

- Мы установили соединение (трубу)
- С какой скоростью можно передавать данные через это соединение, не теряя пакеты?

# TCP – Congestion Control

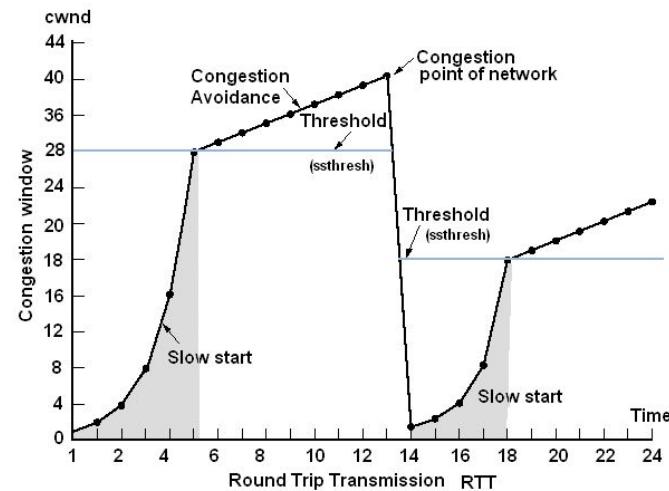
- Мы установили соединение (трубу)
- С какой скоростью можно передавать данные через это соединение, не теряя пакеты?

Со скоростью самого медленного сегмента



# TCP – Congestion Control

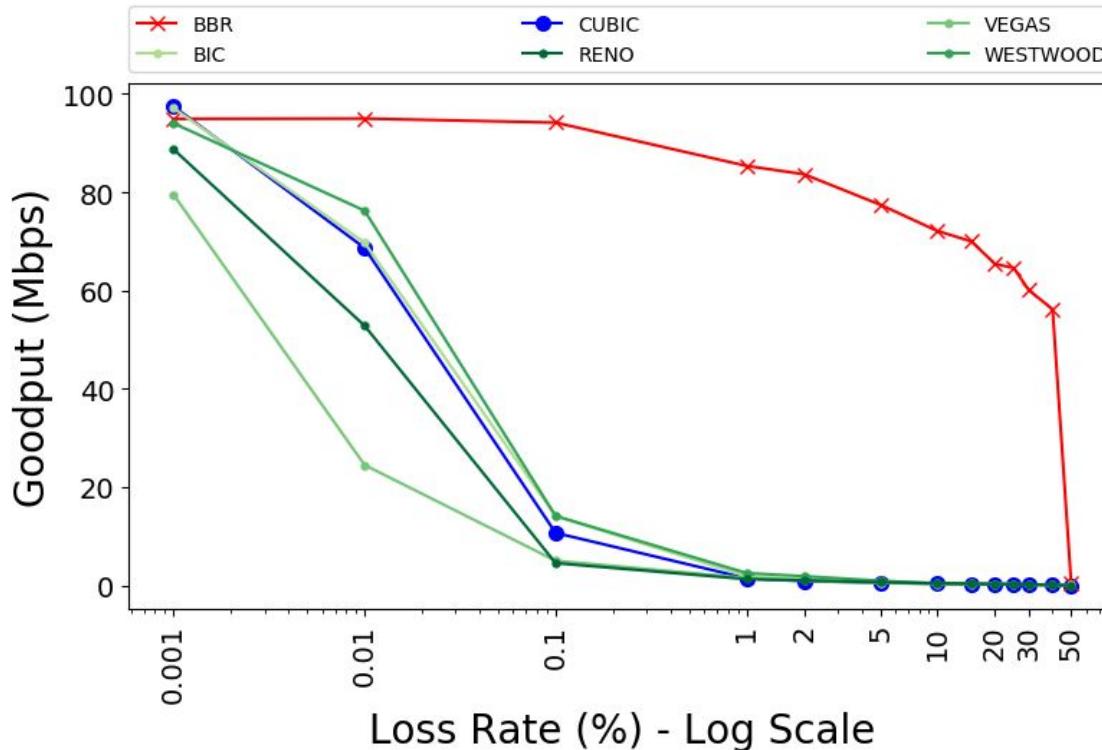
- Управляем CWND (congestion window)
- CWND определяет сколько данных может быть “в проводе”
- Масса алгоритмов, общая идея:
  - slow start
  - congestion avoidance
  - congestion detection



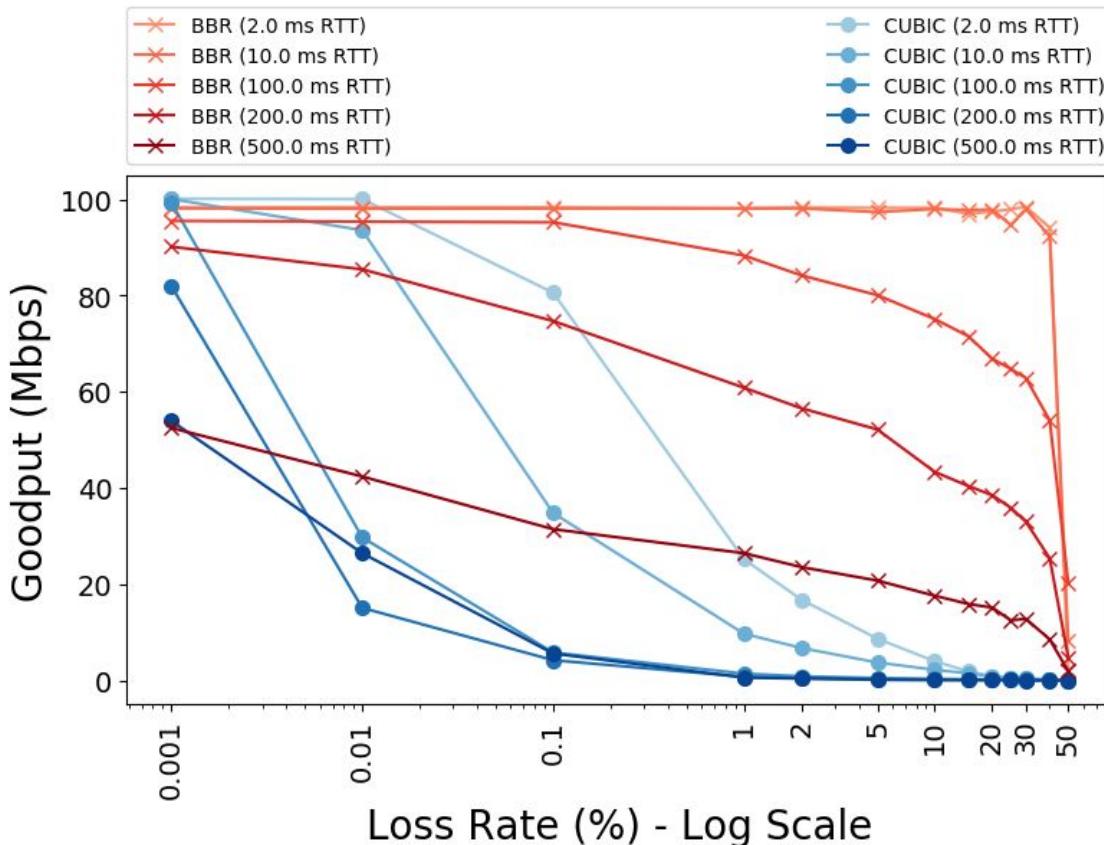
# TCP – Congestion Control

-> go to vpc

# TCP – Congestion Control



# TCP – Congestion Control



# TCP – Congestion Control

Реальный большой нагруженный сервер в интернете на десятки гигабит

```
vd504.okcdn.ru has address 185.226.55.14
```

```
Service:          VIDEO
Group:           video_download
```

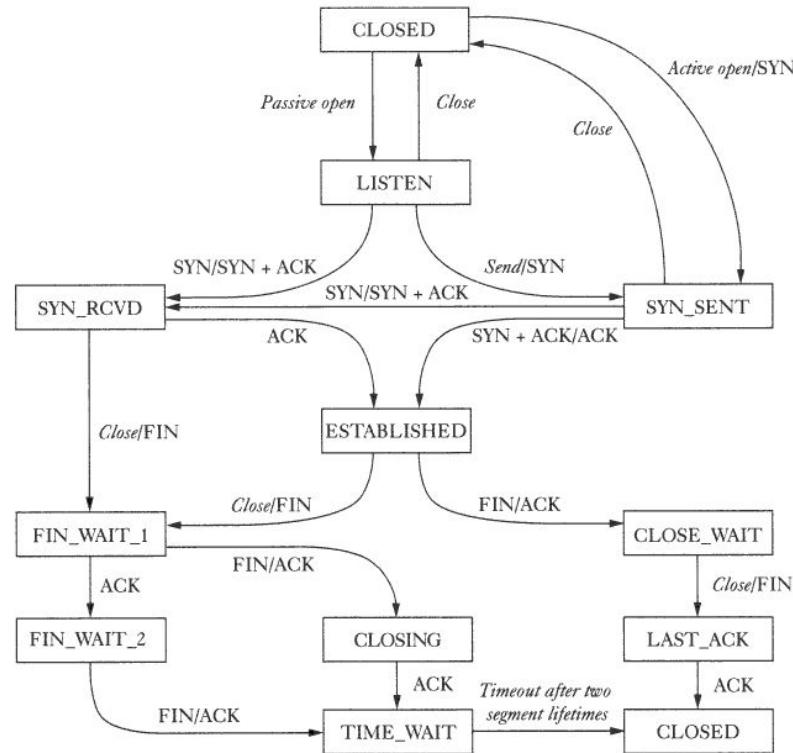
```
[root@host ~]# sysctl -a | grep congestion_control
```

```
net.ipv4.tcp_allowed_congestion_control = reno cubic bbr
net.ipv4.tcp_available_congestion_control = reno cubic bbr
net.ipv4.tcp_congestion_control = bbr
```

[TCP BBR: быстрый и простой способ ускорения загрузки страниц. Доклад Яндекса / Хабр \(habr.com\)](#)

<https://habr.com/ru/companies/yandex/articles/533530/>

# TCP State machine



# TCP

```
[root@host ~]# sysctl -a | grep udp | wc -l  
12
```

```
[root@host ~]# sysctl -a | grep tcp | wc -l  
109
```

```
[root@host ~]# sysctl -a | grep tcp | head -5  
  
net.ipv4.tcp_abort_on_overflow = 0  
net.ipv4.tcp_adv_win_scale = 1  
net.ipv4.tcp_allowed_congestion_control = reno cubic bbr  
net.ipv4.tcp_app_win = 31  
net.ipv4.tcp_autocorking = 1
```

# Где-то в Google в середине нулевых



Неолурк

[https://neolurk.org/wiki/Фатальный\\_недостаток](https://neolurk.org/wiki/Фатальный_недостаток) · Translate this page

## Фатальный недостаток

Jun 15, 2024 — Фатальный недостаток — локальный мем айтишной тусовки. Вкратце означает, что кто-то от нежелания платить за авторские права, ЧСВ, зависти, скуч ...



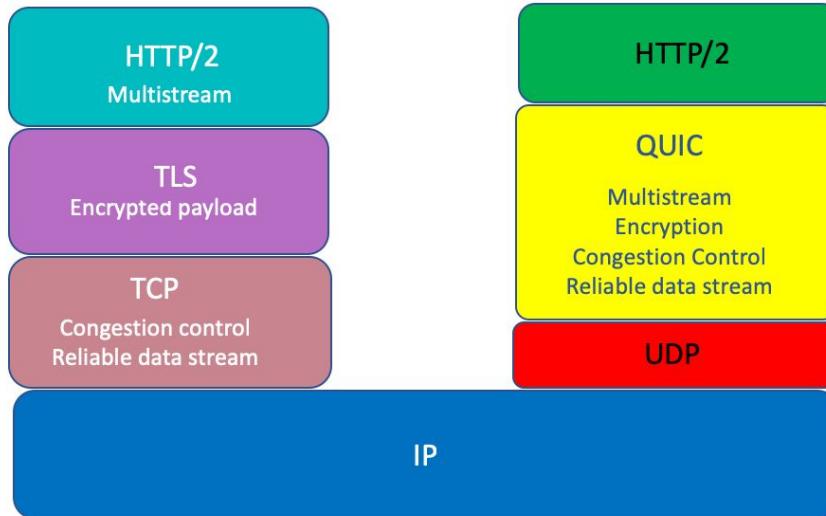
Википедия

[https://ru.wikipedia.org/wiki/Синдром\\_неприятия\\_чужой\\_разработки](https://ru.wikipedia.org/wiki/Синдром_неприятия_чужой_разработки) · Translate this page

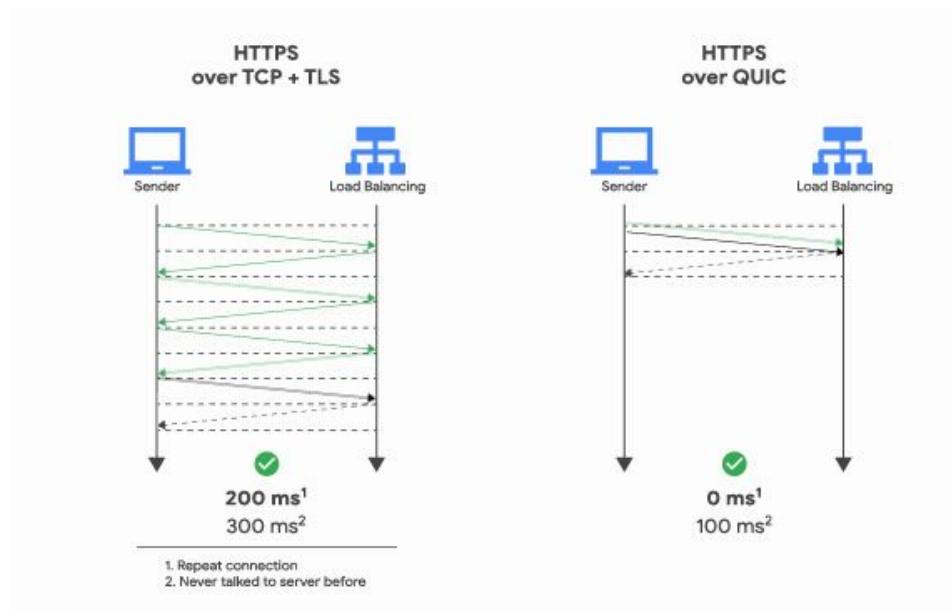
## Синдром неприятия чужой разработки

Как социальное явление, эта философия проявляется нежеланием принять идею или продукт, потому что тот происходит из другой культуры, форма трайбализма. Термин ...

# QUIC



# QUIC

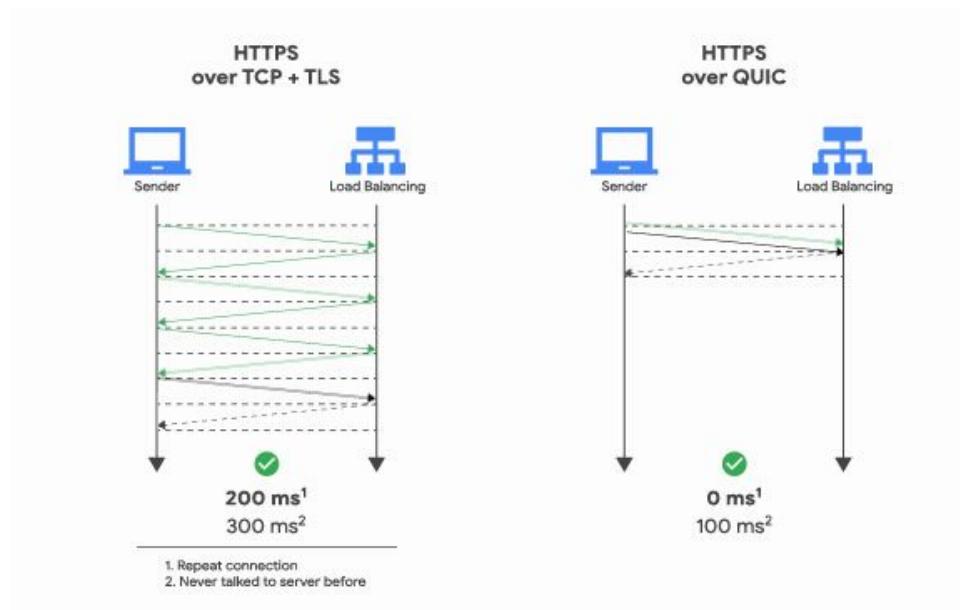


- Для плохих сетей
- Для динамичных подключений



???

# QUIC

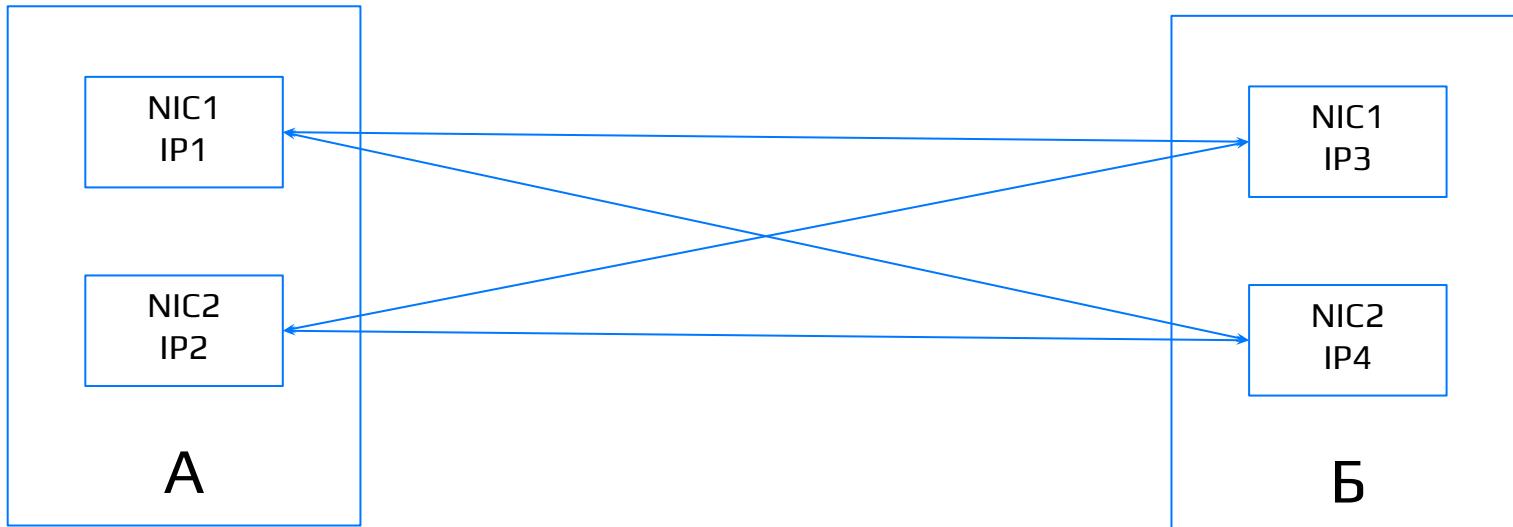


- Для плохих сетей
- Для динамичных подключений



Мобильный интернет

# SCTP

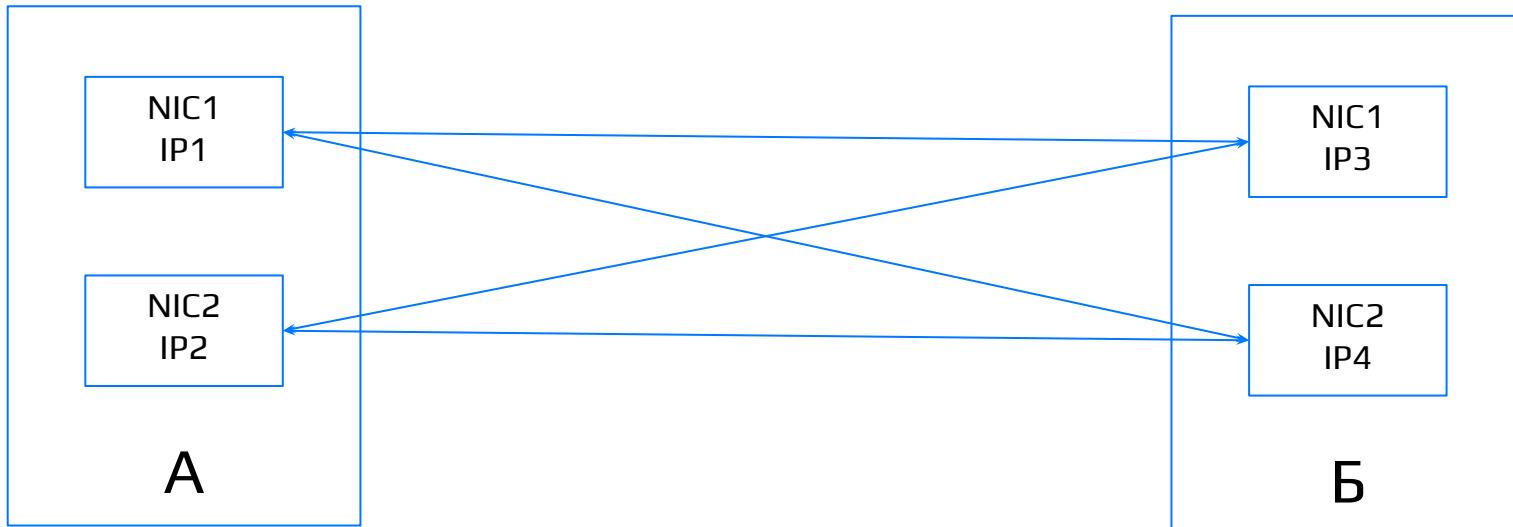


- Надёжно
- Быстро
- Статично



???

# SCTP

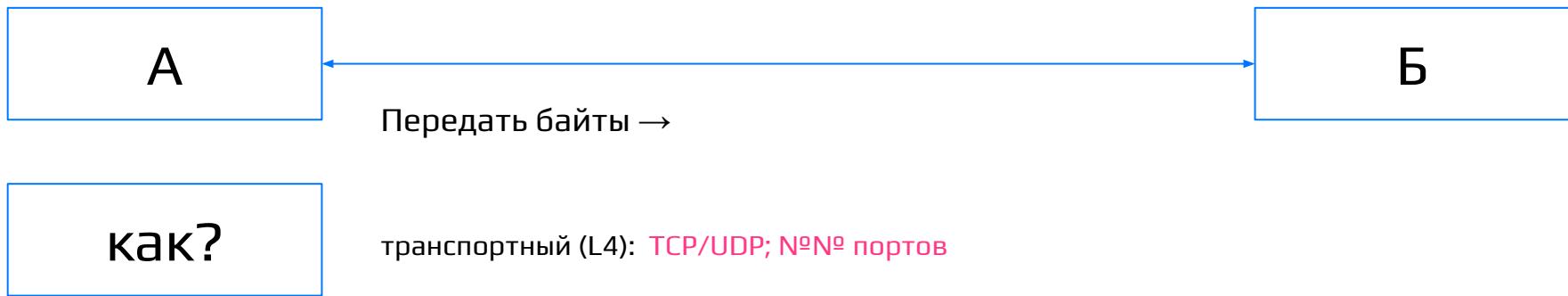


- Надёжно
- Быстро
- Статично

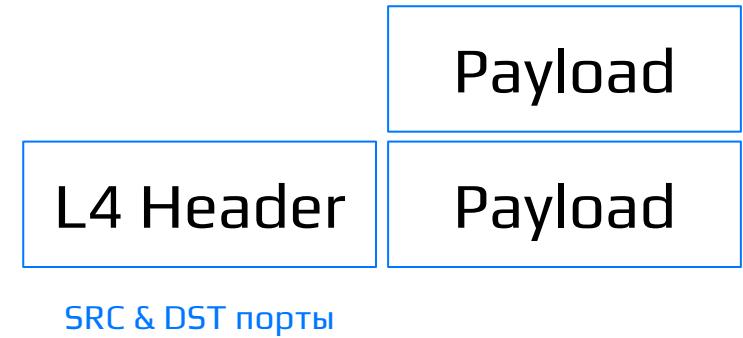


Телеком

# Знаем как отправляем



## Пакет уровня L4



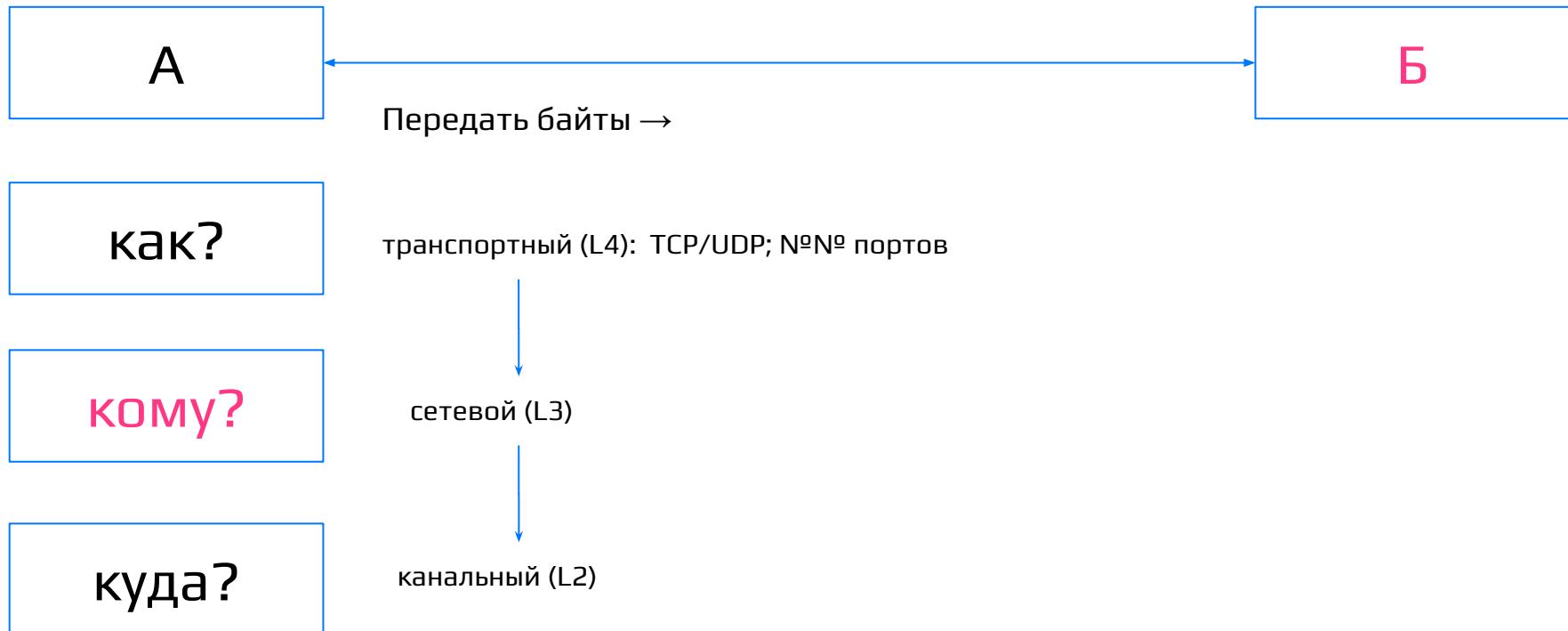


# Вопросы?



- L4 – транспортный
- L3 – сетевой
- L2 – канальный

# Кому отправляем?



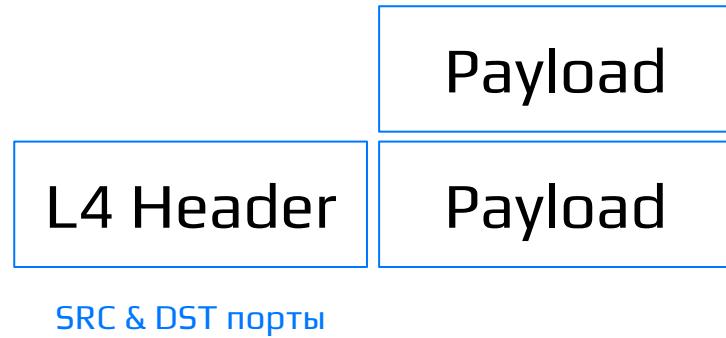
# Кому отправляем?



→ ~ host corp.vkcdn.ru  
corp.vkcdn.ru has address **5.181.61.0**

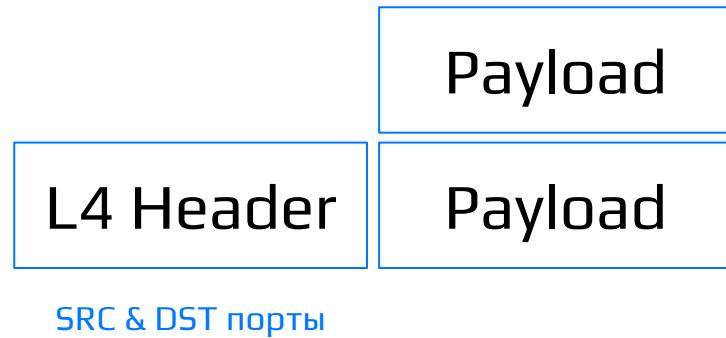
→ ~ host ns1.ok.ru  
ns1.ok.ru has address 5.61.23.9  
ns1.ok.ru has IPv6 address **2a00:b4c0:1b0:6::**

## Пакет уровня L4



- Но здесь нет информации про L3 адреса
- Нужно добавить некую описательную часть (снова!)

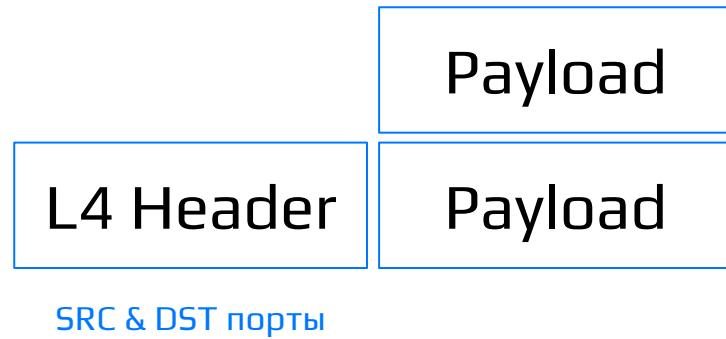
## Пакет уровня L4



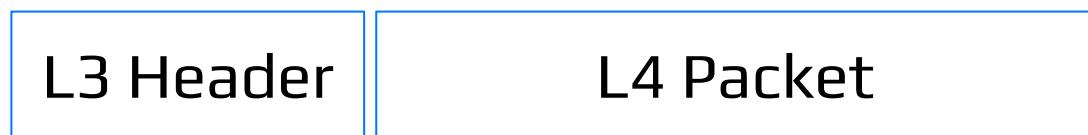
- Нужно добавить некую описательную часть (снова!)



## Пакет уровня L4

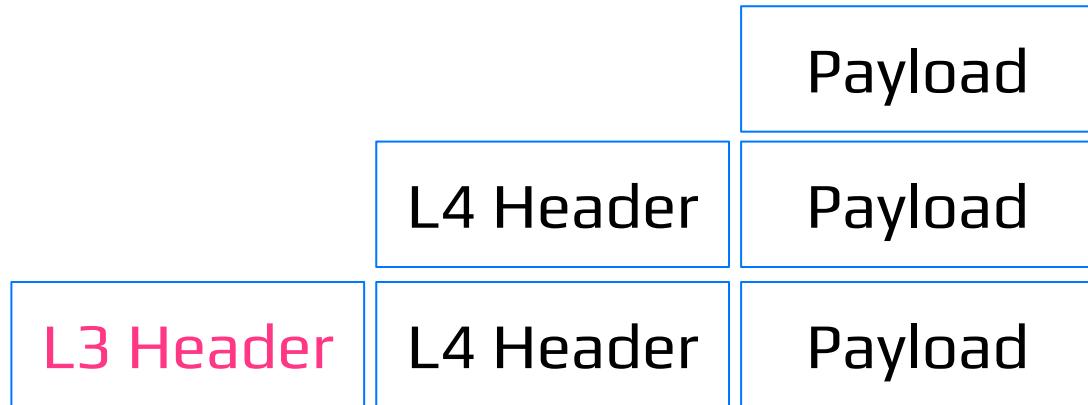


- Нужно добавить некую описательную часть (снова!)



- Инкапсуляция

L3



L4 протокол  
+  
IP адреса

Порты

# IPv4 заголовок

## IPv4 header format

# IPv4 адреса и сети

- **IP адрес** это просто число из 32 бит = 4 октета
- **Подсеть** это группа IP адресов

$10.0.0.1/24 = 10.0.0.1/255.255.255.0$	=254 адреса
$10.0.0.1/28 = 10.0.0.1/255.255.255.240$	=14 адресов

Логическое разделение сети на сегменты

# IPv4 адреса и сети

- **IP-адрес:** 192.168.1.2
- **Маска подсети:** 255.255.255.0

В двоичном виде это выглядит так:

- **IP-адрес:** 11000000.10101000.00000001.00000010
- **Маска подсети:** 11111111.11111111.11111111.00000000

Адрес сети получается путем побитового И (AND) между IP-адресом и маской подсети:

- **Адрес сети:** 192.168.1.0 (11000000.10101000.00000001.00000000)

# IPv4 адреса и сети

Минимальная подсеть?

-

# IPv4 адреса и сети

Минимальная подсеть?

- $/31 = 2$  адреса = p2p сеть (ну или  $/32 == 1$  адрес)

# IPv4 адреса и сети

Минимальная подсеть?

- $/31 = 2$  адреса = p2p сеть (ну или  $/32 == 1$  адрес)

Может ли быть адрес 10.0.0.0 ?

-

# IPv4 адреса и сети

Минимальная подсеть?

- $/31 = 2$  адреса = p2p сеть (ну или  $/32 == 1$  адрес)

Может ли быть адрес 10.0.0.0 ?

- да, почему бы и нет, смотря в какой сети

# IPv4 адреса и сети

Минимальная подсеть?

- $/31 = 2$  адреса = p2p сеть (ну или  $/32 == 1$  адрес)

Может ли быть адрес 10.0.0.0 ?

- да, почему бы и нет, смотря в какой сети

Что значит 0/0 или 0.0.0.0/0?

-

# IPv4 адреса и сети

Минимальная подсеть?

- $/31 = 2$  адреса = p2p сеть (ну или  $/32 == 1$  адрес)

Может ли быть адрес 10.0.0.0 ?

- да, почему бы и нет, смотря в какой сети

Что значит 0/0 или 0.0.0.0/0?

- весь диапазон IPv4

# IPv4 адреса и сети

Минимальная подсеть?

- $/31 = 2$  адреса = p2p сеть (ну или  $/32 == 1$  адрес)

Может ли быть адрес 10.0.0.0 ?

- да, почему бы и нет, смотря в какой сети

Что значит 0/0 или 0.0.0.0/0

- весь диапазон IPv4

1.1.1.1 и 1.0.0.1 в одной подсети?

-

# IPv4 адреса и сети

Минимальная подсеть?

- $/31 = 2$  адреса = p2p сеть (ну или  $/32 == 1$  адрес)

Может ли быть адрес 10.0.0.0 ?

- да, почему бы и нет, смотря в какой сети

Что значит 0/0 или 0.0.0.0/0

- весь диапазон IPv4

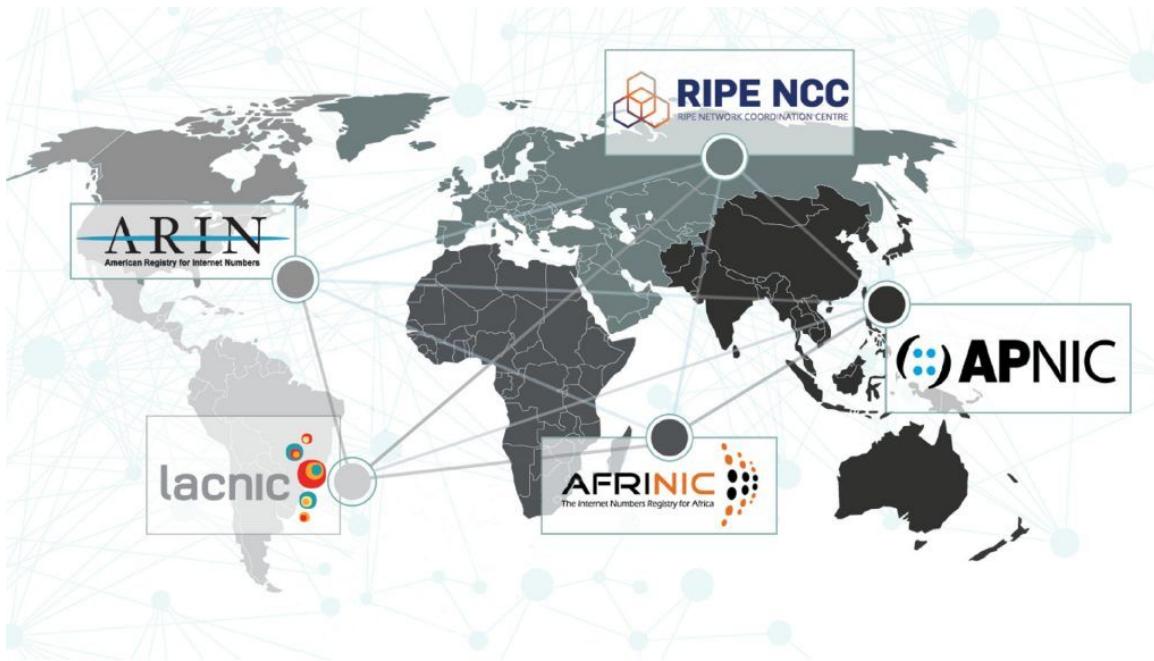
1.1.1.1 и 1.0.0.1 в одной подсети?

- 1.0.0.0/8

<https://ip-calculator.ru/>

<https://www.cidr.eu/en/calculator>

IANA → RIR → NIR → LIR



# IPv4 run out

<https://www.ripe.net/manage-ips-and-asns/ipv4/ipv4-run-out/>

When	Date	How IPv4 Requests Are Processed
When we have less than one /8 block of IPv4 addresses remaining	15/09/2012	Each LIR can receive one /22 IPv4 allocation <b>in the form of a single prefix</b>
Once we have no more /22 prefixes	02/10/2019	Each LIR can receive one /22 IPv4 allocation <b>in the form of multiple smaller prefixes (/23s and/or /24s)</b>
Once all available IPv4 address space is exhausted	25/11/2019	LIRs can enter a <a href="#">waiting list</a> to receive one <b>/24 IPv4 allocation</b> when addresses are returned in the future

# IPv6

- **IP адрес** это просто число из 128 бит =  $8 \times 16$  (хексстеты)
- **Подсеть** это группа IP адресов
- Вместо маски – префикс (хоть и выглядит так же)

# IPv6

## Префикс /64

Адрес: 2001:0db8:85a3:0000:0000:8a2e:0370:7334/64

- **Префикс:** 2001:0db8:85a3:0000::/64
- **Адрес узла:** 0000:0000:8a2e:0370:7334

## Префикс /48

Адрес: 2001:0db8:1234::/48

- **Префикс:** 2001:0db8:1234::/48
- **Диапазон адресов:** от 2001:0db8:1234:0000:0000:0000:0000 до 2001:0db8:1234:ffff:ffff:ffff:ffff:ffff1

# IPv6

→ ~ dig +short facebook.com AAAA

2a03:2880:f113:81:**face:b00c:0:25de**

**Google:** Один из их адресов - **2001:4860:4860::8888** - это адрес их публичного DNS-сервера.

Он легко запоминается благодаря повторяющимся цифрам.

**Cloudflare:** Их публичный DNS-сервер имеет адрес **2606:4700:4700::1111**, который также легко запомнить из-за повторяющихся единиц.

**Cisco:** Один из их адресов - **2001:420:80:1::5** - содержит последовательность, которая может быть легко запомнена.

**ya.ru has IPv6 address 2a02:6b8::2:242**

# IPv6

```
static final Pattern XX = Pattern.compile(  
"((25[0-5]|2[0-4][0-9]| [01]?[0-9][0-9]?)\\.){3}(25[0-5]|2[0-4][0-9]| [01]?[0-9][0-9]?) / (3[0-2]  
|[12]?[0-9])");  
  
static final Pattern YY = Pattern.compile(XX.pattern() + "(/\d{1,2})?");
```

# IPv6

```
static final Pattern XX = Pattern.compile(  
"(([0-9a-fA-F]{1,4}):){7,7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}):{1,7}:|([0-9a-fA-F]{1,4}):{1,6}  
:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}):{1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}):{1,4}(:  
[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}):{1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}):{1  
,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:((:[0-9a-fA-F]{1,4}){1,6}))|:(([0-9a-fA-F]{1,4})  
{1,7}|:)|fe80:(:[0-9a-fA-F]{0,4}){0,4}%[0-9a-zA-Z]{1,}|::(fffff(:0{1,4}){0,1}):{0,1}|((25[0-5]|  
(2[0-4]|1{0,1}[0-9]){0,1}[0-9])\\.){3,3}(25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9])|([0-9a-fA-F]  
{1,4}):{1,4}|((25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9])\\.){3,3}(25[0-5]|(2[0-4]|1{0,1}[0-9]){|  
0,1}[0-9)))");  
  
static final Pattern YY = Pattern.compile(XX.pattern() + "(/\\d{1,2})?");
```

# IPv6

IPv6 Header

Version	Traffic Class	Flow Label	
Payload Length		Next Header	Hop Limit
Source Address			
Destination Address			

IPv4 Header

Version	IHL	Type of Service	Total Length		
Identification		Flags	Fragment Offset		
TTL	Protocol	Header Checksum			
Source Address					
Destination Address					
Options		Padding			

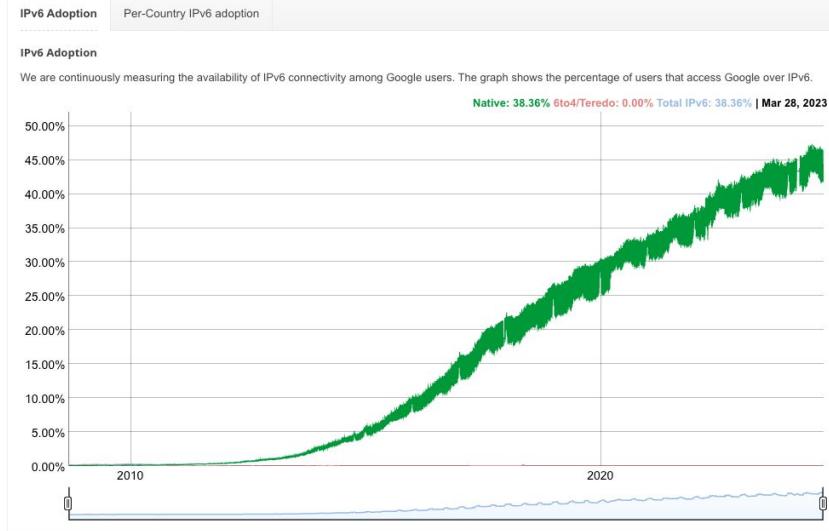
## Legend

- Fields **kept** in IPv6
- Fields **kept** in IPv6, but name and position changed
- Fields **not kept** in IPv6
- Fields that are **new** in IPv6

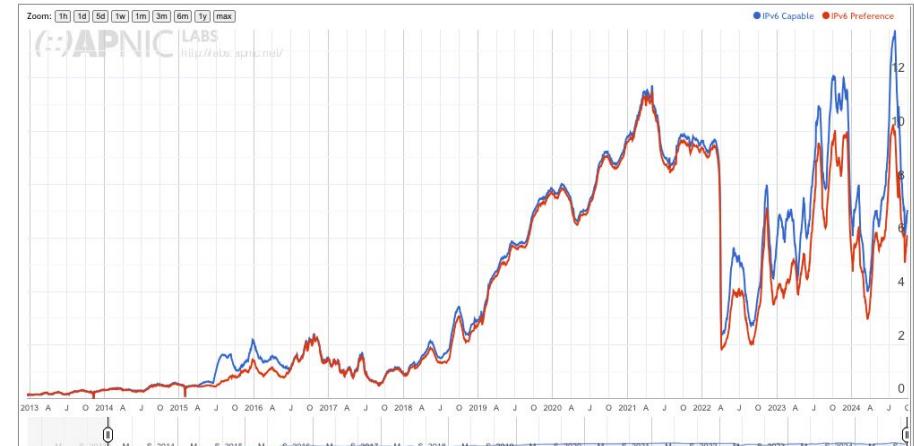
# Google IPv6

## Statistics

Google collects statistics about IPv6 adoption in the Internet on an ongoing basis. We hope that publishing this information will help Internet providers, website owners, and policy makers as the industry rolls out IPv6.



## Use of IPv6 for Russian Federation (RU)



<https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption>

<https://stats.labs.apnic.net/ipv6/RU>

## Суммарный трафик участников, передаваемый через MSK-IX

За сутки   За неделю   За месяц   **За год**

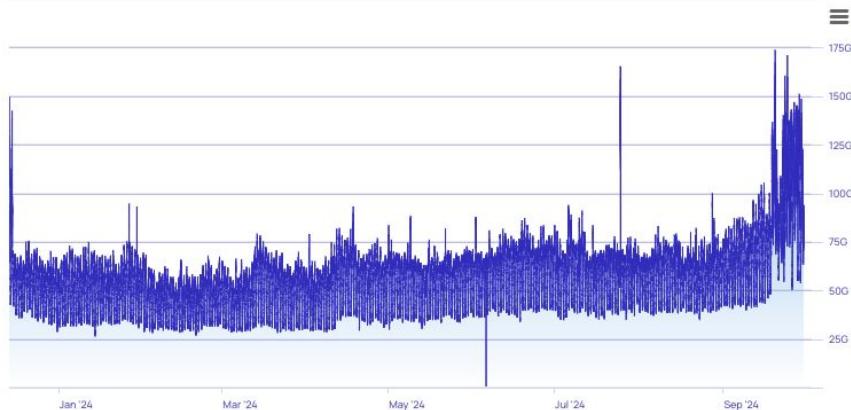


Average: 5226 Gbps   Maximum: 6550 Gbps

Последнее обновление 30.09.2024 07:55

## Трафик IPv6

За сутки   За неделю   За месяц   **За год**



# ICMP

- Технический протокол для работы сети
- Реализует массу **типов**, например
  - эхо реквест / эхо реплай (пинг)
  - доставимость хоста
  - TTL

13:10:33.252496 IP **172.31.255.1** > **100.64.156.96**: ICMP echo request, id 51674, seq 25443, length 64  
13:10:33.252532 IP **100.64.156.96** > **172.31.255.1**: ICMP echo reply, id 51674, seq 25443, length 64

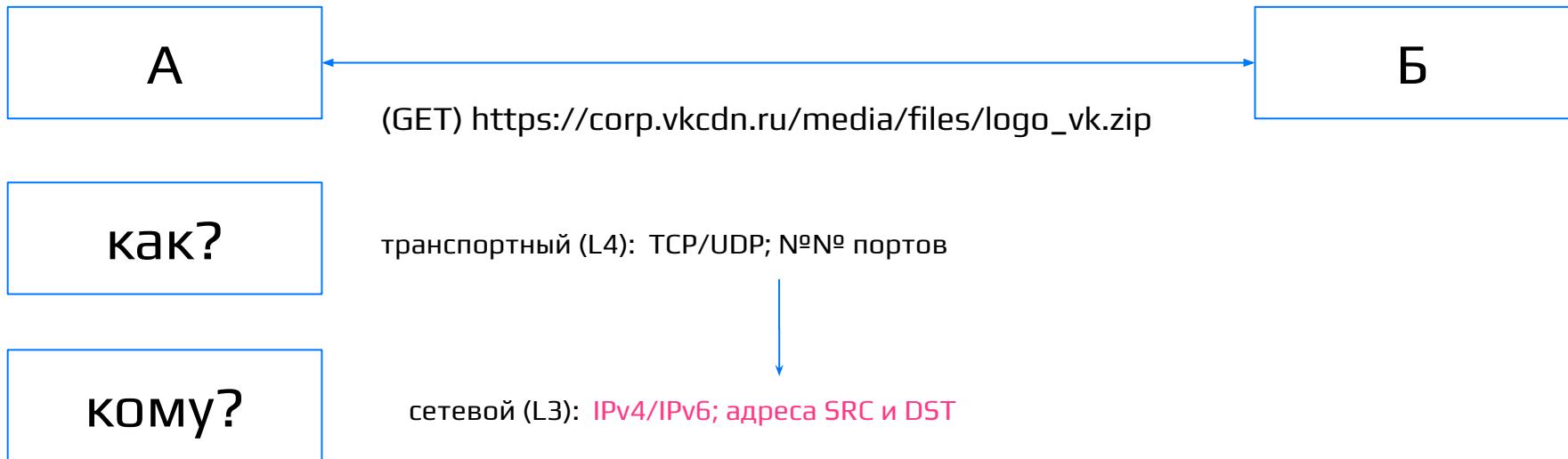


# Вопросы?

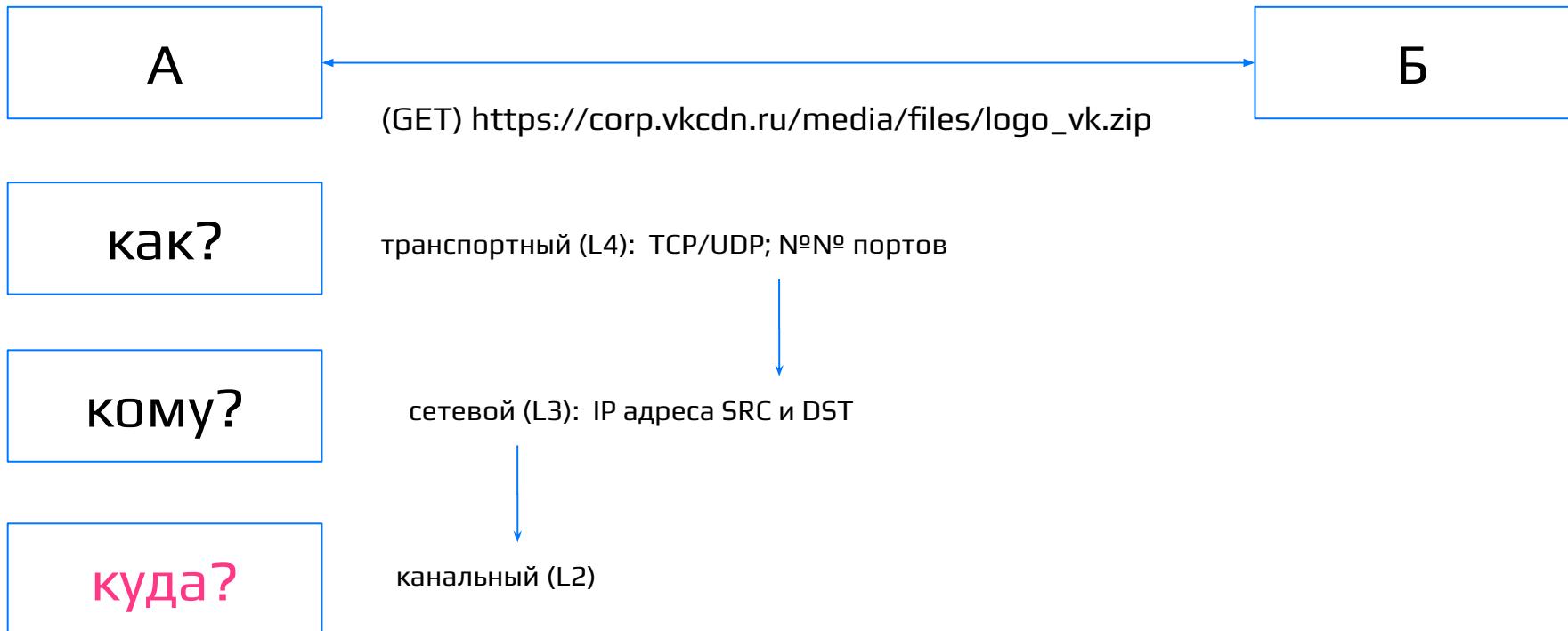


- L4 – транспортный
- L3 – сетевой
- L2 – канальный

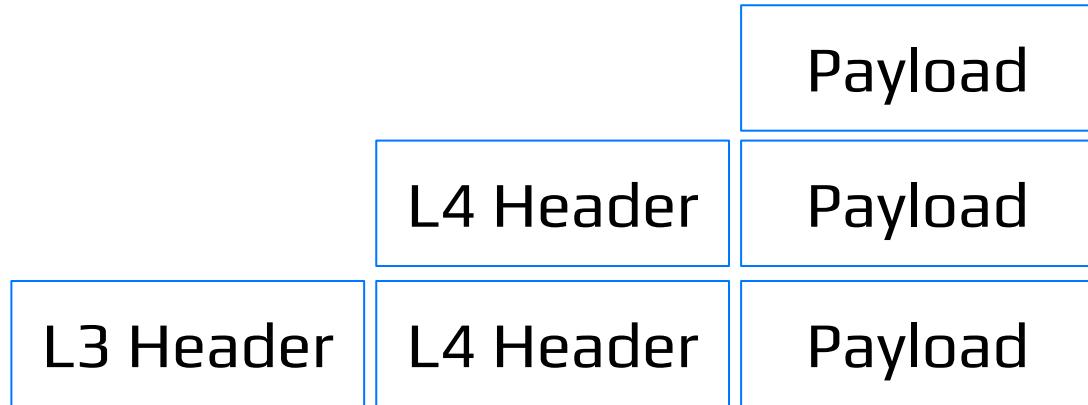
# Знаем кому отправляем



# Куда отправляем?



L3

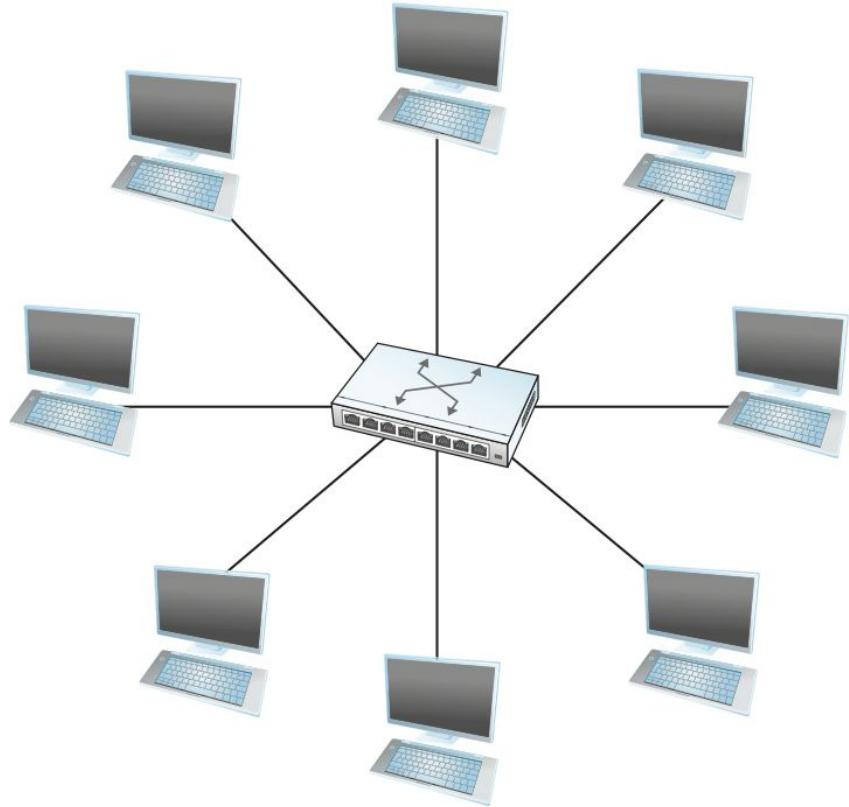


L4 протокол  
+  
SRC IP адрес,  
DST IP адрес

Порты

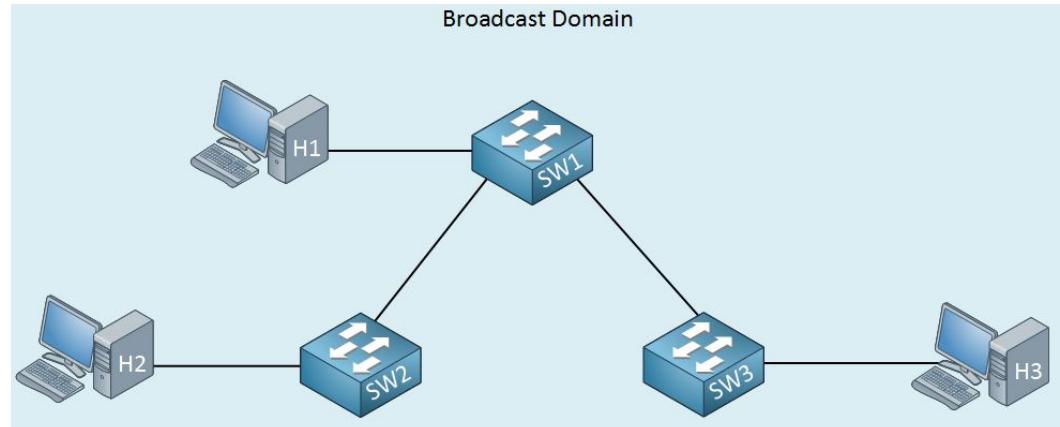
# Ethernet

- Каждый “слышит” каждого
- Широковещательный L2 сегмент
- Нужно идентифицировать получателя
- MAC адрес (Medium Access Control)



# Ethernet

- Каждый “слышит” каждого
  - Широковещательный L2 сегмент
- 
- Нужно идентифицировать получателя
  - MAC адрес (Medium Access Control)

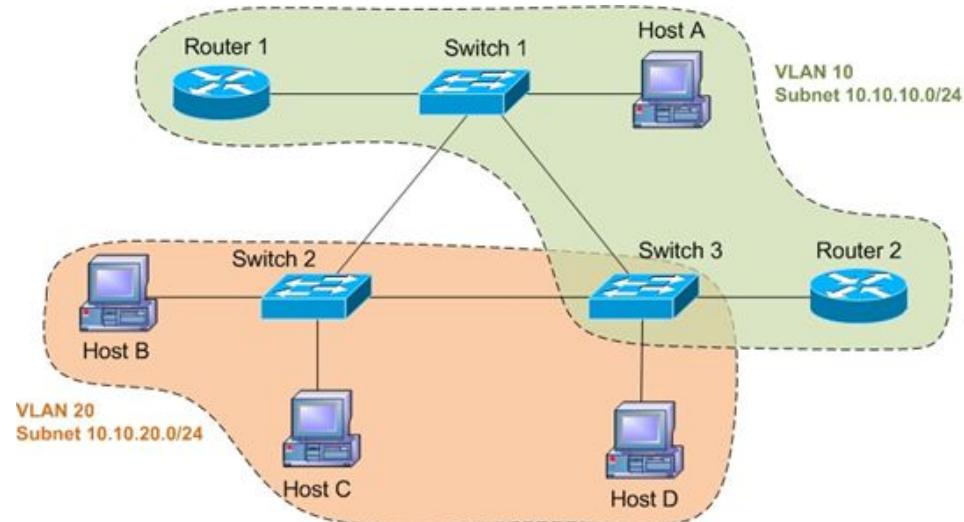


# MAC адрес

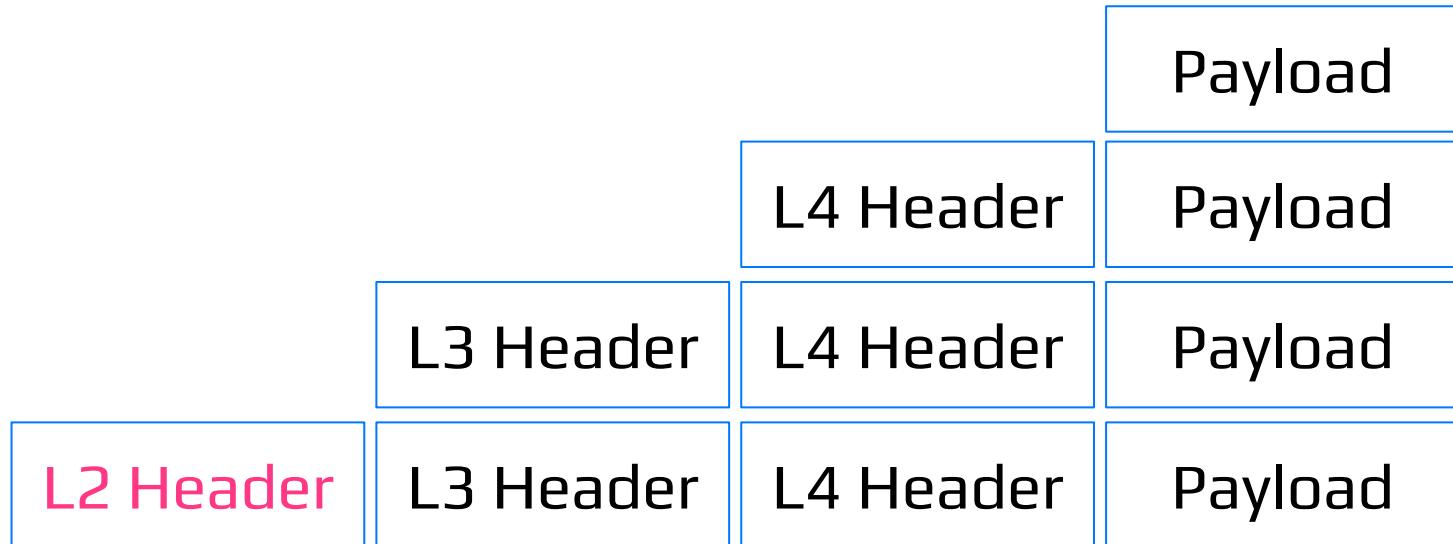
- Уникальный идентификатор сетевого устройства
  - 48 битное число (50:64:2b:5a:ca:47)
- Первые 24 бита это OUI (идентификатор производителя)

# VLAN

- Логический сегмент сети
- Изоляция



L2

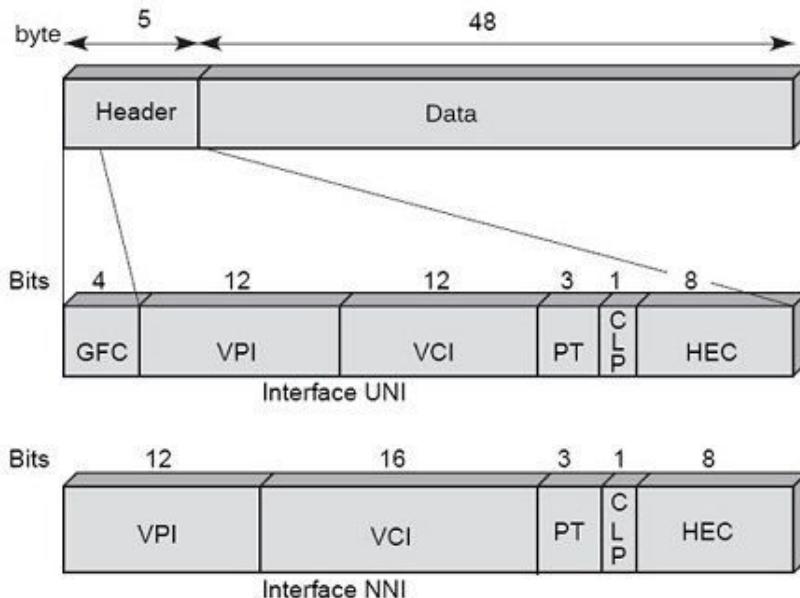


L3 протокол  
+  
MAC адреса

L4 протокол  
+  
IP адреса

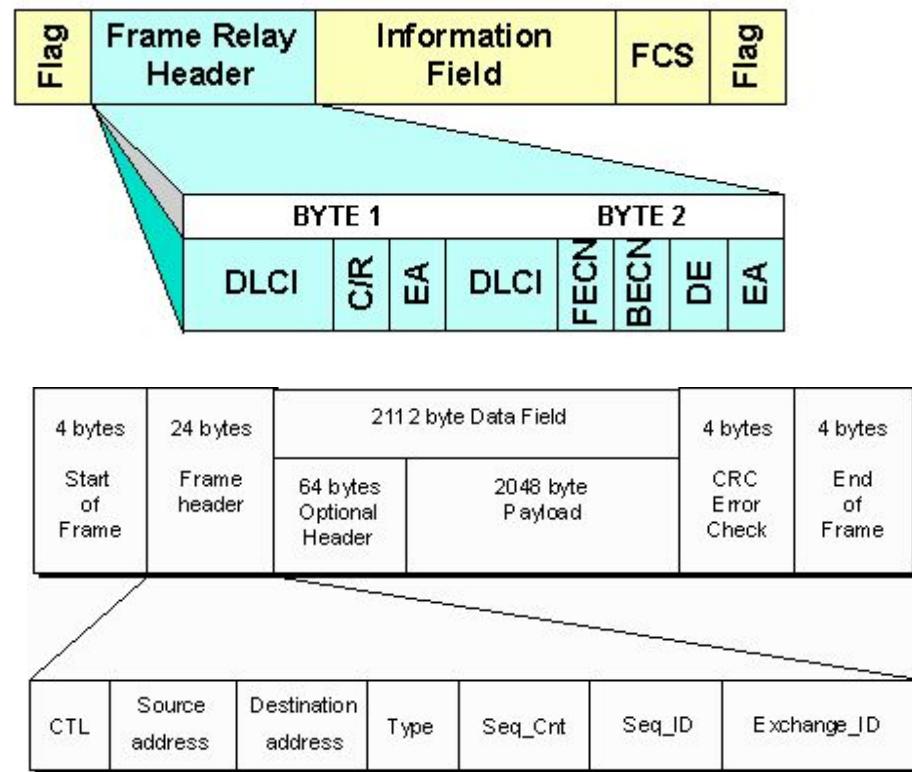
Порты

# Вообще-то бывает и по-другому



GFC (Generic Flow Control)  
VCI (Virtual Channel Identifier)  
VPI (Virtual Path Identifier)  
PT (Payload Type)  
CLP (Cell Loss Priority)

Format of the ATM Cell Header



# Ethernet

Layer	Preamble	Start frame delimiter (SFD)	MAC destination	MAC source	802.1Q tag (optional)	Ethertype (Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32-bit CRC)	Interpacket gap (IPG)
Length (octets)	7	1	6	6	(4)	2	42–1500 <sup>[c]</sup>	4	12
Layer 2 Ethernet frame	(not part of the frame)		← 64–1522 octets →						(not part of the frame)

# Определяем DST MAC

- DST IP и SRC IP **в одной подсети** (connected)



- MAC DST = MAC устройства, где настроен IP DST

# ARP

- Понять какому MAC адресу соответствует IP

13:23:06.713733 ARP, Request **who-has 10.0.0.2 tell 10.0.0.75**, length 28

13:23:06.715716 ARP, Reply **10.0.0.1 is-at 50:64:2b:5a:ca:47**, length 28

- Широковещательный запрос

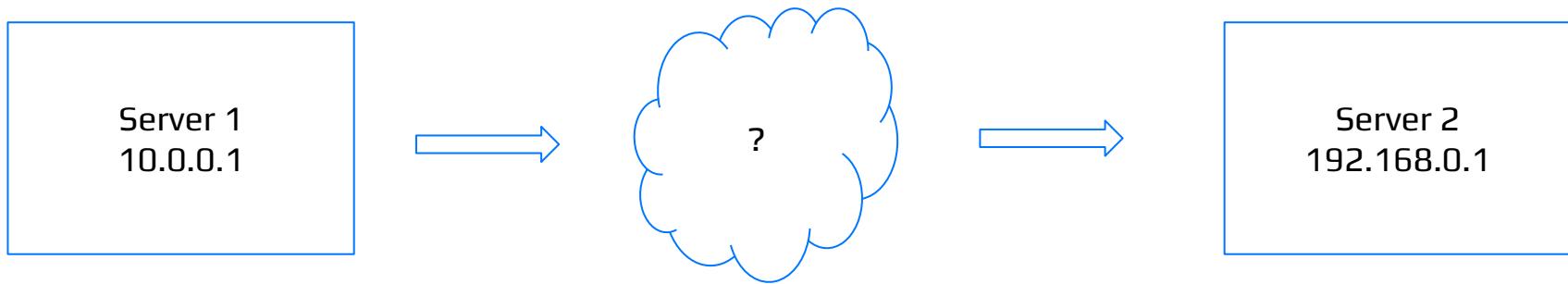
# ARP

arp

No.	Time	Source	Destination	Protocol	Length	Info
55	1.608033	Routerboardc_6f:4e:39	Broadcast	ARP	56	Who has 10.171.75.118? Tell 10.171.75.1
56	1.665301	de:b1:de:b1:29:76	Broadcast	ARP	42	Who has 10.171.75.193? Tell 10.171.75.24
162	4.356583	MoxaTechnolo_29:ce:b8	Broadcast	ARP	60	Who has 10.171.75.1? Tell 10.171.75.12
164	4.453166	MoxaTechnolo_29:ce:b8	Broadcast	ARP	60	ARP Announcement for 10.171.75.12
178	4.621014	c6:18:a1:73:dc:ac	Broadcast	ARP	56	ARP Announcement for 10.171.75.208
180	4.757046	c6:18:a1:73:dc:ac	Broadcast	ARP	56	Who has 10.171.75.1? Tell 10.171.75.208
183	4.804943	Routerboardc_6f:4e:39	Broadcast	ARP	56	Who has 10.171.75.118? Tell 10.171.75.1
184	4.856796	MoxaTechnolo_29:ce:b8	Broadcast	ARP	60	Who has 10.171.75.1? Tell 10.171.75.12
217	5.638825	c6:18:a1:73:dc:ac	Broadcast	ARP	56	Who has 10.171.75.1? Tell 10.171.75.208
228	5.789172	ba:1e:08:2f:a1:ee	Broadcast	ARP	42	Who has 10.171.75.53? Tell 10.171.75.75
234	5.872370	Routerboardc_6f:4e:39	Broadcast	ARP	56	Who has 10.171.75.118? Tell 10.171.75.1
238	5.955711	c6:18:a1:73:dc:ac	Broadcast	ARP	56	ARP Announcement for 10.171.75.208
284	6.912952	Routerboardc_6f:4e:39	Broadcast	ARP	56	Who has 10.171.75.118? Tell 10.171.75.1
315	7.518517	Apple_41:2a:b9	4e:d9:8c:c1:... ARP		42	Who has 10.171.75.10? Tell 10.171.75.140
317	7.523574	4e:d9:8c:c1:39:a5	Apple_41:2a:... ARP		56	10.171.75.10 is at 4e:d9:8c:c1:39:a5
331	7.818434	a2:4a:25:f6:12:bd	Broadcast	ARP	42	Who has 10.171.75.1? Tell 10.171.75.97
346	8.249780	KYOCERADispl_72:47:83	Broadcast	ARP	60	Gratuitous ARP for 10.171.75.234 (Request)
392	9.072493	Routerboardc_6f:4e:39	Broadcast	ARP	56	Who has 10.171.75.118? Tell 10.171.75.1
412	9.713709	Intel_35:87:70	Broadcast	ARP	42	Who has 10.171.75.1? Tell 10.171.75.29
417	9.749384	Intel_35:87:70	Broadcast	ARP	42	Who has 10.171.75.29? (ARP Probe)
443	10.7272...	Intel_35:87:70	Broadcast	ARP	42	Who has 10.171.75.29? (ARP Probe)
468	11.7320...	Intel_35:87:70	Broadcast	ARP	42	Who has 10.171.75.29? (ARP Probe)
500	12.2729...	Routerboardc_6f:4e:39	Broadcast	ARP	56	Who has 10.171.75.118? Tell 10.171.75.1
521	12.7305...	Intel_35:87:70	Broadcast	ARP	42	ARP Announcement for 10.171.75.29

# Определяем DST MAC

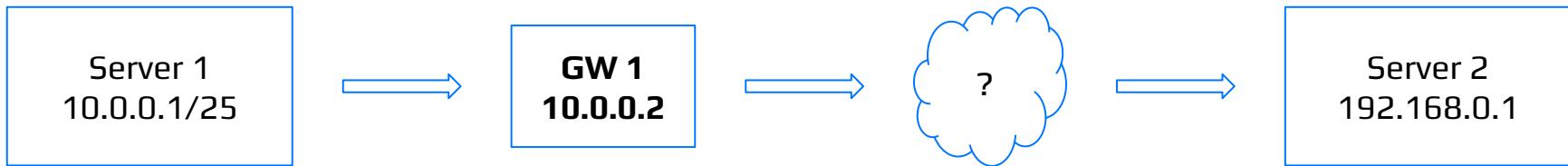
- DST IP и SRC IP **в разных подсетях** (НЕ connected)



- MAC DST = MAC устройства **в том же L2 сегменте**, которое *предположительно* знает как достичь DST IP

# Статическая маршрутизация

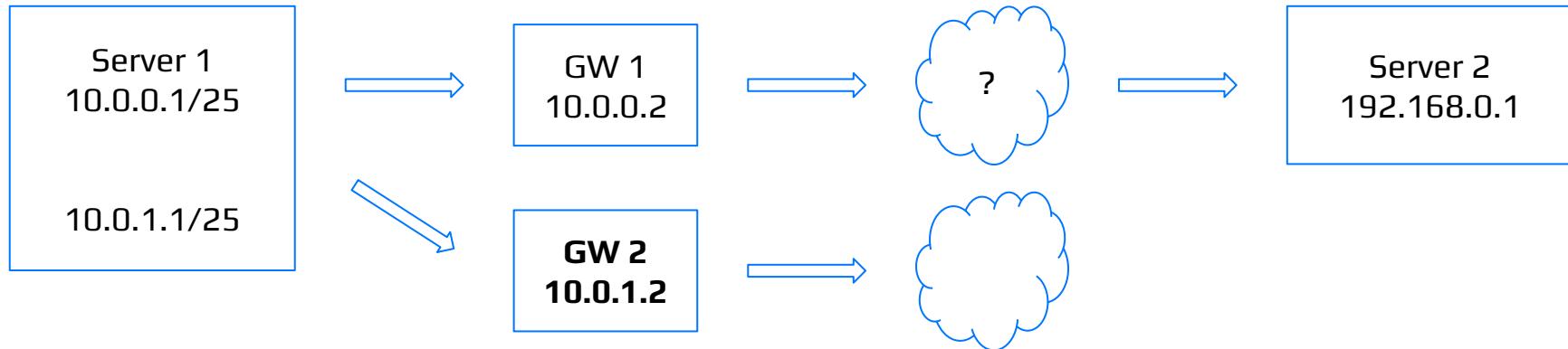
```
[root@host ~]# ip r  
default via 10.0.1.2 dev eth0  
192.168.0.0/16 via 10.0.0.2 dev eth1
```



- MAC DST = MAC **шлюза / маршрутизатора**

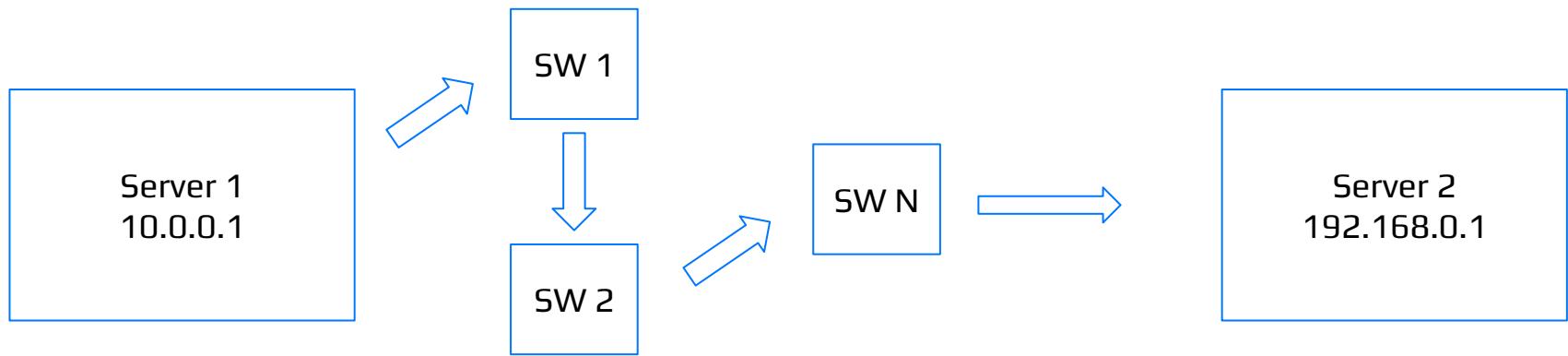
# Статическая маршрутизация

```
[root@host ~]# ip r
default via 10.0.1.2 dev eth0
192.168.0.0/16 via 10.0.0.2 dev eth1
```

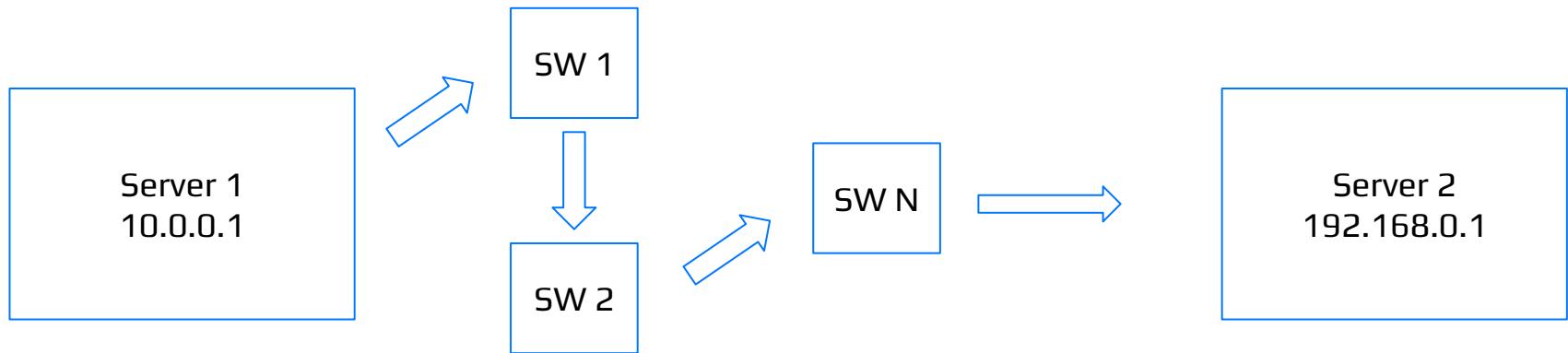


- MAC DST = MAC **шлюза / маршрутизатора**

# Почему нельзя сделать один большой L2?

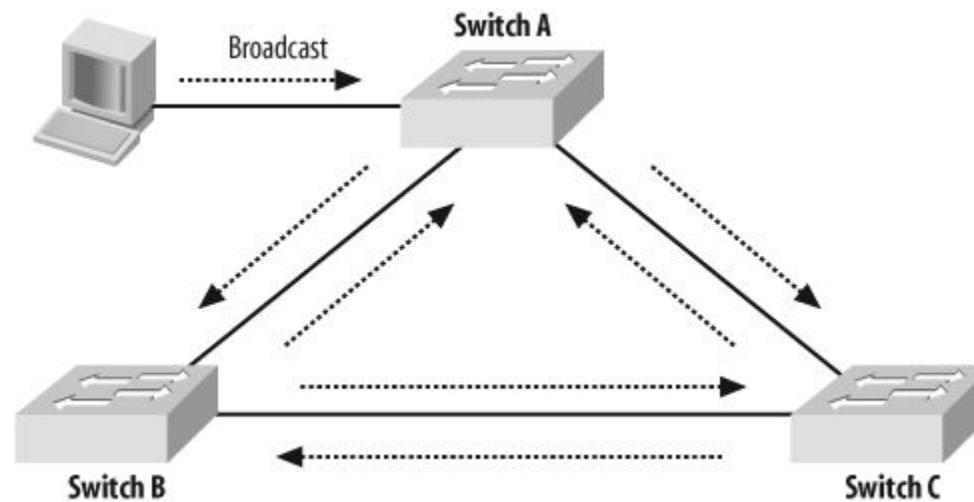


# Почему нельзя сделать один большой L2



- Низкий обсервабилити
- Высокие риски
- Лишний паразитный трафик

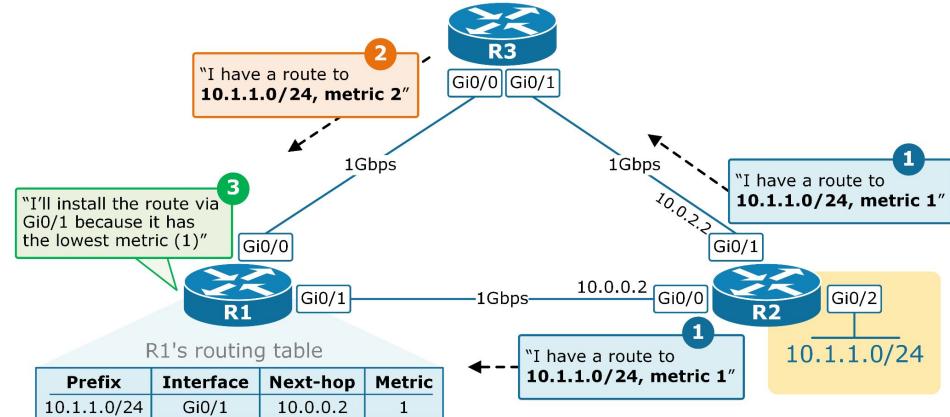
Это OK?



# Динамическая маршрутизация. OSPF

Open Shortest Path First

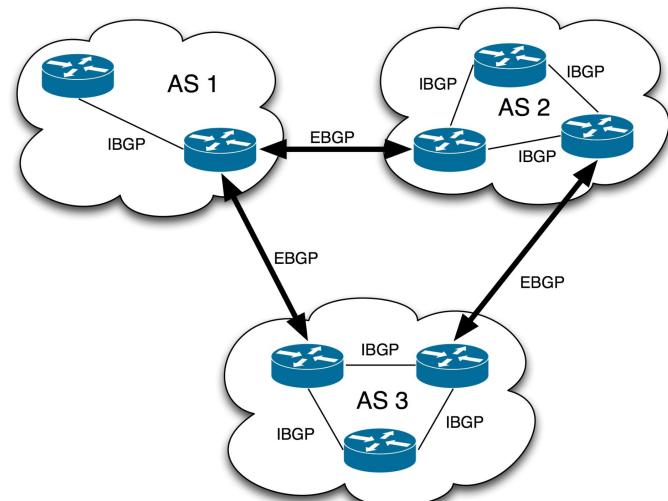
- Link-state
- Широковещательные анонсы (*нет соседства*)
- Алгоритм Дейкстры
- Картинка для привлечения внимания
- Используется в локальных сетях
- Время сходимости – секунды



# Динамическая маршрутизация. BGP

Border Gateway Protocol

- Автономные системы (зоны)
- Соседства и обмен маршрутами
- Distance-vector (path)
- Картинка для привлечения внимания
- Используется **везде**
- Время сходимости – минуты



# Динамическая маршрутизация. RIP

Routing Information Protocol



- Простой
- Медленный
- 15 хопов максимум
- Должен быть давно закопан

# Не все IP адреса одинаково полезны

## IPv4

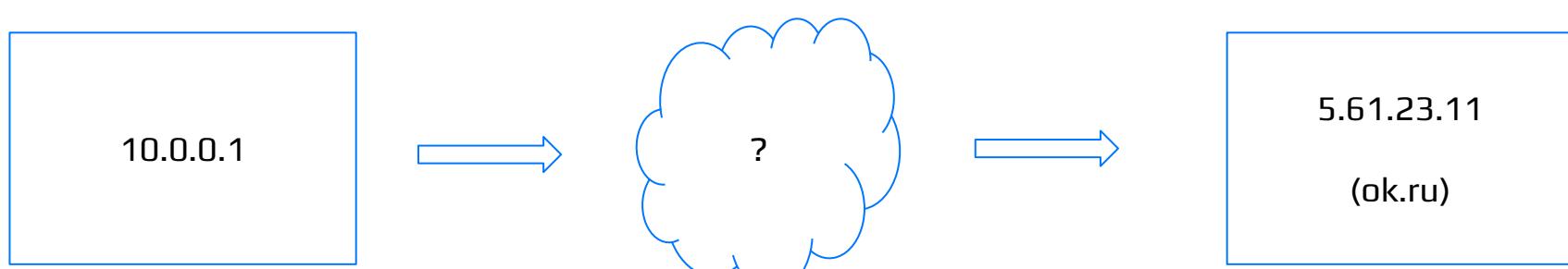
1. **10.0.0.0 - 10.255.255.255** (10.0.0.0/8)
2. **172.16.0.0 - 172.31.255.255** (172.16.0.0/12)
3. **192.168.0.0 - 192.168.255.255** (192.168.0.0/16)
4. ...

## IPv6

1. **fc00::/7**
2. **fe80::/10**
3. ...

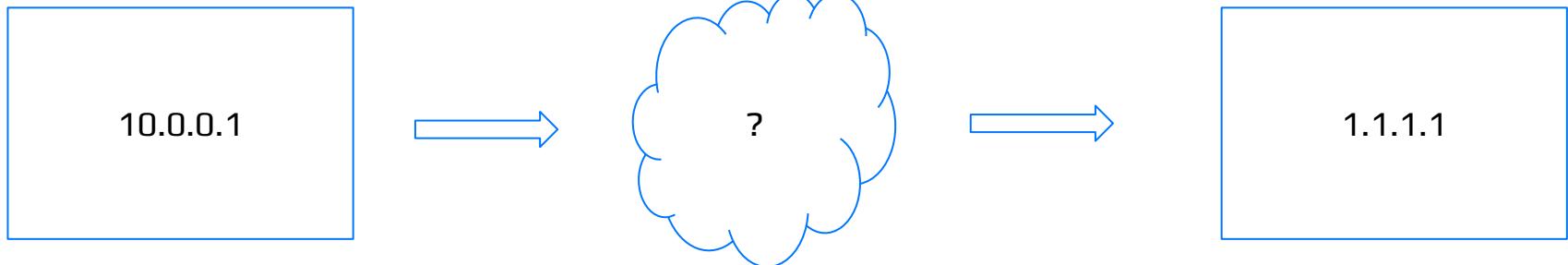
Эти диапазоны **НЕ** маршрутизируют в глобальном интернете

# Если надо вот так?



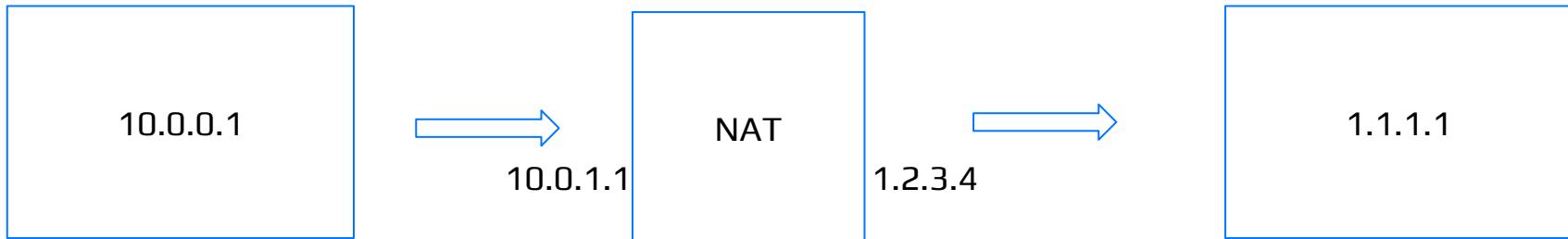
Оба сервера внутри VK

# Если надо вот так?



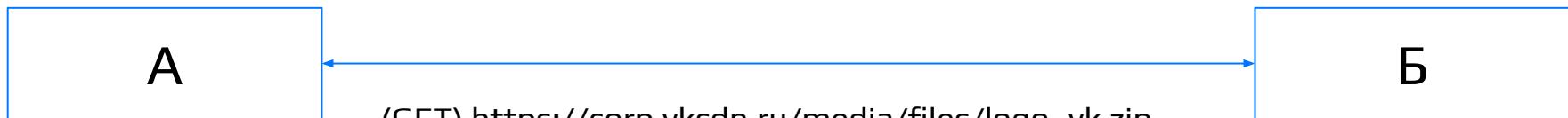
# NAT

Network Address Translation



Внутренний IP-адрес	Внутренний порт	Внешний IP-адрес	Внешний порт
10.0.0.1	12345	1.2.3.4	54321

# Знаем куда отправляем



как?

транспортный (L4): TCP/UDP; №№ портов

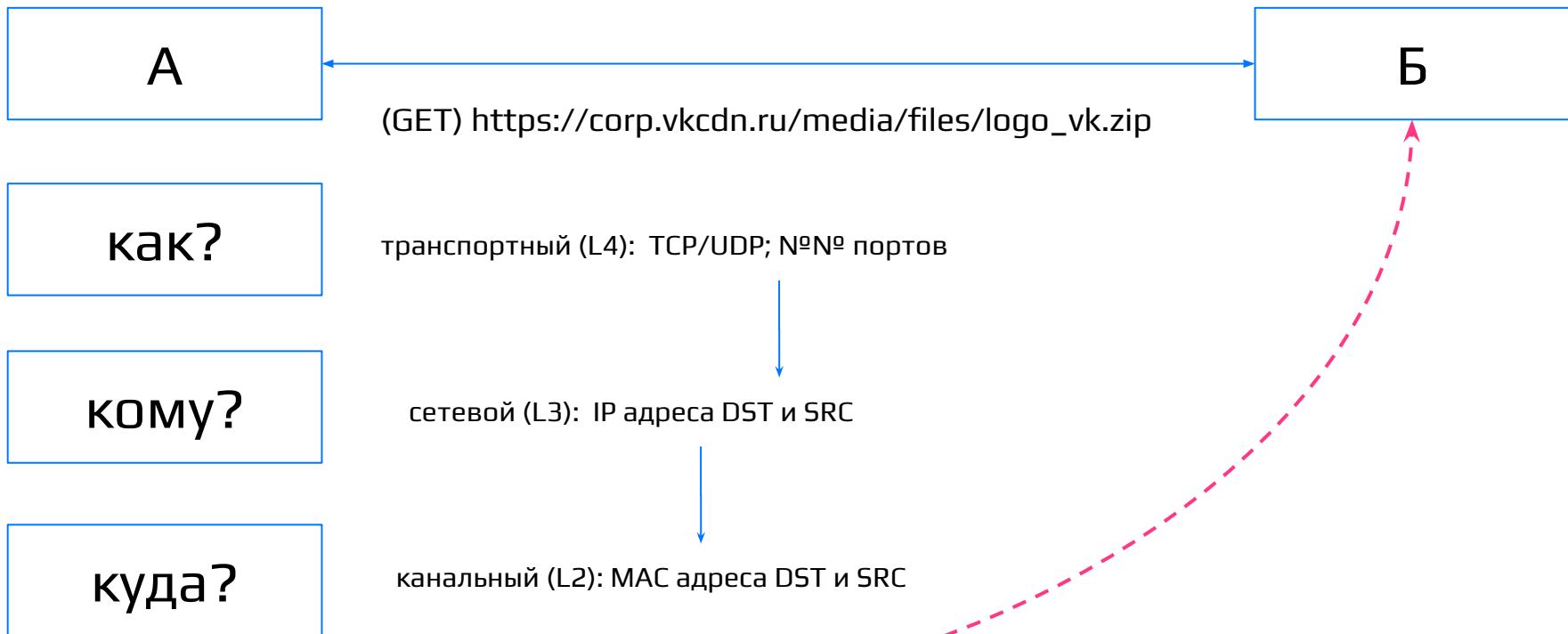
кому?

сетевой (L3): IP адреса DST и SRC

куда?

канальный (L2): MAC адреса DST и SRC

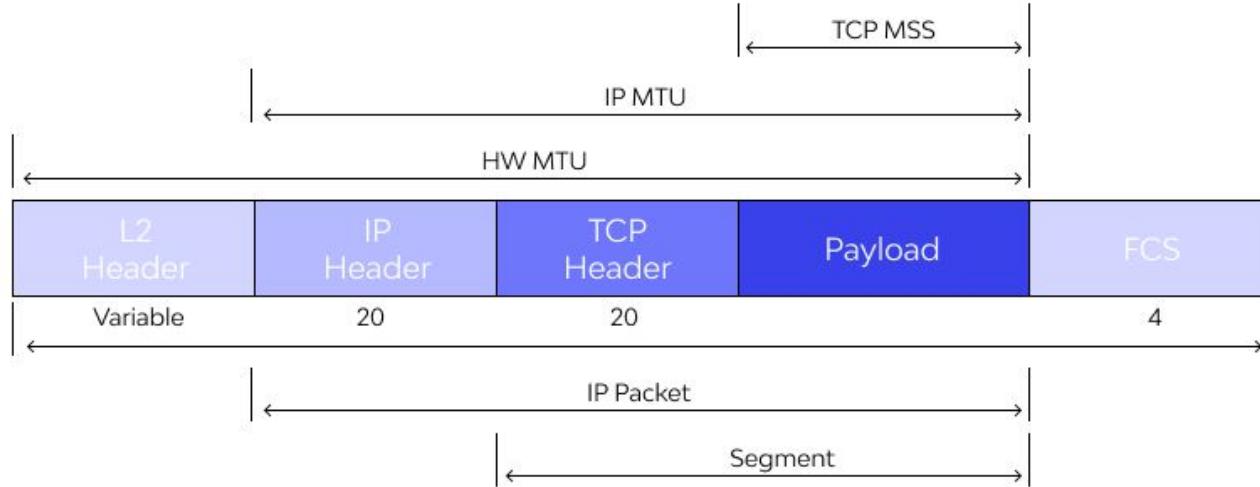
# Знаем куда отправляем



# Полный набор данных к отправке



# Полный набор данных к отправке



## История из жизни



Сколько времени будет передаваться **100 ГБ** файл по **10 Гб/с** сети?



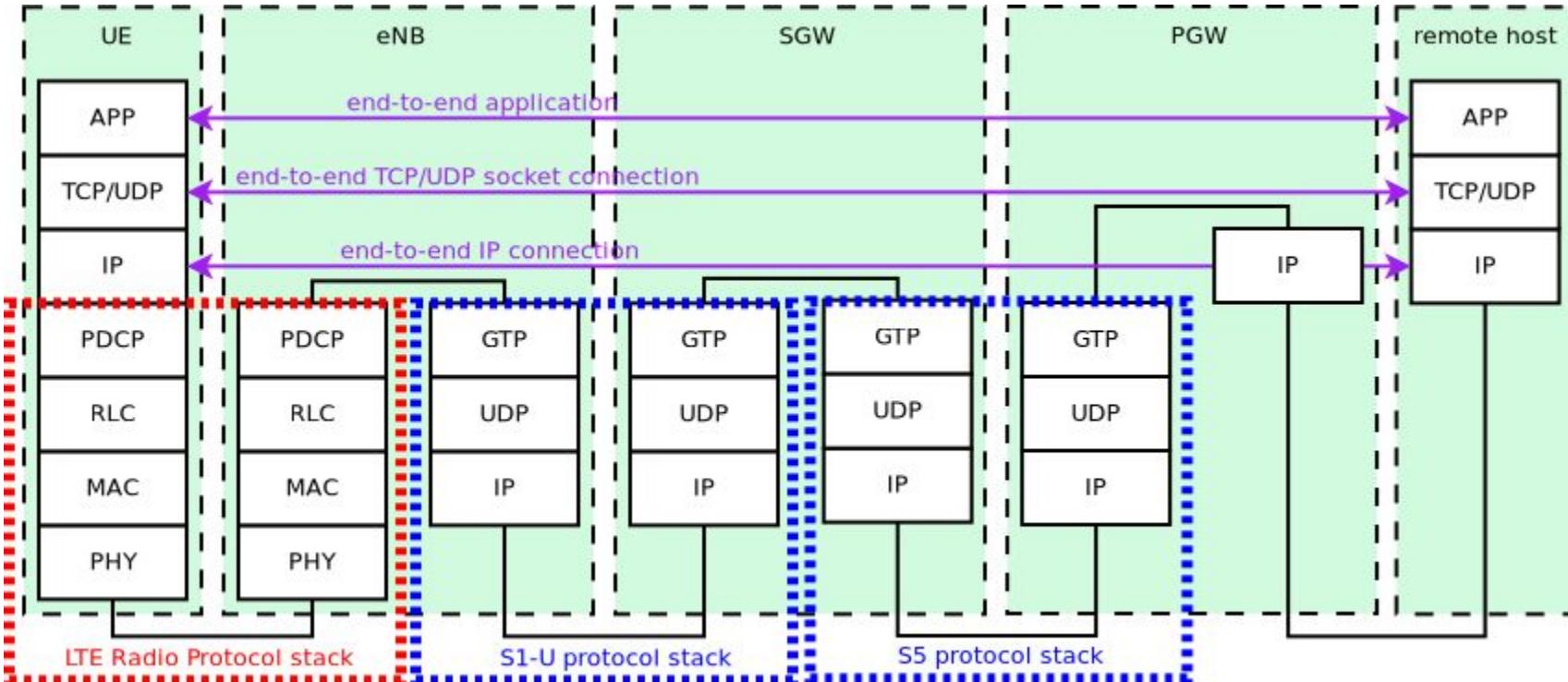
# Вопросы?



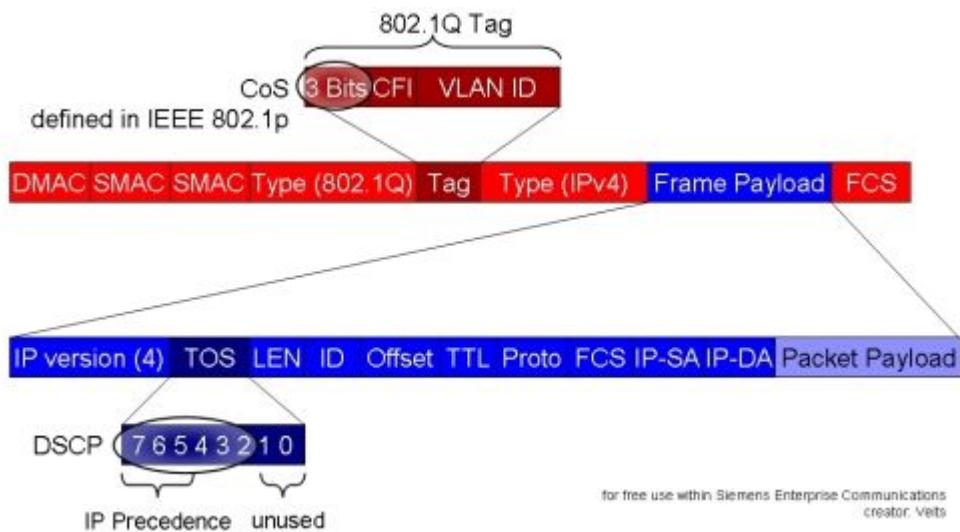
Дальше будет куда-то в сторону



# LTE data-plane



# QoS



# QoS

IPP	ToS	BIN	DSCP	DEC	CLASS CS1	CLASS CS2	CLASS CS3	CLASS CS4	IPP	DROP
0	0	000 000	CS0	0					Best Effort	
1	1	001 000	CS1	8					Scavenger	
1	1	001 010	AF11	10	AF11				Priority	Low
1	1	001 100	AF12	12	AF12				Priority	Medium
1	1	001 110	AF13	14	AF13				Priority	High
2	2	010 000	CS2	16					OAM	
2	2	010 010	AF21	18		AF21			Intermediate	Low
2	2	010 100	AF22	20		AF22			Intermediate	Medium
2	2	010 110	AF23	22		AF23			Intermediate	High
3	3	011 000	CS3	24					Signaling	
3	3	011 010	AF31	26			AF31		Flash	Low
3	3	011 100	AF32	28			AF32		Flash	Medium
3	3	011 110	AF33	30			AF33		Flash	High
4	4	100 000	CS4	32					Realtime	
4	4	100 010	AF41	34				AF41	Flash Override	Low
4	4	100 100	AF42	36				AF42	Flash Override	Medium
4	4	100 110	AF43	38				AF43	Flash Override	High
5	5	101 000	CS5	40					Video	
5	5	101 110	EF	46					Critical	
6	6	110 000	CS6	48					Routing	
7	7	111 000	CS7	56						

# QoS

QCI	Resource Type	Priority	Packet Delay Budget (NOTE 1)	Packet Error Loss Rate (NOTE 2)	Example Services
1 (NOTE 3)	GBR	2	100 ms	$10^{-2}$	Conversational Voice
2 (NOTE 3)		4	150 ms	$10^{-3}$	Conversational Video (Live Streaming)
3 (NOTE 3)		3	50 ms	$10^{-3}$	Real Time Gaming
4 (NOTE 3)		5	300 ms	$10^{-6}$	Non-Conversational Video (Buffered Streaming)
5 (NOTE 3)	Non-GBR	1	100 ms	$10^{-6}$	IMS Signalling
6 (NOTE 4)		6	300 ms	$10^{-6}$	Video (Buffered Streaming) TCP-based (e.g., www, e-mail, chat, ftp, p2p file sharing, progressive video, etc.)
7 (NOTE 3)		7	100 ms	$10^{-3}$	Voice, Video (Live Streaming) Interactive Gaming
8 (NOTE 5)		8	300 ms	$10^{-6}$	Video (Buffered Streaming) TCP-based (e.g., www, e-mail, chat, ftp, p2p file sharing, progressive video, etc.)
9 (NOTE 6)		9			

Дальше будет немного картинок



## Какие-то железяки



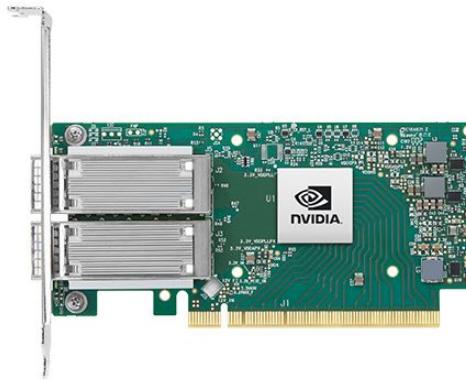
- 10/25/40/100GE
- Оптика/Медь
- SFP (CFP..)

# Какие-то железяки

## Single/Dual-Port 100Gb/s Ethernet Adapter

ConnectX-5 Ethernet adapter cards provide a high performance and flexible solution that deliver a range of innovative offloads and accelerators in hardware for increased efficiency for data center network and storage connectivity. ConnectX-5 adapter cards bring advanced Open vSwitch offloads to telecommunications and cloud service providers and enterprise data centers to drive extremely high packet rates and throughput, thus boosting data center infrastructure efficiency.

[Learn More >](#)



## 10/25/40/50 Gigabit Ethernet Adapter Cards

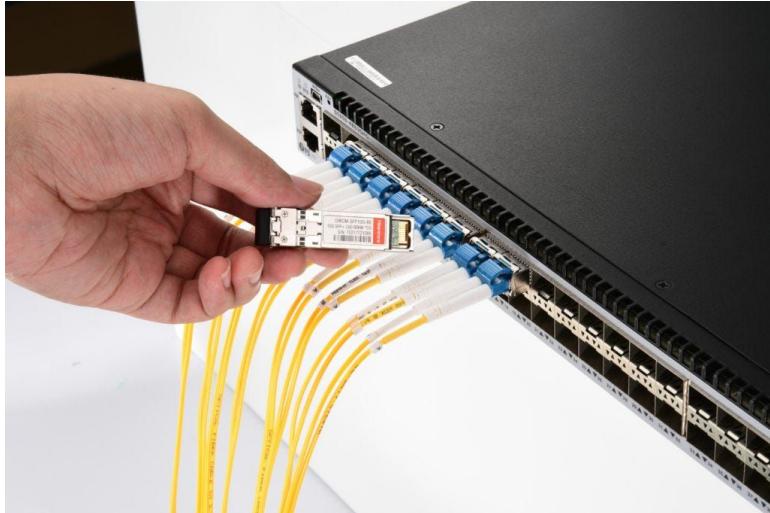
ConnectX-4 Lx adapter cards enable data centers to leverage leading interconnect adapters to increase their operational efficiency to improve server utilization and maximize application productivity, while reducing total cost of ownership (TCO).

ConnectX-4 Lx provides support for 1, 10, 25, 40, and 50GbE bandwidth, sub-microsecond latency and a 70 million packets per second message rate. It includes native hardware support for RDMA over converged Ethernet, Ethernet stateless offload engines, overlay networks, and NVIDIA GPUDirect® Technology.

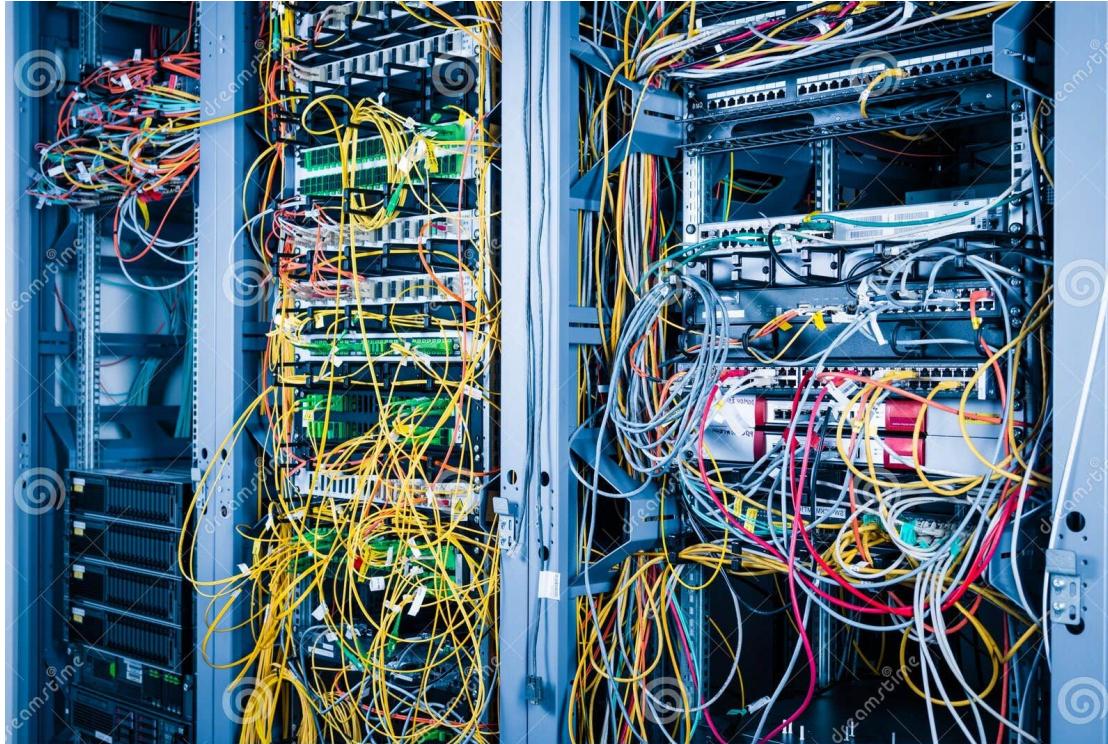
[LEARN MORE >](#)



# Ещё железяки

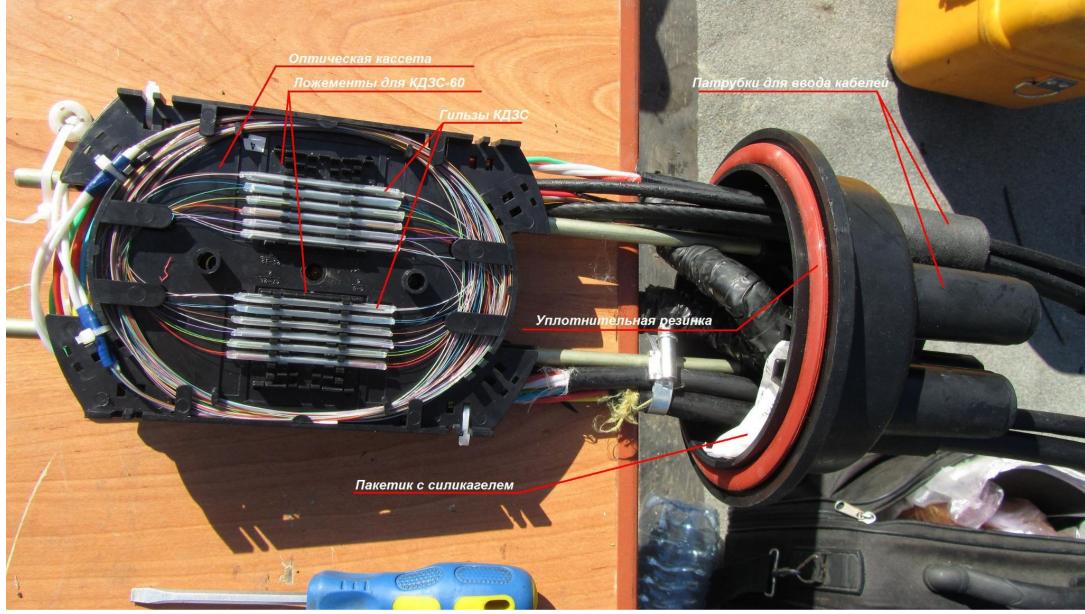


# Творческий подход к кабель менеджменту



# Скукотища





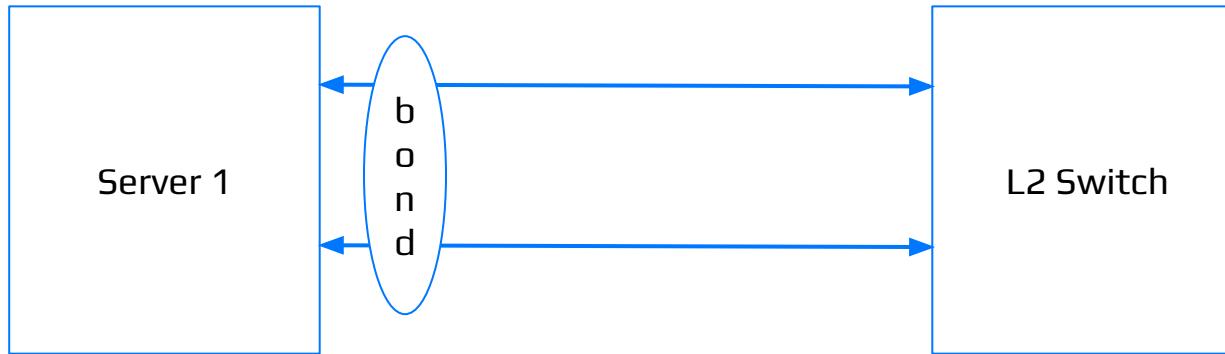
Картинки закончились





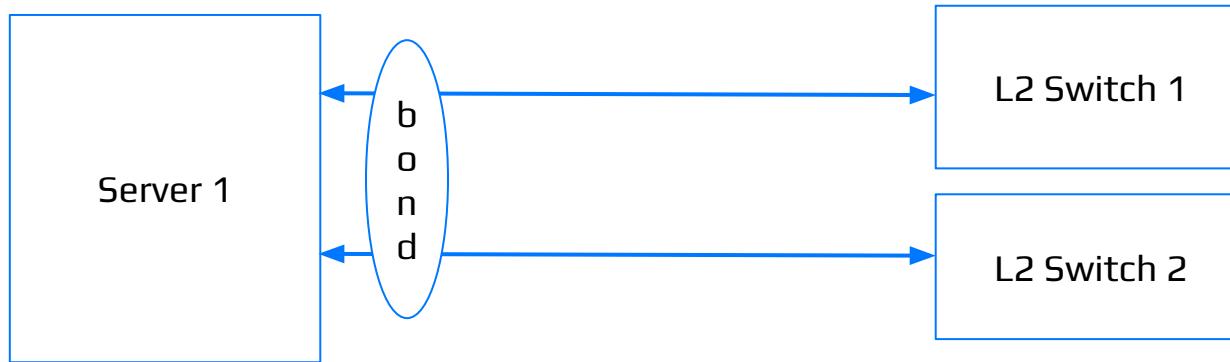
- Что делать, если у нас мощный сервер и один линк не вывозит?

# Агрегация интерфейсов



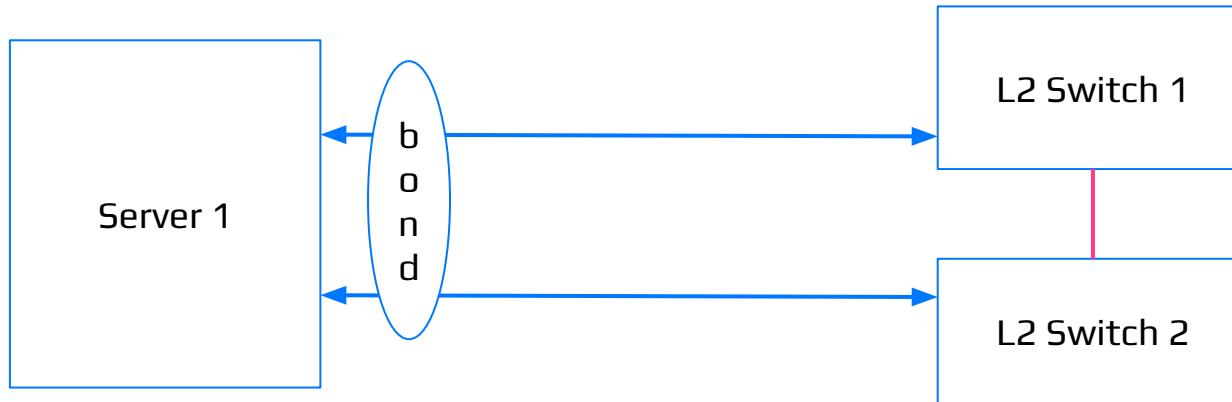
- Что делать, если у нас мощный сервер и один линк не вывозит?
  - подключить **ещё** линки

# Агрегация интерфейсов



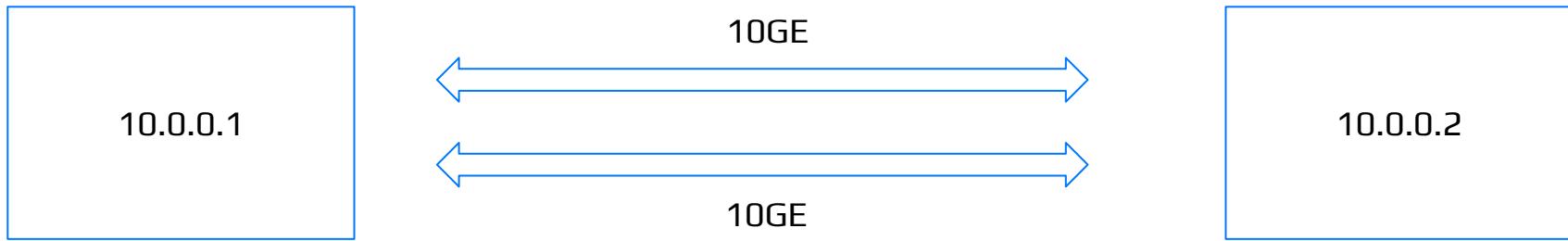
- Зачем нам это может понадобиться?
- Можно ли так сделать?

# Агрегация интерфейсов



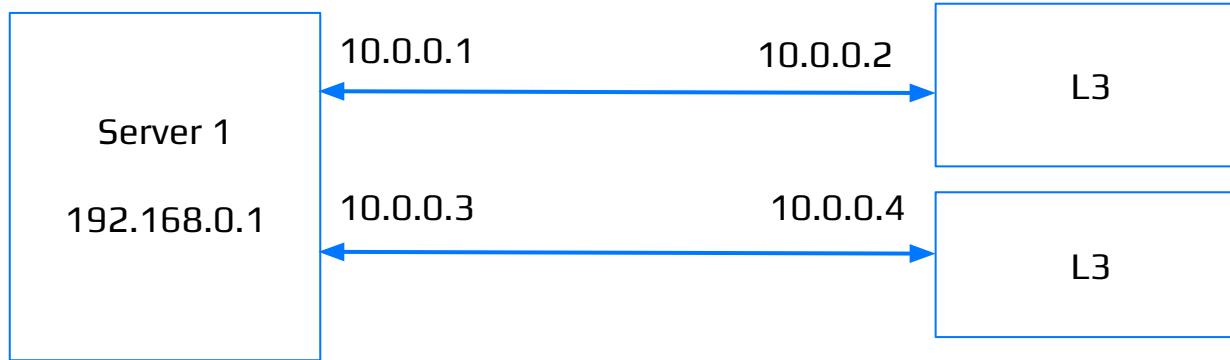
- Зачем нам это может понадобиться? – **отказоустойчивость**
- Можно ли так сделать? –**да, но нужна связь между свитчами**

## История из жизни 2

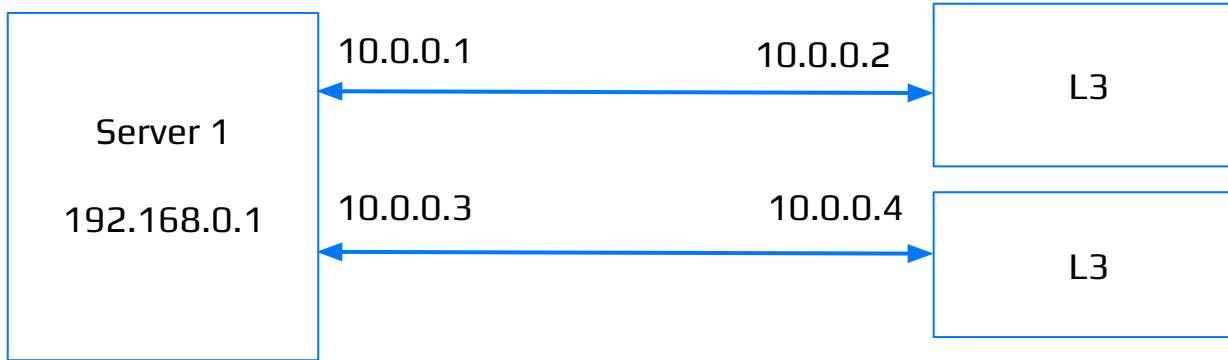


Сколько времени будет передаваться **100 ГБ** файл через **bond 10 Гб/с + 10 Гб/с** сети?

## А если линки L3?



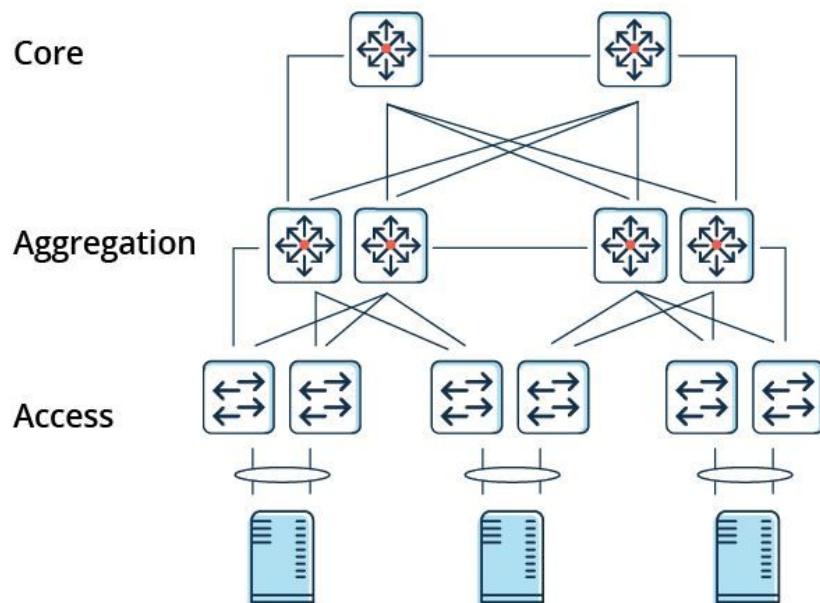
# ECMP



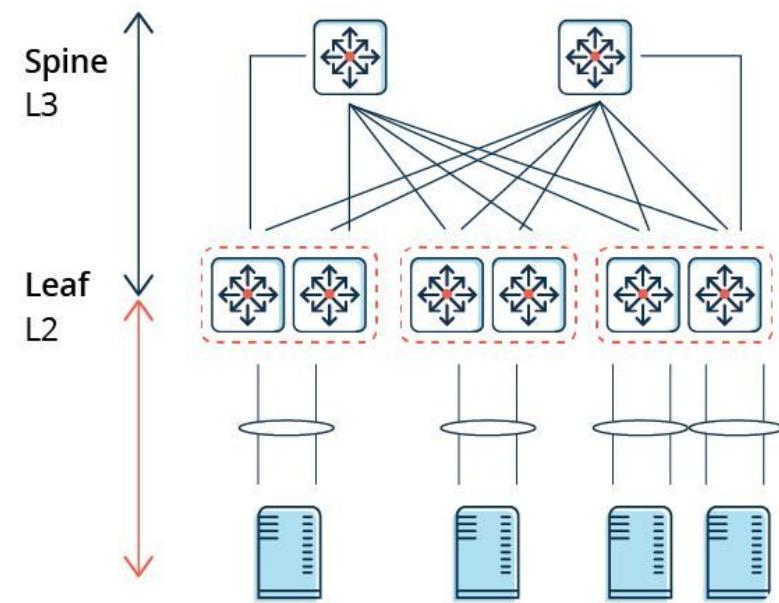
```
[root@host ~]# ip r
default proto static src 192.168.0.1
    nexthop via 10.0.0.2 dev eth1 weight 1
    nexthop via 10.0.0.4 dev eth2 weight 1
```

# Архитектура сети

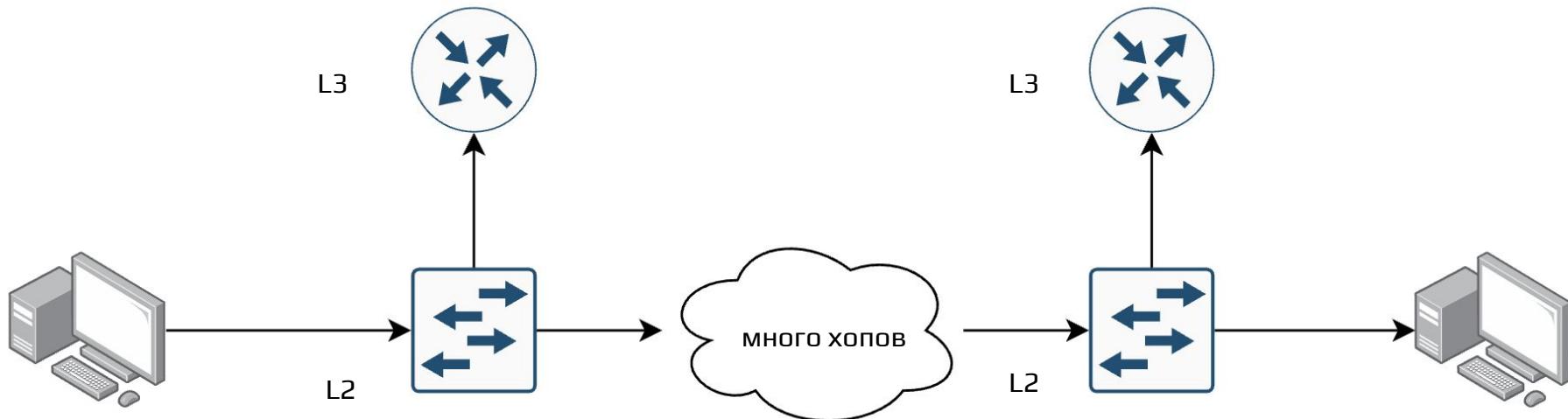
## Traditional 3-Tier Architecture



## 2-Tier Spine-Leaf Architecture



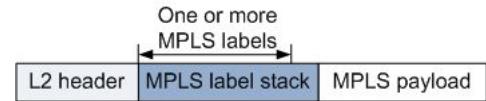
# Иногда нужно быстрее



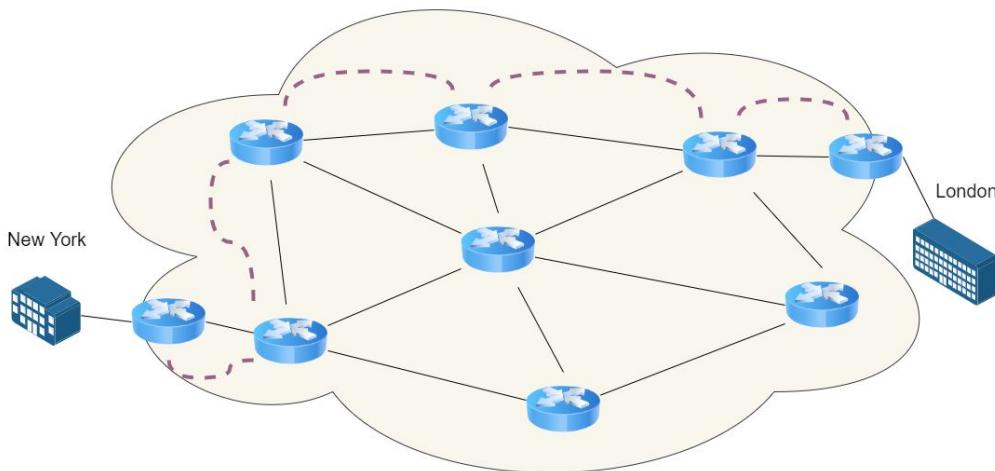
Маршрутизация на базе хостов, т.е. знания nexthop

# MPLS

Добавляем метку и смотрим только на неё, без анализа всего заголовка



Label Switched Path - - - -



# Итоги раздела:

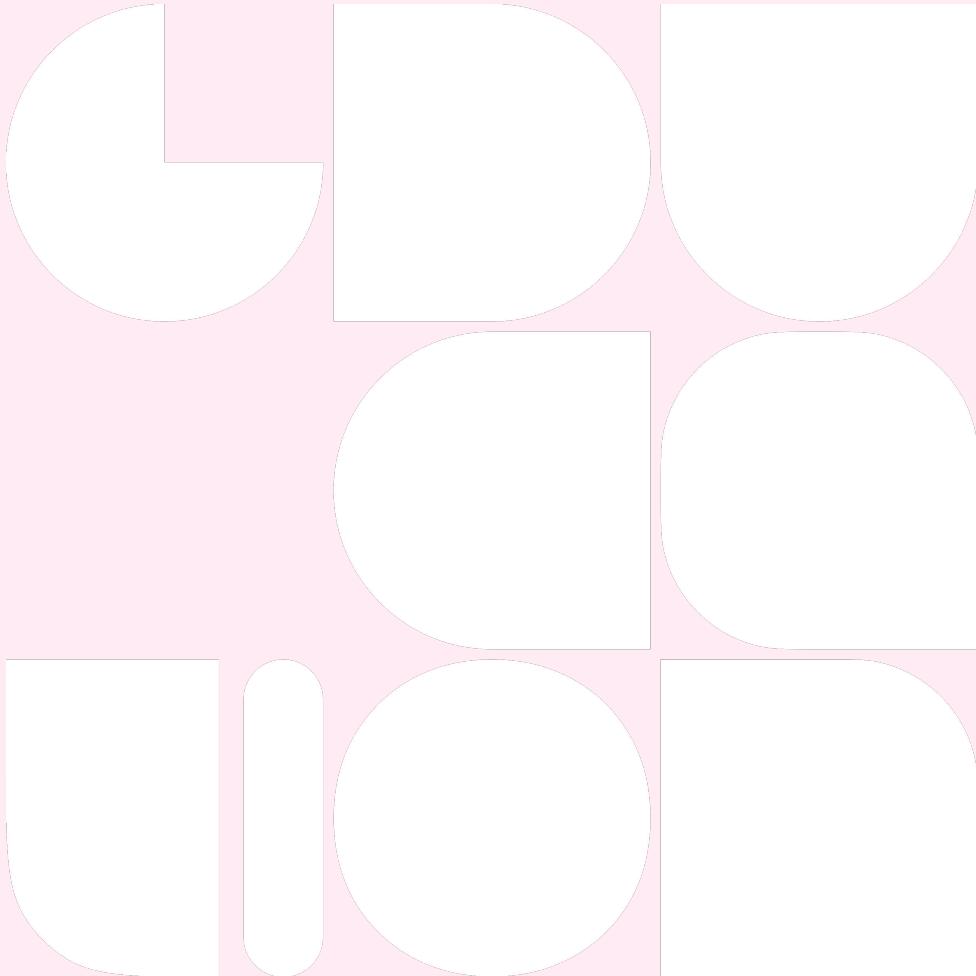
- Для передачи данных необходимо добавить к ним служебную информацию
  - L4, Транспортный уровень: TCP/UDP порты
  - L3, Сетевой уровень: IP адреса
  - L2, Канальный уровень: MAC адреса
- Процесс называют инкапсуляция
- Устройство уровня L2 – свитч; L3 – роутер
- Сегментация L2 по vlan
- Маршрутизация по BGP
- Можно объединять линки для увеличения ёмкости/надёжности



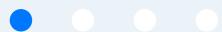
# Вопросы?



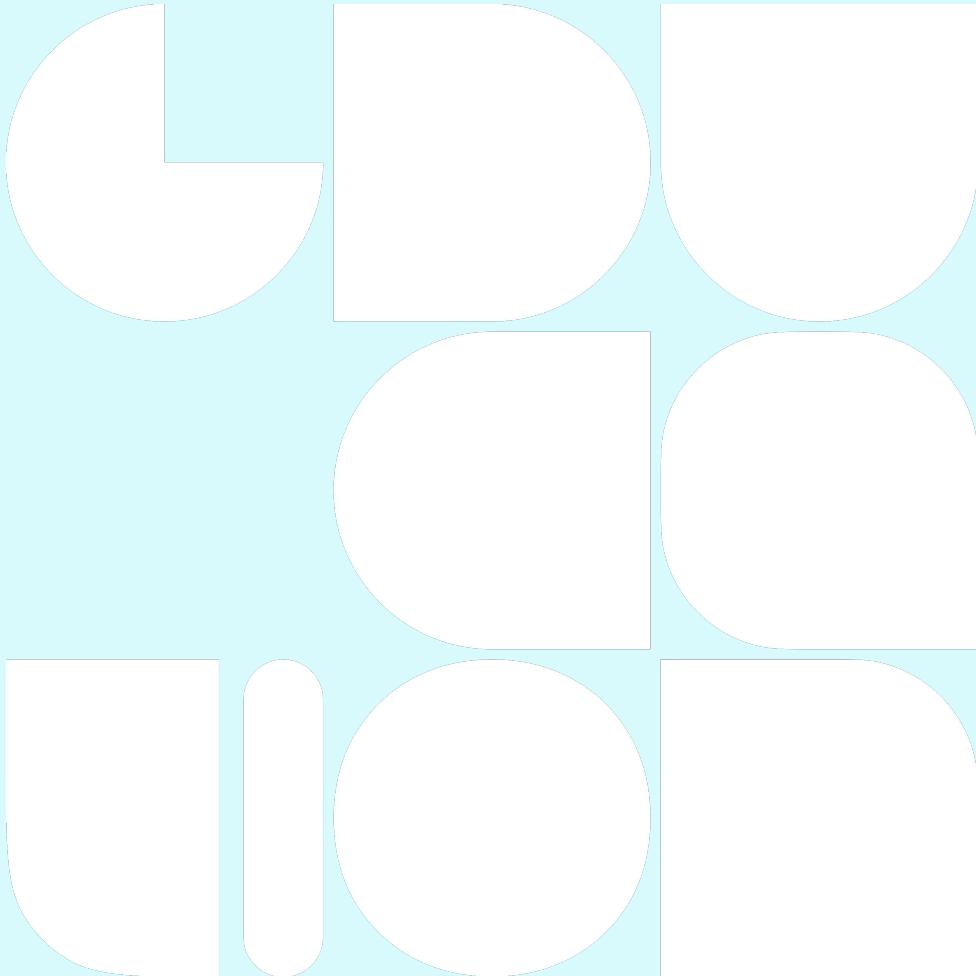
# Перерыв



# Сетевой стек Linux



- Общее описание
- Входящий пакет
- Исходящий пакет



# Прежде чем начнём

1. Ядро (Kernel):
  - Управляет всеми аспектами сетевого стека.
  - Обработка пакетов, маршрутизация, фильтрация.
2. Драйвер (Driver):
  - Обеспечивает взаимодействие между сетевыми картами и ядром.
  - Управление аппаратными ресурсами сетевых карт.
3. Интерфейсы (Interfaces):
  - Точки подключения к сети (физические и виртуальные)
4. Сокеты (Sockets):
  - Программные интерфейсы для обмена данными через сеть

# Сетевой стек Linux

<https://dev.to/amrelhusseiny/linux-networking-part-1-kernel-net-stack-180l>

<https://youtu.be/6Fl1rsxk4JQ>

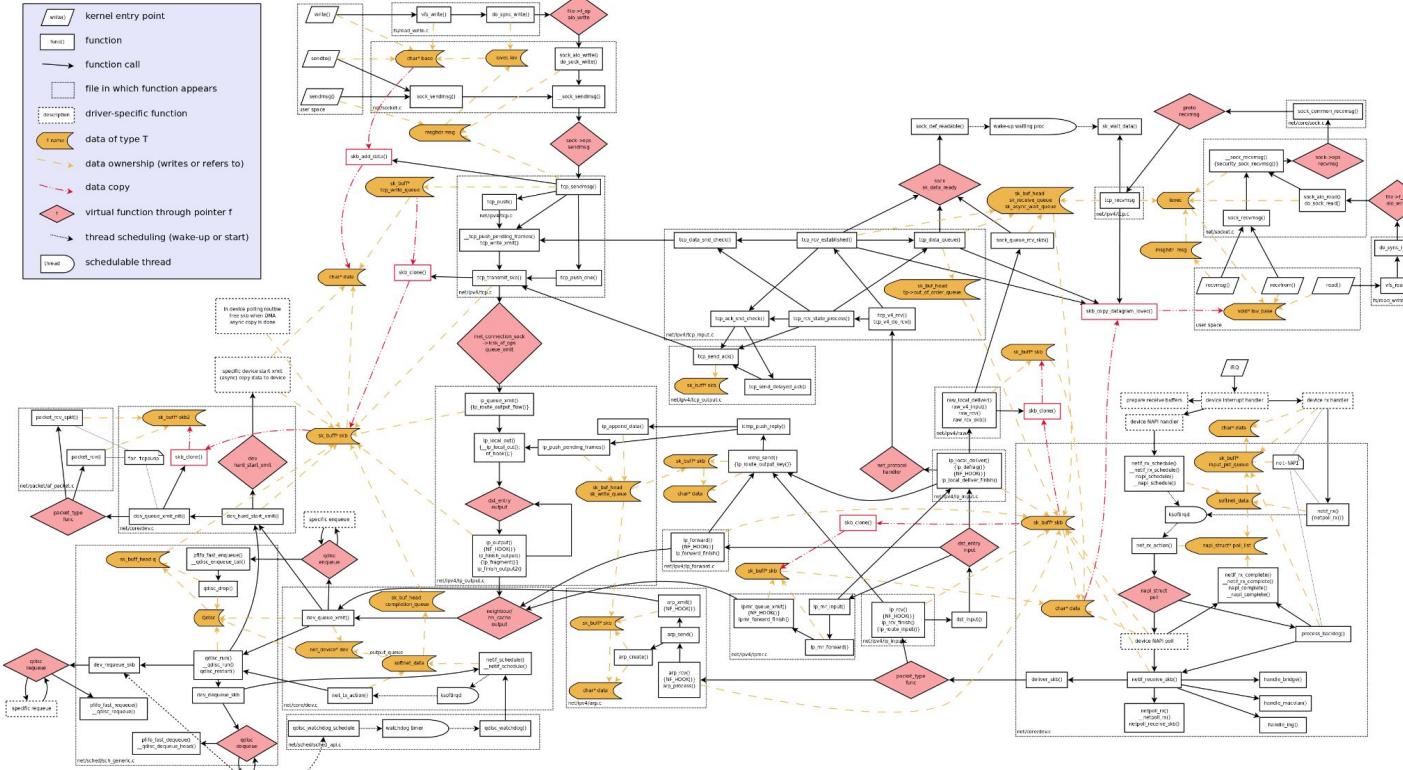
<https://blog.packagecloud.io/monitoring-tuning-linux-networking-stack-sending-data>

<https://blog.packagecloud.io/monitoring-tuning-linux-networking-stack-receiving-data>

<https://habr.com/ru/companies/vk/articles/314168>

<https://ntk148v.github.io/posts/linux-network-performance-ultimate-guide>

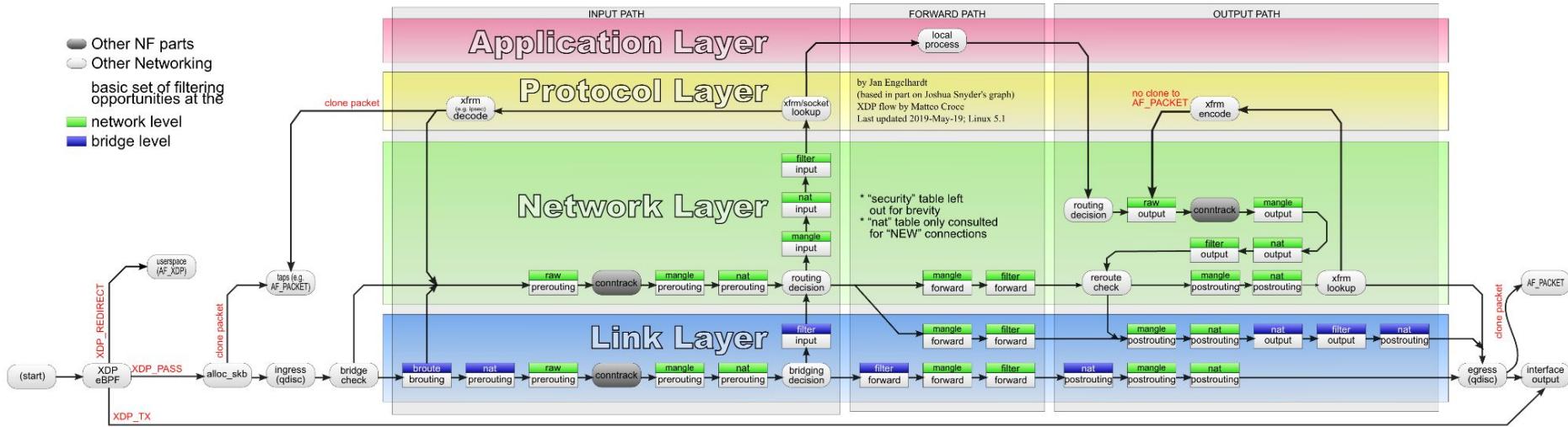
# Сетевой стек Linux



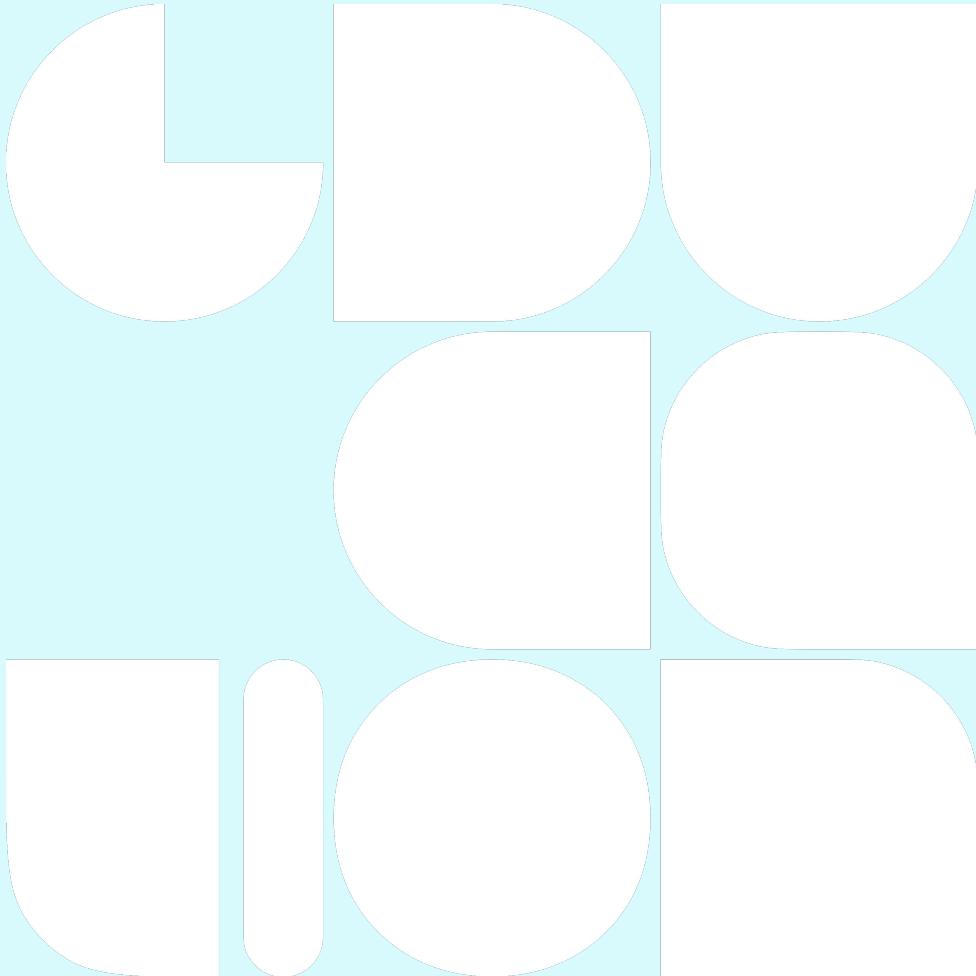
<https://the-linux-channel.the-toffee-project.org/i/LINKS/1/Network%20API%20and%20data%20flow%20within%20Linux%20kernel.png>

# Сетевой стек Linux

- Other NF parts
- Other Networking
- basic set of filtering opportunities at the  
network level
- bridge level



- Общее описание
- Входящий пакет
- Исходящий пакет



NIC

CPU

NIC

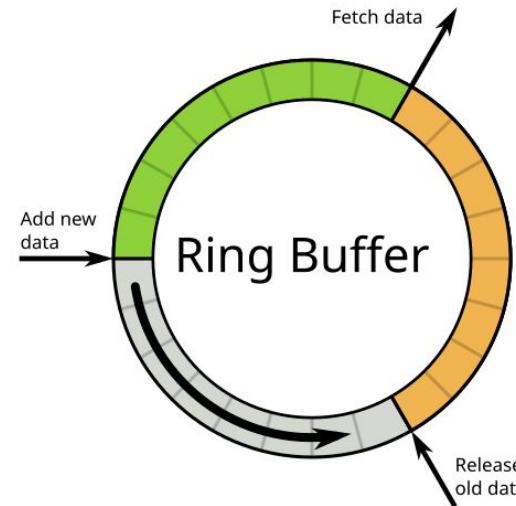
- проверяет DST MAC и чексы
- сохраняет пакет в память (**DMA**)

## NIC

- проверяет DST MAC и чексы
- сохраняет пакет в память (**DMA**)

```
[root@host ~]# ethtool -i eth1
...
driver: mlx5_core

[root@host ~]# ethtool -g eth1
...
Current hardware settings:
RX:      8192
RX Mini: n/a
RX Jumbo: n/a
TX:      8192
```

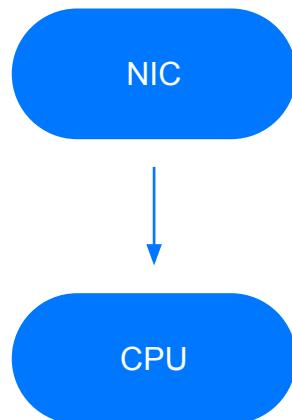


NIC

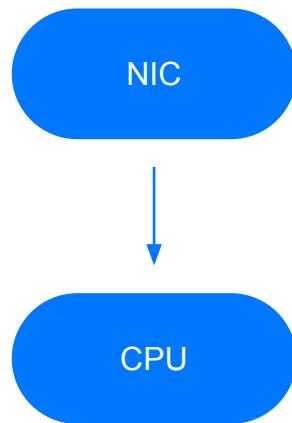
CPU

NIC

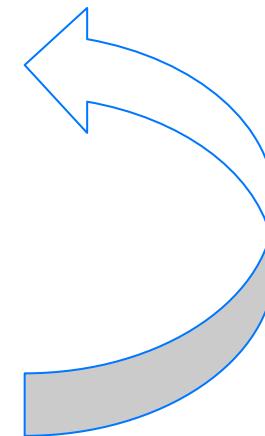
- проверяет DST MAC и чекsumмы
- сохраняет пакет в память (**DMA**)
- вызывает прерывание процессора



- проверяет DST MAC и чекsumмы
  - сохраняет пакет в память (**DMA**)
  - вызывает прерывание процессора
- 
- подтверждает получение пакета
  - добавляет пакет в очередь softirq



- проверяет DST MAC и чекsumмы
  - сохраняет пакет в память (**DMA**)
  - вызывает прерывание процессора
- 
- подтверждает получение пакета
  - добавляет пакет в очередь softirq



```
[andrey.trofimov@host ~]$ ethtool -l eth1
Channel parameters for eth1:
Pre-set maximums:
RX:          0
TX:          0
Other:        1
Combined:    8
Current hardware settings:
RX:          0
TX:          0
Other:        1
Combined: 8
```

- HW прерывания **ОЧЕНЬ** дорогие
- Вместо этого делаем polling через SW прерывания
- *NAPI (**New API**) was introduced to the Linux kernel in **2001***

```
[root@host ~]# pgrep ksoftirqd | wc -l  
128  
[root@host ~]# nproc  
128
```

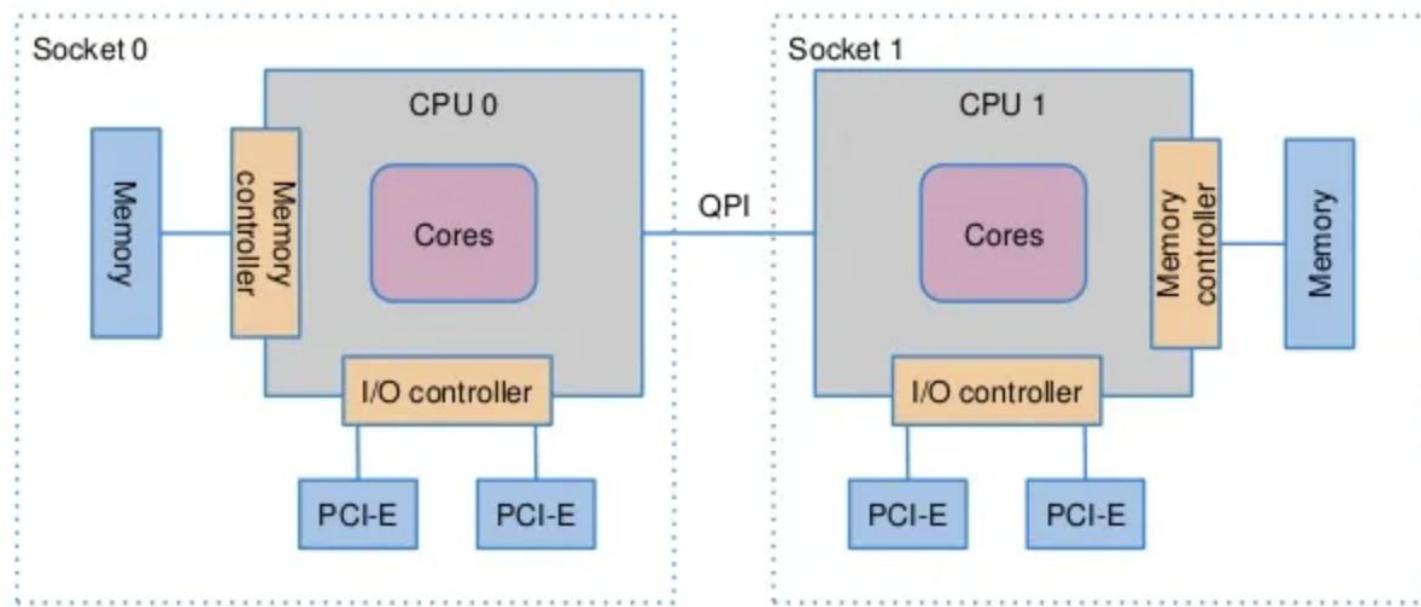
```
# watch -n1 grep RX /proc/softirqs
```

```
[root@srvu545 ~]# cat /proc/interrupts | grep -e CPU -e eth
CPU0   CPU1   CPU2   CPU3   CPU4   CPU5   CPU6   CPU7   CPU8   CPU9   CPU10  CPU11  CPU12  CPU13
33:    0       0       2       0       0       0       0       0       0       0       0       0       0       0
34:    0       0       0       4       0       0       0       2203960670 1364175389 0       0       0       0       0
35:    0       0       0       0       5       0       0       0       1938512565 1357946612 0       0       0       0
36:    0       0       0       0       0       3       0       0       0       1864711769 1347774507 0       0       0
37:    0       0       0       1842875370 0       0       3       0       0       0       0       0       1323896565 0
38:    0       0       0       0       2311043261 0       0       3       0       0       0       0       0       1353055754 0
39:    0       0       0       0       0       1819441803 0       0       3       0       0       0       0       0       1356158529
40:    0       0       0       0       0       0       1762792591 0       0       3       0       0       0       0       0       0       1
41:    0       0       0       0       0       0       0       0       0       0       0       489679757 0       0       0       0
44:    0       0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
45: 1328522169 0       0       0       0       0       0       0       0       0       0       0       0       0       0       0
46: 0       1362295174 0       0       0       0       0       0       0       0       0       0       0       0       0       0
47: 0       0       1353223626 0       0       0       0       0       0       0       0       0       0       0       0       0
48: 0       0       0       1346246833 0       0       0       0       0       0       0       0       0       0       0       0
49: 0       0       0       0       1347975349 0       0       0       0       0       0       0       0       0       0       0
50: 0       0       0       0       0       1385455169 0       0       0       0       0       0       0       0       0       0
51: 0       0       0       0       0       0       1328096265 0       0       0       0       0       0       0       0       0
52: 0       0       0       1347336577 0       0       0       0       0       0       1325995332 0       0       0       0       0
```

CPU13	CPU14	CPU15	CPU16	CPU17	CPU18	CPU19	CPU20	CPU21	CPU22	CPU23			
0	0	0	0	0	0	0	0	0	0	0	IR-PCI-MSI	2621440-edge	eth1
0	0	0	0	0	0	0	0	3674235	0	0	IR-PCI-MSI	2621441-edge	eth1-TxRx-0
0	0	0	0	0	0	0	0	0	4127442	0	IR-PCI-MSI	2621442-edge	eth1-TxRx-1
0	0	0	0	0	0	0	0	0	0	3349755	IR-PCI-MSI	2621443-edge	eth1-TxRx-2
0	0	0	0	4091045	0	0	0	0	0	0	IR-PCI-MSI	2621444-edge	eth1-TxRx-3
0	0	0	0	0	3932112	0	0	0	0	0	IR-PCI-MSI	2621445-edge	eth1-TxRx-4
8529	0	0	0	0	0	3527512	0	0	0	0	IR-PCI-MSI	2621446-edge	eth1-TxRx-5
0	1356772431	0	0	0	0	0	3353595	0	0	0	IR-PCI-MSI	2621447-edge	eth1-TxRx-6
0	0	584960333	0	0	0	0	0	0	0	0	IR-PCI-MSI	2621448-edge	eth1-TxRx-7
0	0	0	2	0	0	0	0	0	0	0	IR-PCI-MSI	2623488-edge	eth0
0	0	0	0	5	0	0	0	2122604495	0	0	IR-PCI-MSI	2623489-edge	eth0-TxRx-0
0	0	0	0	0	7	0	0	0	2261094175	0	IR-PCI-MSI	2623490-edge	eth0-TxRx-1
0	0	0	0	0	0	4	0	0	0	1937764555	IR-PCI-MSI	2623491-edge	eth0-TxRx-2
0	0	0	0	1931515135	0	0	4	0	0	0	IR-PCI-MSI	2623492-edge	eth0-TxRx-3
0	0	0	0	0	2222731942	0	0	7	0	0	IR-PCI-MSI	2623493-edge	eth0-TxRx-4
0	0	0	0	0	0	1792250668	0	0	4	0	IR-PCI-MSI	2623494-edge	eth0-TxRx-5
0	0	0	0	0	0	1859323912	0	0	4	4	IR-PCI-MSI	2623495-edge	eth0-TxRx-6
0	0	0	0	0	0	0	0	0	0	0	IR-PCI-MSI	2623496-edge	eth0-TxRx-7

-> go to real world

- NUMA (Non-Uniform Memory Access)

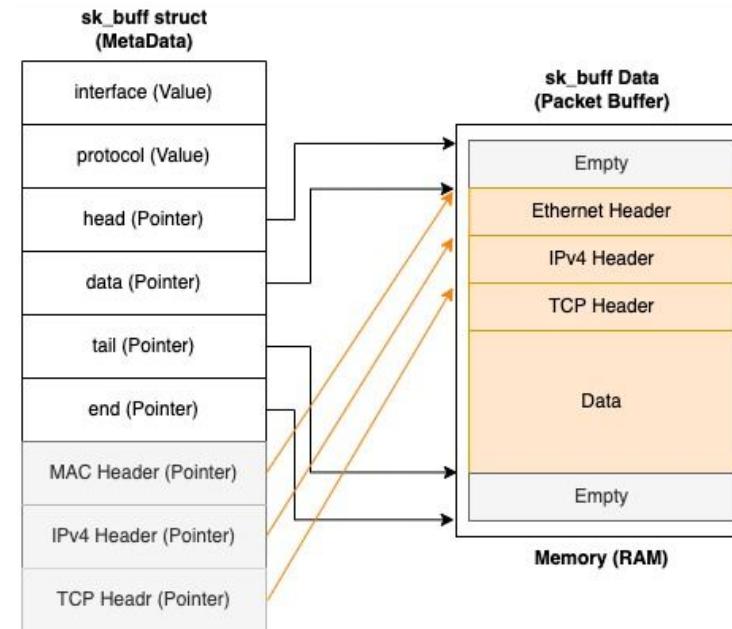


```
[root@host ~]# grep '' /sys/class/net/*	device/numa_node  
/sys/class/net/eth0/device/numa_node:1  
/sys/class/net/eth1/device/numa_node:0  
/sys/class/net/eth2/device/numa_node:1  
/sys/class/net/eth3/device/numa_node:0
```

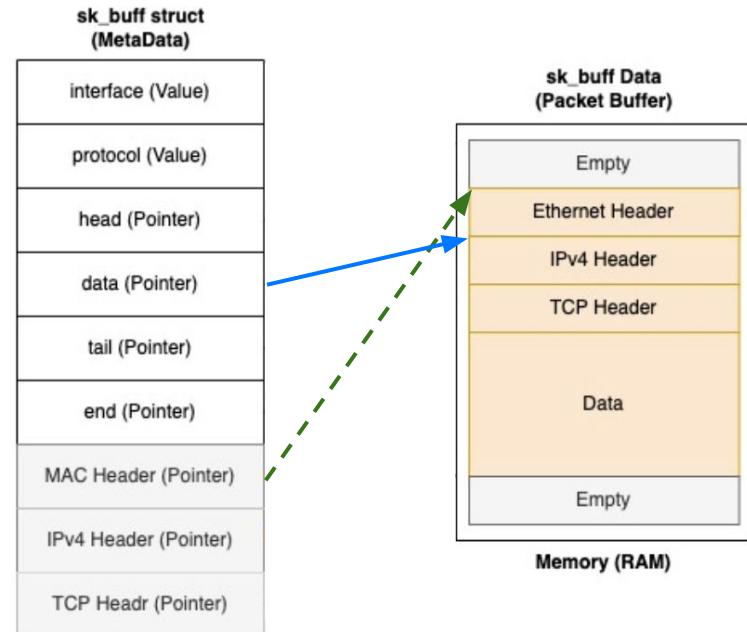
```
[root@host ~]# numactl --hardware

available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94
95
node 0 size: 257038 MB
node 0 free: 9242 MB
node 1 cpus: 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
59 60 61 62 63 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116
117 118 119 120 121 122 123 124 125 126 127
node 1 size: 258028 MB
node 1 free: 7240 MB
node distances:
node    0    1
 0:  10  20
 1:  20  10
```

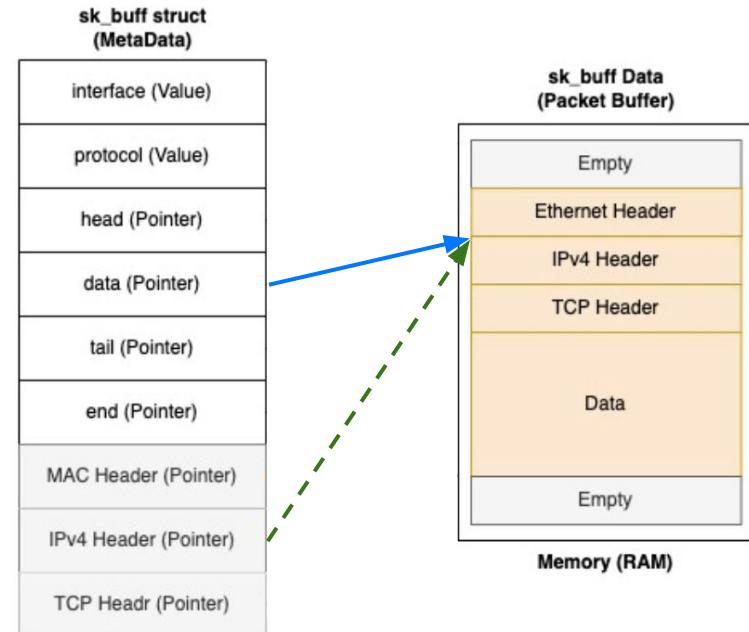
- создаёт **sk\_buff (skb)**



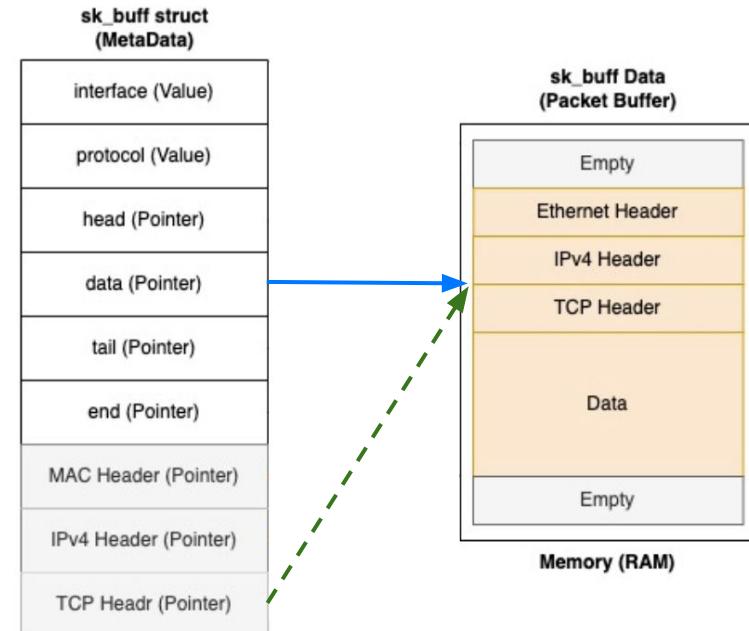
- создаёт **sk\_buff (skb)**
- заполняет protocol/interface
- заполняет указатель **mac\_header** (MAC Header)
- заполняет указатель **data** -> IPv4 Header  
(удаляет Ethernet Header)
- передаёт **skb** в сетевой стек



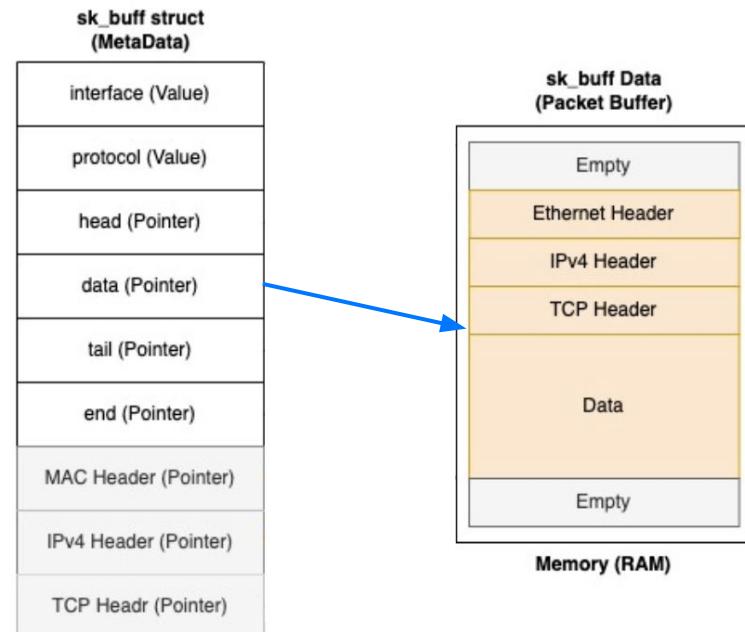
- заполняет указатель **network\_header** (IPv4 Header)
- копирует skb в TAPs (tcpdump)
- вызывает **tc ingress qdisc**
- если есть wlan или master -> передаёт обработку
- вызывает функцию L3 протокола (например IP)



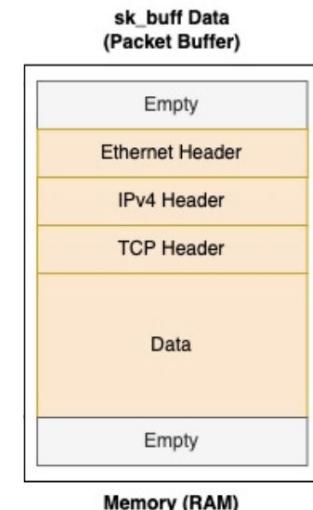
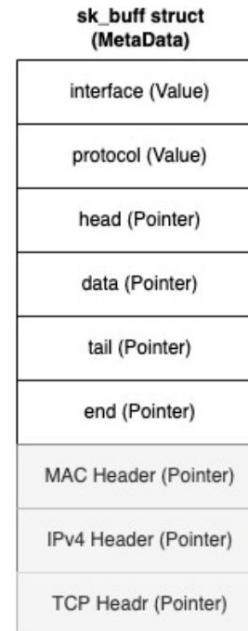
- меняет указатель **data** -> TCP Header (удаляет IPv4 Header)
- заполняет указатель **transport\_header** (TCP Header)
- вызывает **netfilter** (PRE\_ROUTING)
- проверяет **маршрутизацию** -> forward / delivery
- вызывает **netfilter** (LOCAL\_IN)
- вызывает функцию L4 протокола (например TCP)

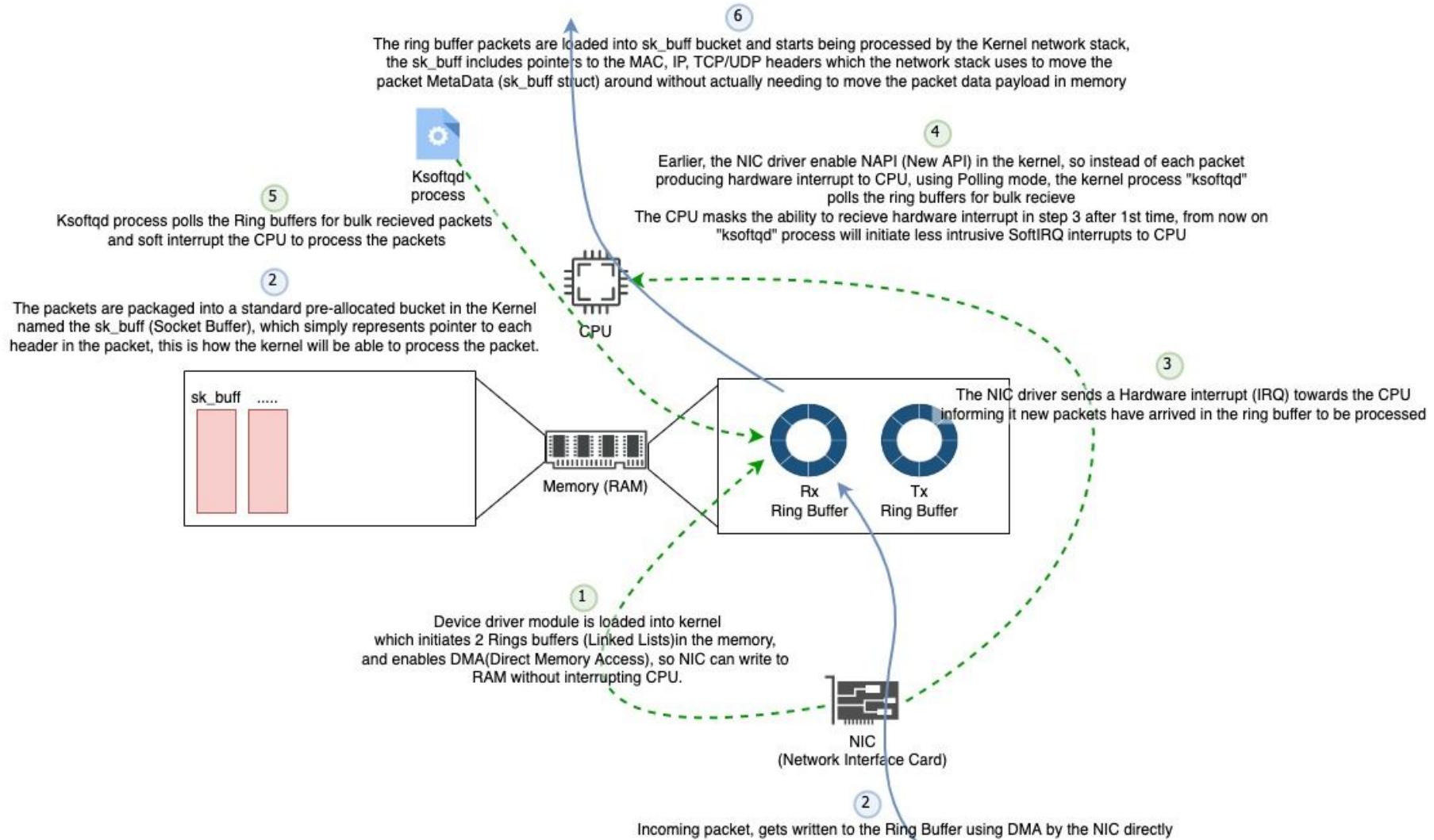


- меняет указатель **data** -> Data (удаляет TCP Header)
- ищет приложение (**сокет**) которому адресован пакет
- триггерит **TCP State**
- помещает пакет во входящую **очередь сокета**
- **информирует приложение о новых данных для него**

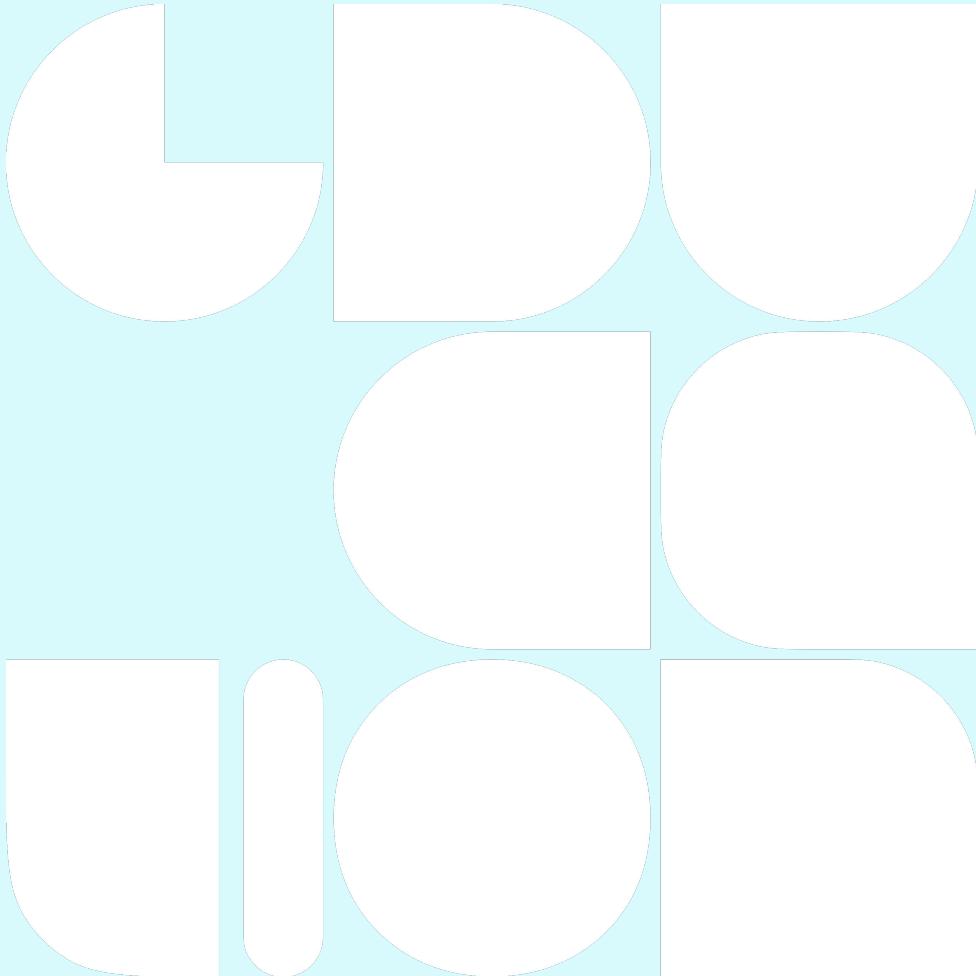


- читает пакет из входящей очереди
- копирует данные в буфер приложения
- **удаляет skb**



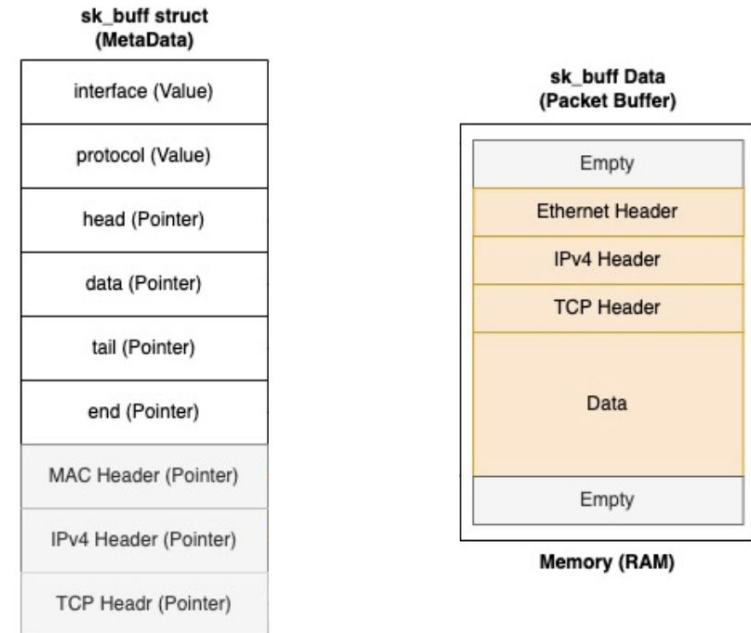


- Общее описание
- Входящий пакет
- Исходящий пакет

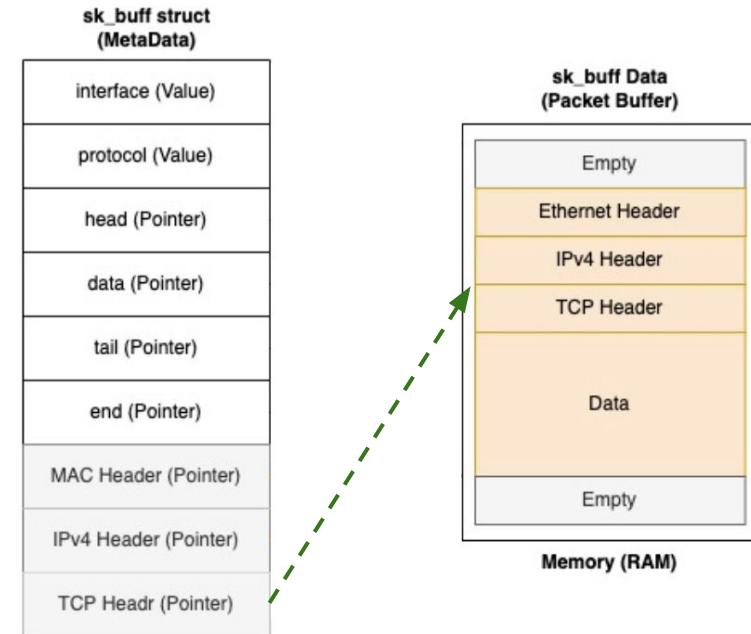


- пишет данные в сокет
- вызывает сисколл

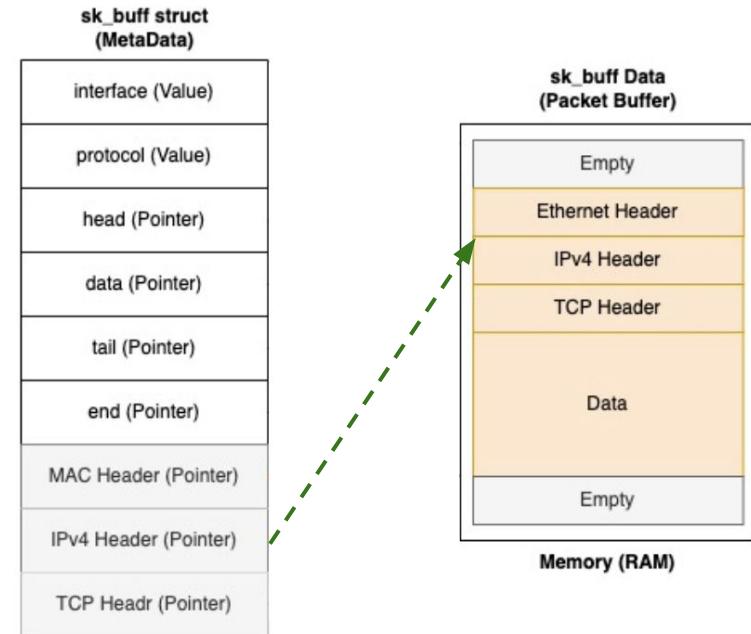
- создаёт **sk\_buff**
- помещает skb в очередь **tcp write**



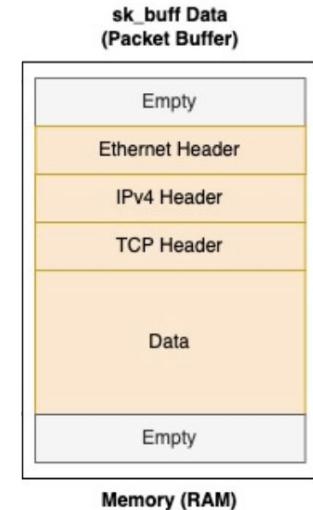
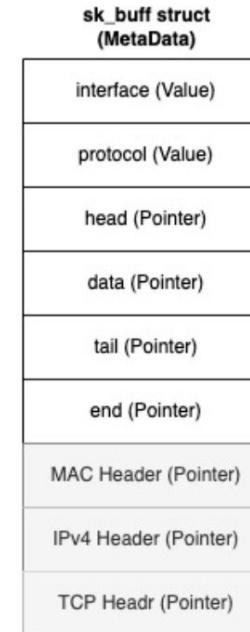
- формирует **TCP Header** (src,dst port, etc)
- записывает TCP Header в skb data
- заполняет указатель **transport\_header** (TCP Header)
- вызывает функцию L3 (знаем какую из сокета)



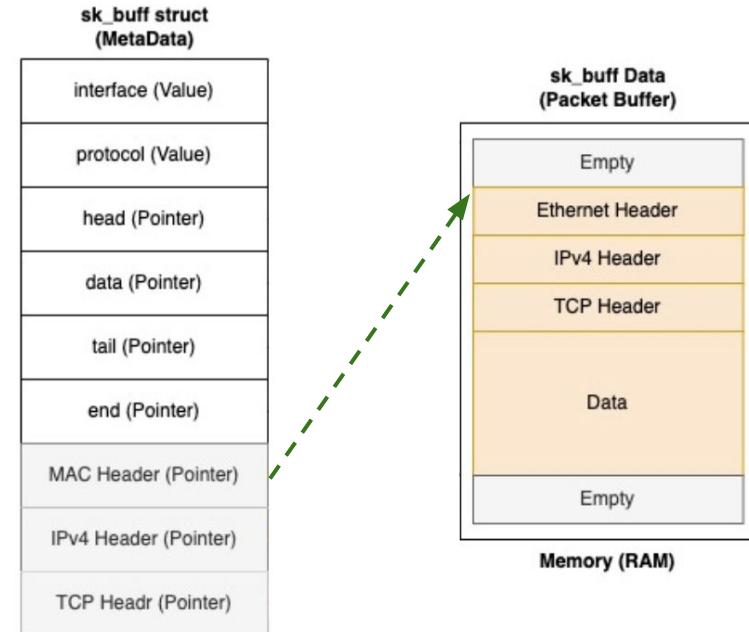
- проверяет **маршрутизацию**
- формирует **IPv4 Header**
- добавляет IPv4 Header в skb data
- заполняет указатель **network\_header** (IPv4 Header)
- вызывает route output function (e.g. IP out)



- дозаполняет метадату в skb
- вызывает **netfilter** (POST\_ROUTING)
- фрагментирует пакет, если нужно
- определяет **DST MAC** (L2 address resolution)
- формирует **Ethernet Header**
- добавляет Ethernet header в skb data
- вызывает L2 send function

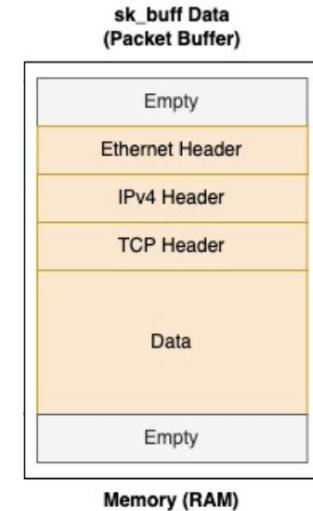


- заполняет указатель **mac\_header** (Ethernet header)
- вызывает **tc egress**
- помещает пакет в очередь queue discipline (**qdisc**)

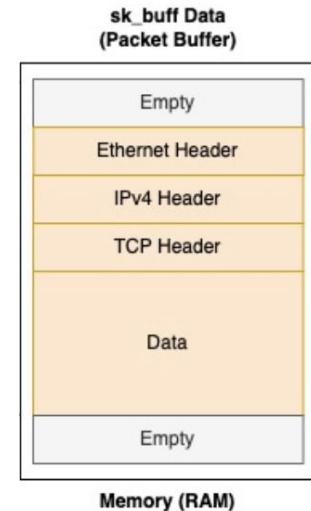
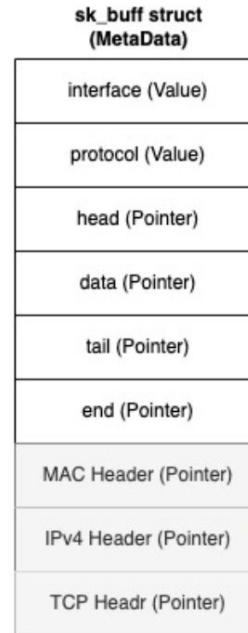


- достает пакет из очереди
- **передает пакет NIC**, если в его буфере есть место
- вызывает NIC driver send function

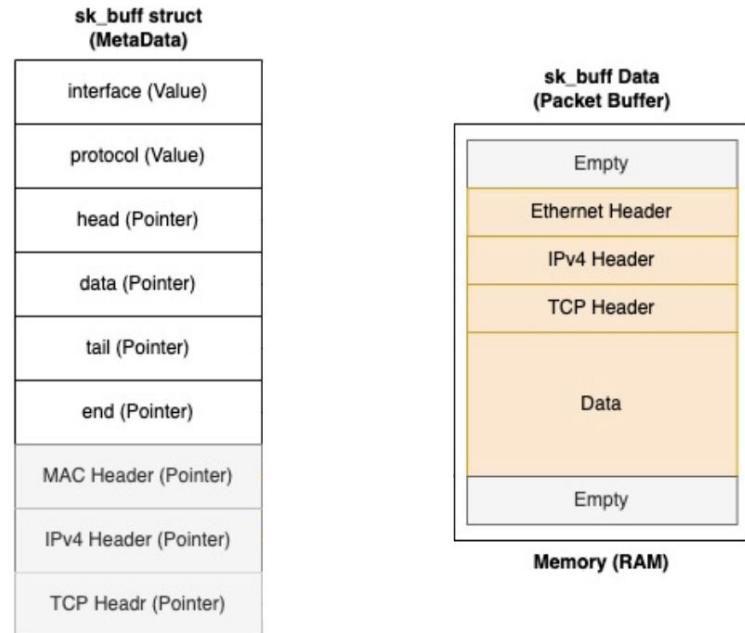
sk_buff struct (MetaData)	
interface (Value)	
protocol (Value)	
head (Pointer)	
data (Pointer)	
tail (Pointer)	
end (Pointer)	
MAC Header (Pointer)	
IPv4 Header (Pointer)	
TCP Headr (Pointer)	



- управляет очередью (информирует qdisc, если заполнена полная)
- когда очередь наполнена информирует NIC начать передачу



- **отправляет** пакет
- информирует об этом через прерывание
- **удаляет skb**



# Сокеты

- **Stream Sockets (SOCK\_STREAM):**

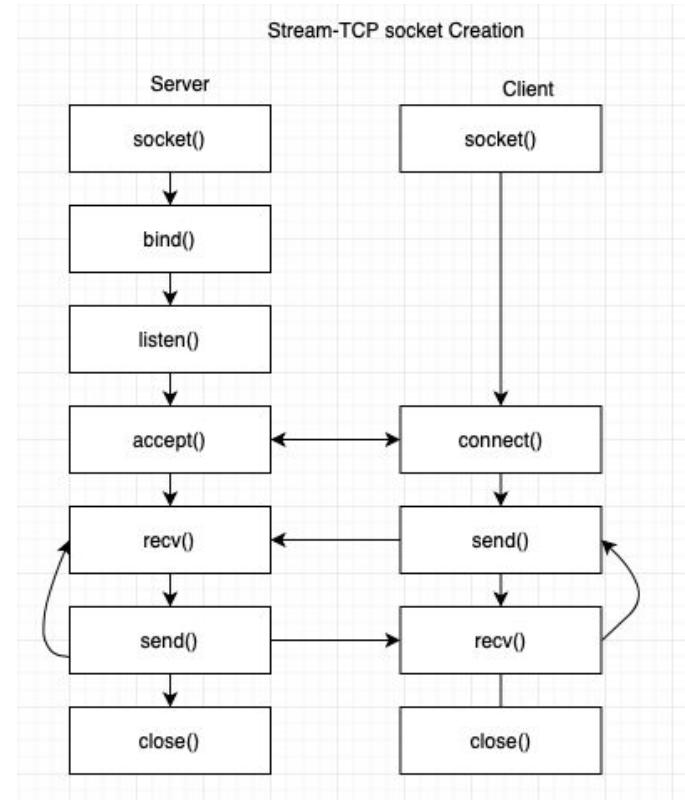
Used for TCP connections

- **Datagram Sockets (SOCK\_DGRAM):**

Used for UDP connections

- **Raw Sockets (SOCK\_RAW):**

Provide access to lower-level network protocols



# Сокеты

<https://man7.org/linux/man-pages/man7/socket.7.html>

## Common SOCK\_STREAM Options

1. **SO\_REUSEADDR**: Allows reuse of local addresses.

```
int opt = 1;  
setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
```

2. **SO\_KEEPALIVE**: Keeps connections alive by sending periodic messages.

```
int opt = 1;  
setsockopt(sockfd, SOL_SOCKET, SO_KEEPALIVE, &opt, sizeof(opt));
```

3. **TCP\_NODELAY**: Disables Nagle's algorithm for low-latency communication.

```
int opt = 1;  
setsockopt(sockfd, IPPROTO_TCP, TCP_NODELAY, &opt, sizeof(opt));
```

4. **SO\_LINGER**: Controls socket closure behavior.

```
struct linger sl = {1, 10}; // Enable linger, 10 seconds  
setsockopt(sockfd, SOL_SOCKET, SO_LINGER, &sl, sizeof(sl));
```

5. **SO\_RCVBUF and SO\_SNDBUF**: Set receive and send buffer sizes.

```
int bufsize = 8192; // 8 KB  
setsockopt(sockfd, SOL_SOCKET, SO_RCVBUF, &bufsize, sizeof(bufsize));  
setsockopt(sockfd, SOL_SOCKET, SO_SNDBUF, &bufsize, sizeof(bufsize));
```

Сейчас в рамках  на даунлоад серверах установлено значение `tcp_retries2=4`. Есть предположение, что для хороших каналов с редкими флапами (например, Lite интернет,  $\text{rtt} \sim 40-50$  мс) это значение мало, так как последний ретрай случается через 3 секунды, а tcp соединение помирает через 6. Также `net.ipv4.tcp_slow_start_after_idle` сейчас установлено 1, что приводит к сужению окна между последовательными запросами клиента за сегментами видео (если интервал между запросами больше  $rto \sim 200$  мс).

Предлагаю поставить на серверах групп `video-download` и `cdn-video` значения:

```
net.ipv4.tcp_retries2=6  
net.ipv4.tcp_slow_start_after_idle=0
```

## Итоги раздела:

- Linux сложный
- Если хотим писать сетевой хайлод, придётся вникнуть



# Вопросы?



# Настройка и диагностика сети



# ip (iproute2)

Просмотр всех интерфейсов:

**ip link show**

Включение интерфейса:

**ip link set dev <интерфейс> up**

Отключение интерфейса:

**ip link set dev <интерфейс> down**

Назначение IP-адреса:

**ip addr add <IP-адрес>/<маска> dev <интерфейс>**

Удаление IP-адреса:

**ip addr del <IP-адрес>/<маска> dev <интерфейс>**

Просмотр таблицы маршрутизации:

**ip route show**

Добавление статического маршрута:

**ip route add <сеть>/<маска> via <шлюз> dev <интерфейс>**

Удаление маршрута:

**ip route del <сеть>/<маска> via <шлюз> dev <интерфейс>**

Замена маршрута:

**ip route replace <сеть>/<маска> via <шлюз> dev <интерфейс>**

# Systemd-networkd

```
[root@host ~]# cat /etc/systemd/network/lan104.netdev
```

```
[NetDev]
Name=lan104
Kind=vlan
```

```
[VLAN]
Id=104
```

```
[root@host ~]# cat /etc/systemd/network/lan104.network
```

```
[Match]
Name=lan104
Type=vlan
```

```
[Network]
LinkLocalAddressing=ipv6
IPv6LinkLocalAddressGenerationMode=eui64
IPv6AcceptRA=no
KeepConfiguration=static
```

```
[Address]
Address=10.88.200.137/31
```

```
[Address]
Address=fd00:1148:9000:9000:111:125:1:402/120
```

# Systemd-networkd

```
[root@host ~]# tree /etc/systemd/network
/etc/systemd/network
├── dummy0.netdev
├── dummy0.network
├── eth0.network
├── eth1.network
├── eth2.network
├── eth3.network
└── lan104.netdev
    ├── lan104.network
    └── lan104.network.d
        ├── route-10_0_0_0_8.conf
        ├── route-100_64_0_0_10.conf
        ├── route-172_16_0_0_12.conf
        ├── route-185_100_104_22.conf
        ├── route-185_16_149_154.conf
        ├── route-185_16_244_29.conf
        └── route-192_168_0_0_16.conf
```

# Systemd-networkd

```
[root@host ~]# cat /etc/systemd/network/lan105.network.d/route-default.conf
```

```
[Route]
PreferredSource=10.216.104.42
Gateway=10.88.236.10
```

```
root@host ~]# cat /etc/systemd/network/lan100.network.d/route-default.conf
```

```
[Route]
PreferredSource=185.226.55.142
MultiPathRoute=10.88.200.128@lan100
MultiPathRoute=10.88.200.130@lan101
MultiPathRoute=10.88.200.132@lan102
MultiPathRoute=10.88.200.134@lan103
```

# Systemd-networkd

```
[root@host ~]# networkctl status
```

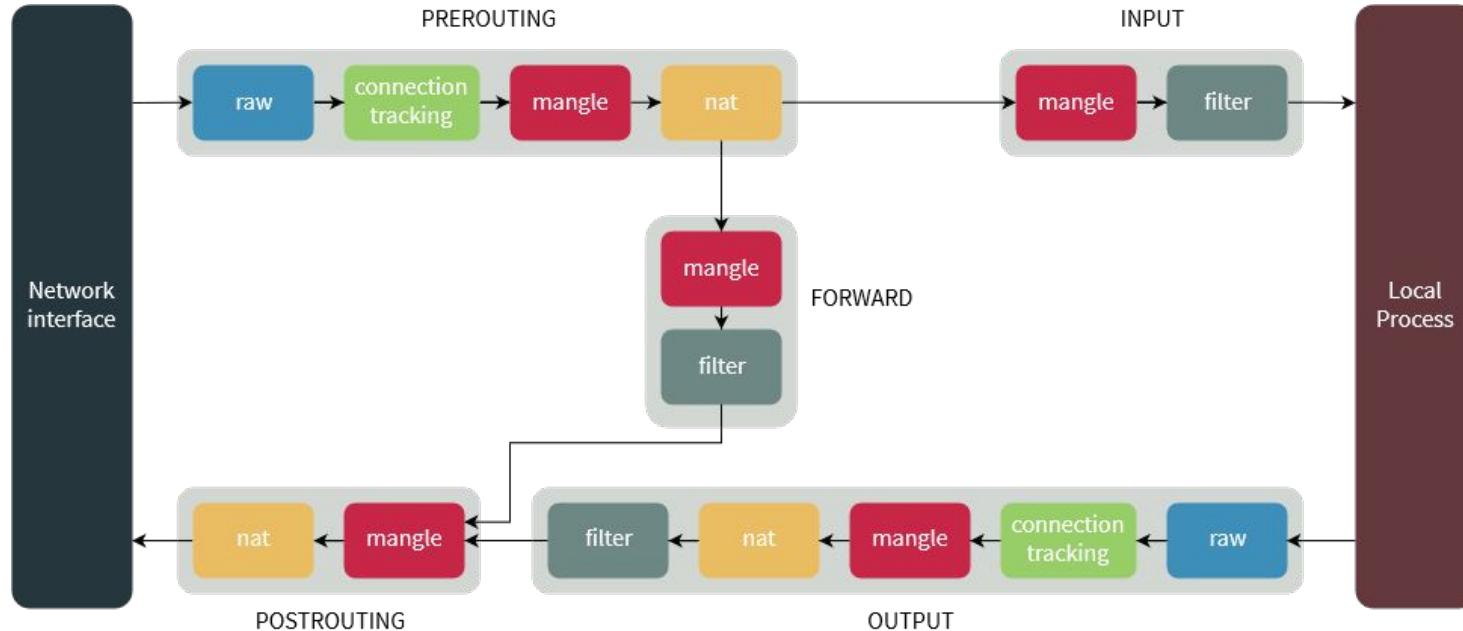
- State: routable  
Online state: online  
Address: 10.216.104.42 on dummy0  
10.88.236.11 on lan105  
fd00:1148:9000:9000:111:127:12:502 on lan105  
fe80::eaeb:d3ff:fe82:2bba on eth1  
fe80::c5b:8bff:febe:bae9 on dummy0  
fe80::eaeb:d3ff:fe82:2bba on lan105  
Gateway: 10.88.236.10 on lan105

```
[root@host ~]# systemctl status systemd-networkd
```

- systemd-networkd.service - Network Configuration  
Loaded: loaded (/usr/lib/systemd/system/systemd-networkd.service; enabled; preset: enabled)  
Drop-In: /etc/systemd/system/systemd-networkd.service.d  
└─networkd-dispatcher.conf  
Active: active (running) since Mon 2024-09-23 12:34:07 MSK; 1 week 0 days ago  
TriggeredBy: • systemd-networkd.socket  
...

# iptables (netfilter)

Утилита для управления цепочками правил, применяемых к сетевым пакетам



```
[root@host inputs]# iptables -S | grep ACR-137254

-A LAN_IN -s 10.21.148.194/32 -p tcp -m tcp --dport 443 -m comment --comment "ACR-137254 statscollector.back-dev" -j RETURN

-A LAN_IN -s 10.21.148.193/32 -p tcp -m tcp --dport 443 -m comment --comment "ACR-137254 one-back.back-dev" -j RETURN

-A LAN_OUT -d 10.21.148.194/32 -p tcp -m tcp --sport 443 --dport 1024:65535 -m tcp ! --tcp-flags FIN,SYN,RST,ACK SYN -m comment --comment "ACR-137254 statscollector.back-dev" -j ACCEPT

-A LAN_OUT -d 10.21.148.193/32 -p tcp -m tcp --sport 443 --dport 1024:65535 -m tcp ! --tcp-flags FIN,SYN,RST,ACK SYN -m comment --comment "ACR-137254 one-back.back-dev" -j ACCEPT
```

```
[root@host inputs]# iptables -nvL | grep ACR-137254

Chain LAN_OUT (1 references)
pkts bytes target      prot opt in     out      source          destination
   11  1334 RETURN      tcp   --  *       *      10.21.148.194      0.0.0.0/0          tcp  dpt:443 /* ACR-137254
statscollector.back-dev */

      0      0 RETURN      tcp   --  *       *      10.21.148.193      0.0.0.0/0          tcp  dpt:443 /* ACR-137254
one-back.back-dev */

Chain LAN_OUT (1 references)
pkts bytes target      prot opt in     out      source          destination
    8  8263 ACCEPT      tcp   --  *       *      0.0.0.0/0          10.21.148.194      tcp  spt:443 dpts:1024:65535 tcp
flags:!0x17/0x02 /* ACR-137254 statscollector.back-dev */

      0      0 ACCEPT      tcp   --  *       *      0.0.0.0/0          10.21.148.193      tcp  spt:443 dpts:1024:65535 tcp
flags:!0x17/0x02 /* ACR-137254 one-back.back-dev */
```

# iptables

Просмотр всех правил:

```
iptables -nvL --line-numbers
```

Добавление правила для разрешения входящих соединений на порт 80 (HTTP):

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Блокировка всех входящих соединений с определенного IP-адреса:

```
iptables -A INPUT -s <IP-адрес> -j DROP
```

Перенаправление трафика с одного порта на другой:

```
iptables -t nat -A PREROUTING -p tcp --dport <старый_порт> -j REDIRECT --to-port <новый_порт>
```

Удаление правила:

```
iptables -D INPUT <номер_строки>
```

Сохранение правил iptables:

```
iptables-save > /etc/iptables/rules.v4
```

# bird

```
[root@host ~]# birdc show protocols  
BIRD 2.15 ready.
```

Name	Proto	Table	State	Since	Info
device1	Device	---	up	2024-09-05	
direct1	Direct	---	up	2024-09-05	
kernel1	Kernel	master4	up	2024-09-05	
dote_0	BGP	---	up	2024-09-05	Established
dote_1	BGP	---	up	2024-09-05	Established
dote_2	BGP	---	up	2024-09-05	Established
dote_3	BGP	---	up	2024-09-05	Established

```
router id 192.0.2.1;  
  
protocol bgp {  
    local as 65001;  
    neighbor 198.51.100.1 as 65002;  
    import all;  
    export filter {  
        if net ~ [ 192.0.2.0/24+ ] then accept;  
        reject;  
    };  
    source address 192.0.2.1;  
}
```

# bird

Перезагрузить конфигурацию без перезапуска BIRD:

`birdc configure`

Показать текущий статус BIRD:

`birdc show status`

Показать все настроенные протоколы:

`birdc show protocols`

Показать подробную информацию о конкретном протоколе:

`birdc show protocols <имя_протокола> all`

Показать маршруты:

`birdc show routes`



# ethtool

Просмотр информации об интерфейсе:

```
ethtool eth0
```

Просмотр информации о драйвере:

```
ethtool -i eth0
```

Изменение скорости и режима дуплекса:

```
ethtool -s eth0 speed 1000 duplex full autoneg on
```

Просмотр статистики:

```
ethtool -S eth0
```

# ethtool

```
[root@host ~]# ethtool -i eth0
driver: mlx5_core
version: 6.6.34-1.ehot.el9.x86_64
firmware-version: 14.32.1010
(MT_2420110034)
expansion-rom-version:
bus-info: 0000:b1:00.1
supports-statistics: yes
supports-test: yes
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: yes
```

```
[root@host ~]# ethtool eth0
Settings for eth0:
Supported ports: [ Backplane ]
Supported link modes: 1000baseKX/Full
                                         10000baseKR/Full
                                         25000baseCR/Full
                                         25000baseKR/Full
                                         25000baseSR/Full
Supported pause frame use: Symmetric
Supports auto-negotiation: Yes
Supported FEC modes: None    RS    BASER
Advertised link modes: 1000baseKX/Full
                                         10000baseKR/Full
                                         25000baseCR/Full
                                         25000baseKR/Full
                                         25000baseSR/Full
Advertised pause frame use: Symmetric
Advertised auto-negotiation: Yes
Advertised FEC modes: BASER
Speed: 25000Mb/s
Duplex: Full
Auto-negotiation: on
Port: Direct Attach Copper
PHYAD: 0
Transceiver: internal
Supports Wake-on: d
Wake-on: d
Link detected: yes
```

# lldpctl

```
[root@host ~]# lldpctl eth0
-----
LLDP neighbors:
-----
Interface: eth0, via: LLDP, RID: 1, Time: 24 days, 21:08:22
Chassis:
  ChassisID: mac 40:9e:a4:75:2c:00
  SysName: XXXXXXXX
  SysDescr: Juniper Networks, Inc. qfx5120-48y-8c Ethernet Switch, kernel JUNOS
22.2R3-S1.9, Build date: 2023-06-10 11:13:02 UTC Copyright (c) 1996-2023 Juniper
Networks, Inc.
  Capability: Bridge, on
  Capability: Router, on
Port:
  PortID: ifname et-0/0/0
  PortDescr: et-0/0/0
  TTL: 120
  MFS: 9216
VLAN: 100 vlan-100
VLAN: 3000, pvid: yes vlan-3000
LLDP-MED:
  Device Type: Network Connectivity Device
  Capability: Capabilities, yes
  Capability: Policy, yes
...
```

# ping, traceroute, mtr

```
[root@host ~]# ping -h

Options:

-c <count>          stop after <count> replies
-I <interface>       either interface name or address
-q                  quiet output

-s <size>           use <size> as number of data bytes to be sent
-w <deadline>        reply wait <deadline> in seconds
-W <timeout>         time to wait for response

[root@cspc ~]# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=60 time=1.06 ms
^C
--- 1.1.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.069/1.069/1.069/0.000 ms
```

# ping, traceroute, mtr

```
[root@host ~]# tcptraceroute 10.107.16.70 443
```

Running:

```
traceroute -T -O info -p 443 10.107.16.70
```

```
traceroute to 10.107.16.70 (10.107.16.70), 30 hops max, 60 byte packets
```

```
1 * * *
```

```
2 * * *
```

```
3 10.198.211.72 (10.198.211.72) 0.386 ms 10.198.211.0 (10.198.211.0) 0.364 ms 10.198.211.64 (10.198.211.64) 1.392 ms
```

```
4 10.255.21.26 (10.255.21.26) 2.965 ms 10.255.19.6 (10.255.19.6) 2.088 ms 10.255.19.10 (10.255.19.10) 3.504 ms
```

```
5 10.255.242.194 (10.255.242.194) 1.562 ms 10.255.242.0 (10.255.242.0) 1.587 ms 10.255.242.10 (10.255.242.10) 1.610 ms
```

```
6 10.255.242.195 (10.255.242.195) 1.341 ms 2.322 ms 2.311 ms
```

```
7 10.255.223.114 (10.255.223.114) 1.725 ms 1.707 ms 1.516 ms
```

```
8 10.107.16.70 (10.107.16.70) <syn,ack> 1.395 ms 1.101 ms 1.370 ms
```

# ping, traceroute, mtr

```
My traceroute [v0.95]
Andreys-MBP (10.0.0.75) -> 1.1.1.1 (1.1.1.1)                               2024-09-22T21:33:55+0300
Keys: Help   Display mode   Restart statistics   Order of fields   quit
                                                Packets                  Pings
Host                                         Loss%     Snt    Last     Avg     Best    Wrst   StDev
1. mi-r3g                                     0.0%     67     2.2     2.5     2.0    9.4    1.0
2. (waiting for reply)
3. router.sknt.ru                           3.0%     67     2.8     2.8     2.5    3.5    0.2
4. (waiting for reply)
5. spx-ix.as13335.net                      86.4%    67     3.7    10.2     3.5    51.7   15.9
6. one.one.one.one                         0.0%     67     3.2     3.0     2.6    3.7    0.2
```

# netcat, telnet

```
[andrey.trofimov@host ~]$ nc -vz ns1.ok.ru 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 5.61.23.9:53.
Ncat: 0 bytes sent, 0 bytes received in 0.04
seconds.
```

```
[andrey.trofimov@host ~]$ nc -vz6 ns1.ok.ru 53
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Connected to 2a00:b4c0:1b0:6:::53.
Ncat: 0 bytes sent, 0 bytes received in 0.04
seconds.
```

```
[andrey.trofimov@host ~]$ telnet ns1.ok.ru 53
Trying 5.61.23.9...
Connected to ns1.ok.ru.
Escape character is '^]'.
^]
telnet> Connection closed.
```

```
[andrey.trofimov@host ~]$ telnet -6 ns1.ok.ru 53
Trying 2a00:b4c0:1b0:6:::53
Connected to ns1.ok.ru.
Escape character is '^]'.
^]
telnet> Connection closed.
```

# netstat, ss

```
[root@host ~]# ss -lnt
```

State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
LISTEN	0	128	127.0.0.1:9090	*:*
LISTEN	0	128	<b>127.0.0.1:9092</b>	*:*
LISTEN	0	128	*:22	*:*
LISTEN	0	100	127.0.0.1:25	*:*
LISTEN	0	128	[::1]:9091	[::]:*
LISTEN	0	128	[::]:46788	[::]:*

# nslookup, dig, host

```
[root@host ~]# nslookup vk.com
```

Server: 8.8.8.8

Address: 8.8.8.8#53

Non-authoritative answer:

Name: vk.com

Address: 87.240.137.164

Name: vk.com

Address: 87.240.132.72

```
[root@host ~]# dig +short facebook.com @1.1.1.1 AAAA
```

2a03:2880:f113:81:face:b00c:0:25de

```
[root@host ~]# host facebook.com
```

facebook.com has address 157.240.205.35

facebook.com has IPv6 address

2a03:2880:f113:81:face:b00c:0:25de

facebook.com mail is handled by 10  
smtpin.vvv.facebook.com.

# tcpdump

```
[root@host ~]# tcpdump -nni eth0 icmp
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes

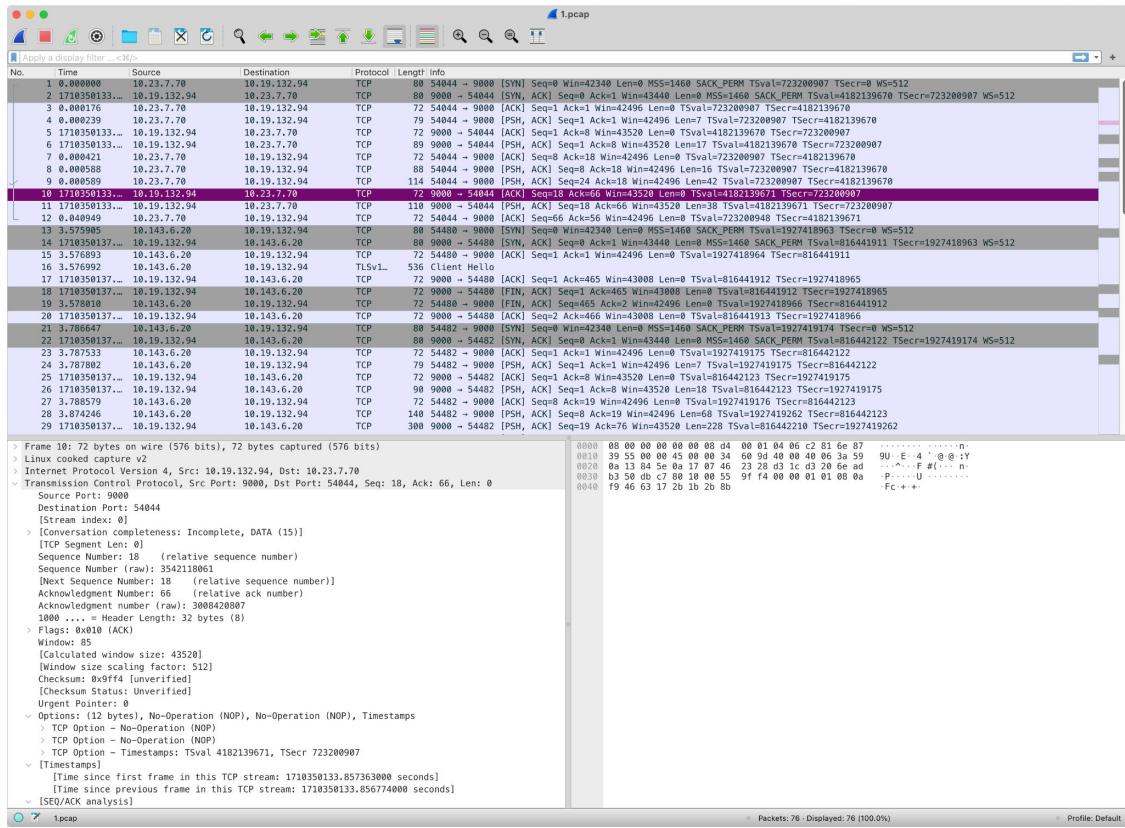
16:23:14.806786 IP 178.178.230.5 > 185.226.55.142: ICMP 178.178.230.5 tcp port 32126 unreachable, length
72
16:23:14.806789 IP 178.178.230.5 > 185.226.55.142: ICMP 178.178.230.5 tcp port 26550 unreachable, length
72
16:23:15.326973 IP 94.41.249.22 > 185.226.55.142: ICMP host 94.41.249.22 unreachable, length 84

[root@host ~]# tcpdump -nni eth1 host 10.216.107.3
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode

listening on eth1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
16:26:54.896283 IP 10.216.107.3 > 10.216.107.1: ICMP echo request, id 5, seq 1, length 64
16:26:54.896301 IP 10.216.107.1 > 10.216.107.3: ICMP echo reply, id 5, seq 1, length 64

[root@host ~]# tcpdump -nni eth1 host 10.216.107.3 -w /tmp/dump.pcap
```

# Wireshark



# Общий подход к проверке связи

Страна А

- Есть ли связь по ICMP
  - Есть ли связь по порту
- 
- ping
  - telnet / netcat

Страна Б

- Слушается ли порт
- 
- ss

# Общий подход к проверке связи

## Страна А

- Есть ли связь по ICMP
  - Правильный ли маршрут к Б
  - Корректный ли резолв
  - Есть ли связь по порту
  - Путь пакета и где последний хоп
  - Фаервол на хосте/в сети
- 
- ping
  - dig, host, nslookup
  - telnet / netcat
  - tcptraceroute
  - ip route get

## Страна Б

- Слушается ли порт
  - Приходят ли пакеты от А
  - Приходят ли пакеты от А с правильного адреса
  - Уходят ли ответы
  - Правильный ли маршрут к А
  - Фаервол на хосте/в сети
- 
- ss
  - tcpdump
  - ip route get

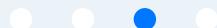
А с чего я взял, что это вообще должно работать?



# Вопросы?



# Актуальные сетевые технологии



# SDN

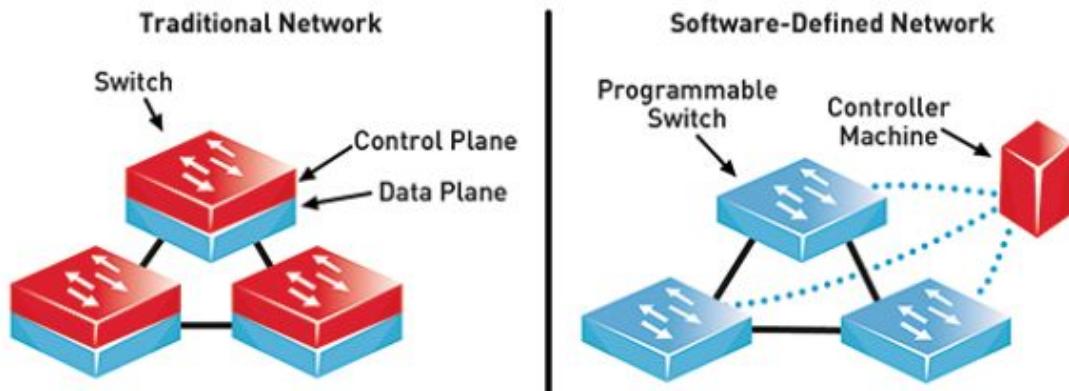
Software-Defined Networking

Обычная сеть:

- Независимые устройства с логикой (**дорого**)
- Вендорозависимо

SDN:

- Централизуем логику
- На доступе простые устройства
- Снижение затрат на поддержку
- **Экономия**

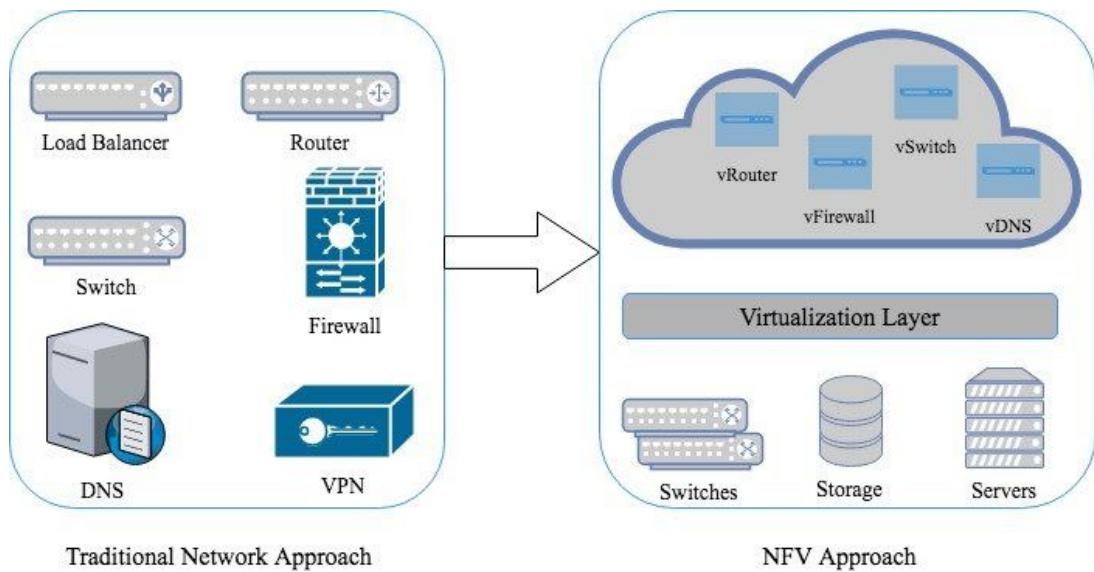


# NFV

## Network Functions Virtualization

Преимущества (те же что и для любой виртуализации):

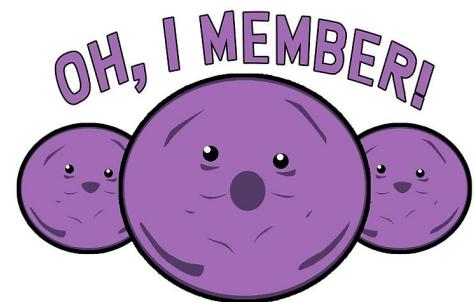
- Дёшево
- Гибко
- Удобно



# VXLAN

Virtual Extensible LAN

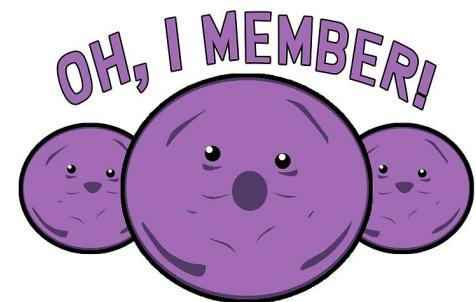
- L2 сегмент нужно минимизировать
- Сложно и нецелесообразно на больших сетях



# VXLAN

Virtual Extensible LAN

- L2 сегмент нужно минимизировать
  - Сложно и нецелесообразно на больших сетях
- 
- L2 это просто и удобно для клиентов



# VXLAN

Virtual Extensible LAN

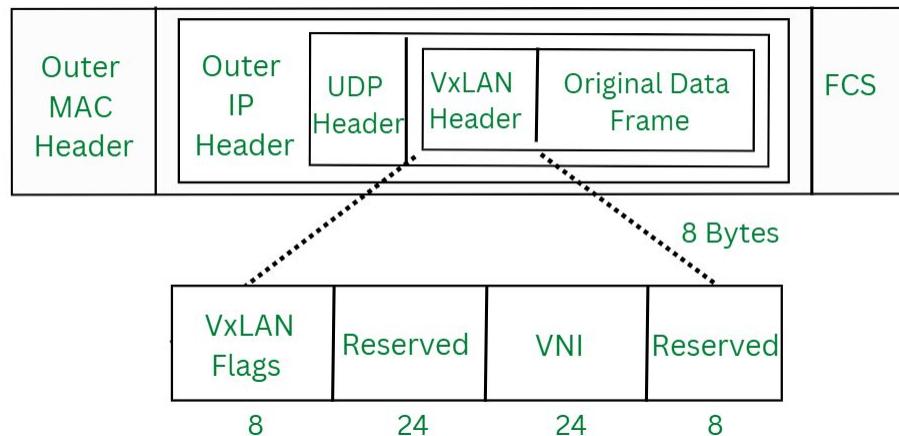
- L2 сегмент нужно минимизировать
- Сложно и нецелесообразно на больших сетях
- L2 это просто и удобно для клиентов

=>

Инкапсулируем L2 кадры в **UDP туннели**

поверх L3

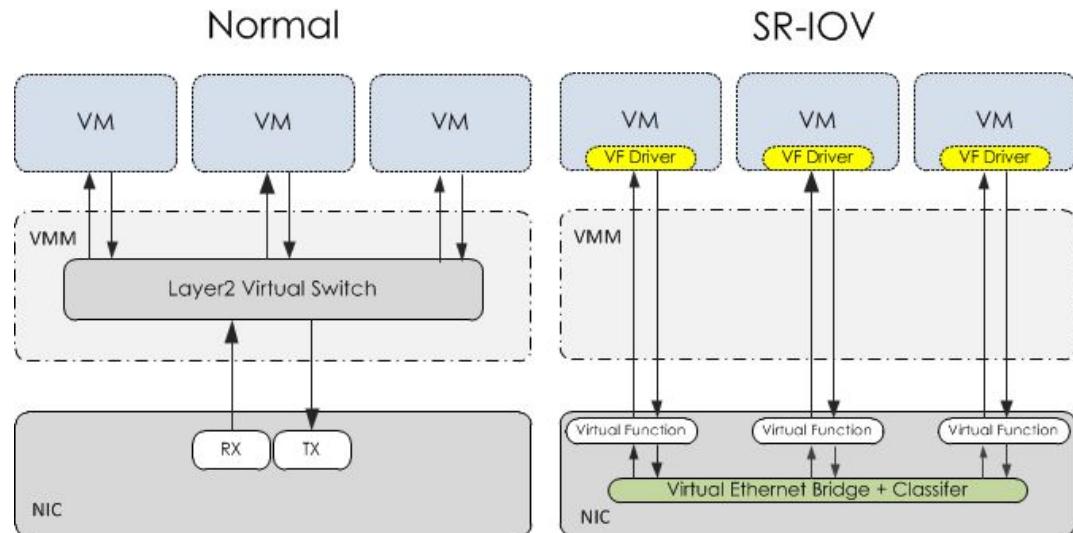
Минимизируем multi- и broadcast

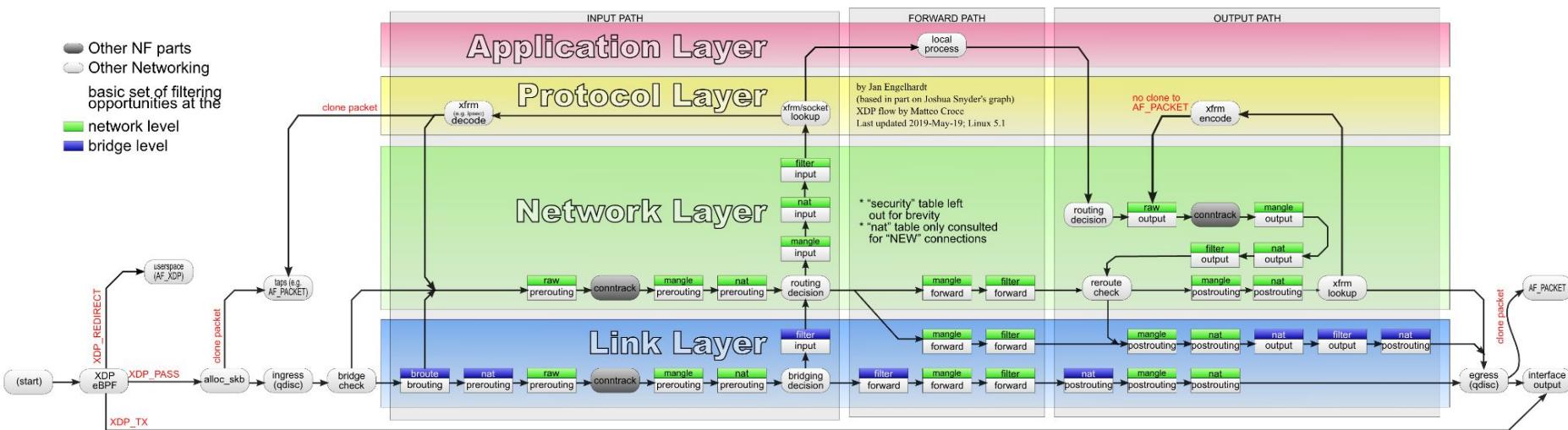


# SR-IOV

## Single Root I/O Virtualization

- Средствами драйвера делаем виртуальное устройство и отдаём его в ВМ

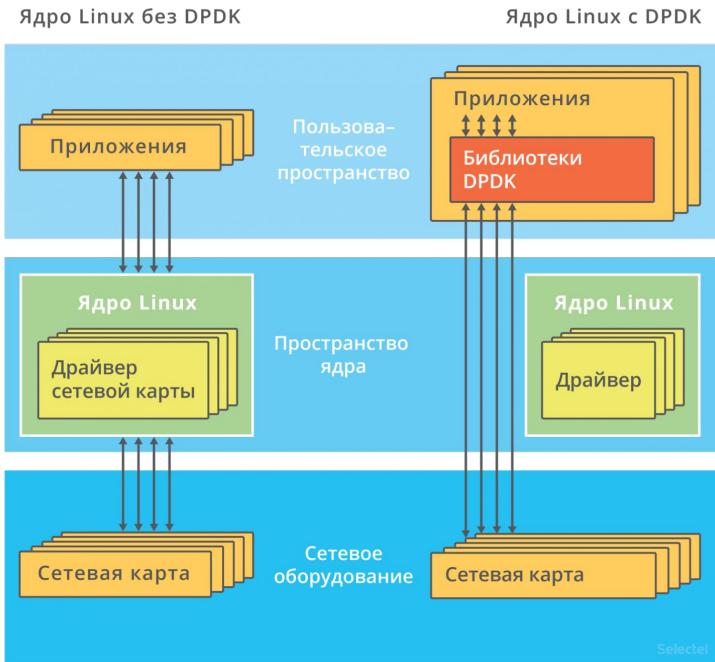




# DPDK

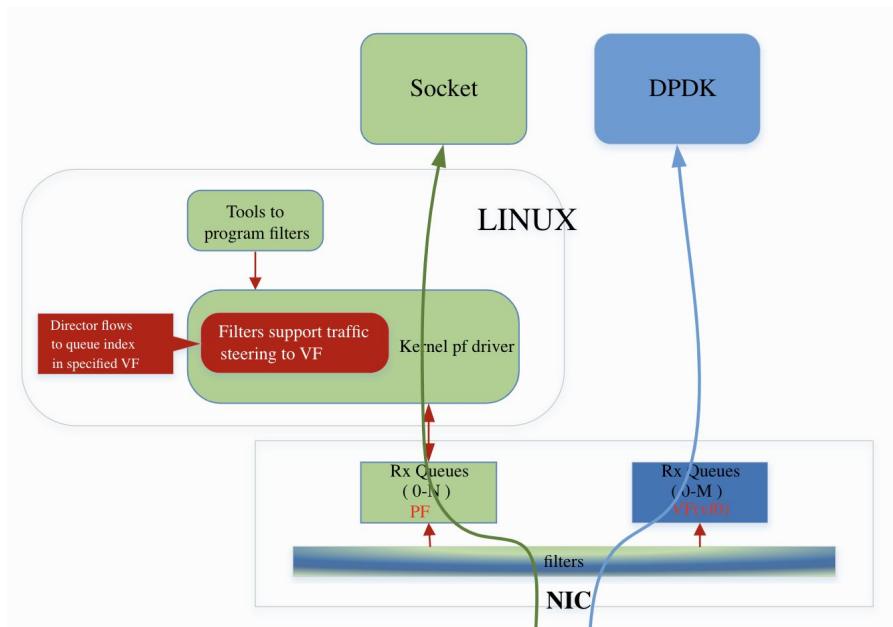
## Data Plane Development Kit

- Уносим сетевой стек в **userspace**



# DPDK

Data Plane Development Kit



## В лучших традициях



@GrandmasterPoi 6 years ago

Тема dpdk совсем не раскрыта, а ведь лекция для разработчиков была.



2



Reply

# eBPF

extended Berkeley Packet Filter

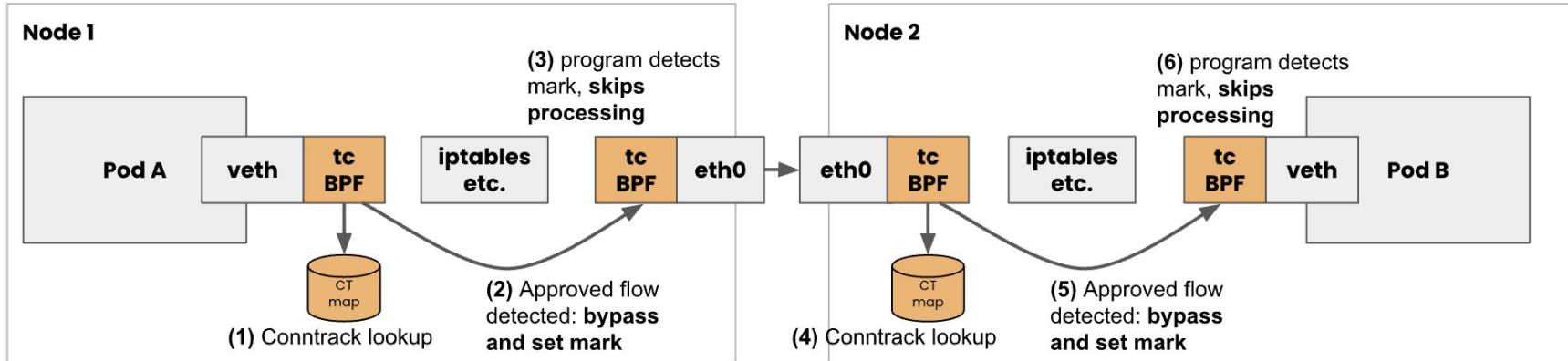
- Виртуальная машина в ядре
- Песочница и безопасность
- JIT-компиляция
- Гибкость и расширяемость
- Использование хуков

# eBPF

extended Berkeley Packet Filter

- Виртуальная машина в ядре
- Песочница и безопасность
- JIT-компиляция
- Гибкость и расширяемость
- Использование хуков
- Стильно
- Модно
- Молодёжно
- Счастье
- Здоровье
- Красота

# eBPF



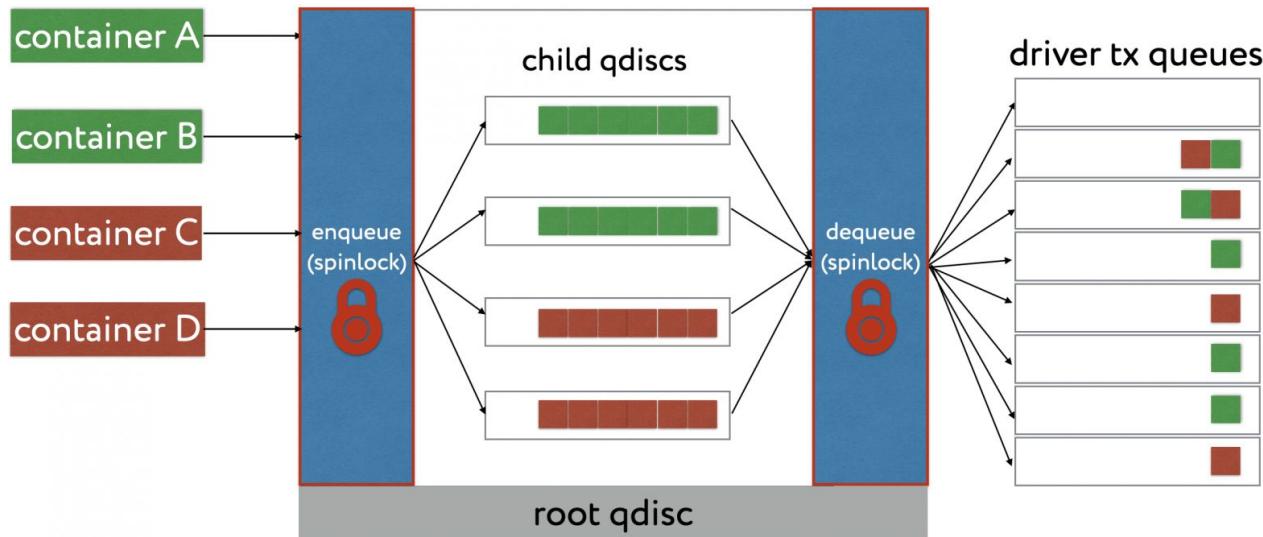
<https://docs.tigera.io/calico/latest/about/kubernetes-training/about-ebpf>

# eBPF

“DPDK наоборот”

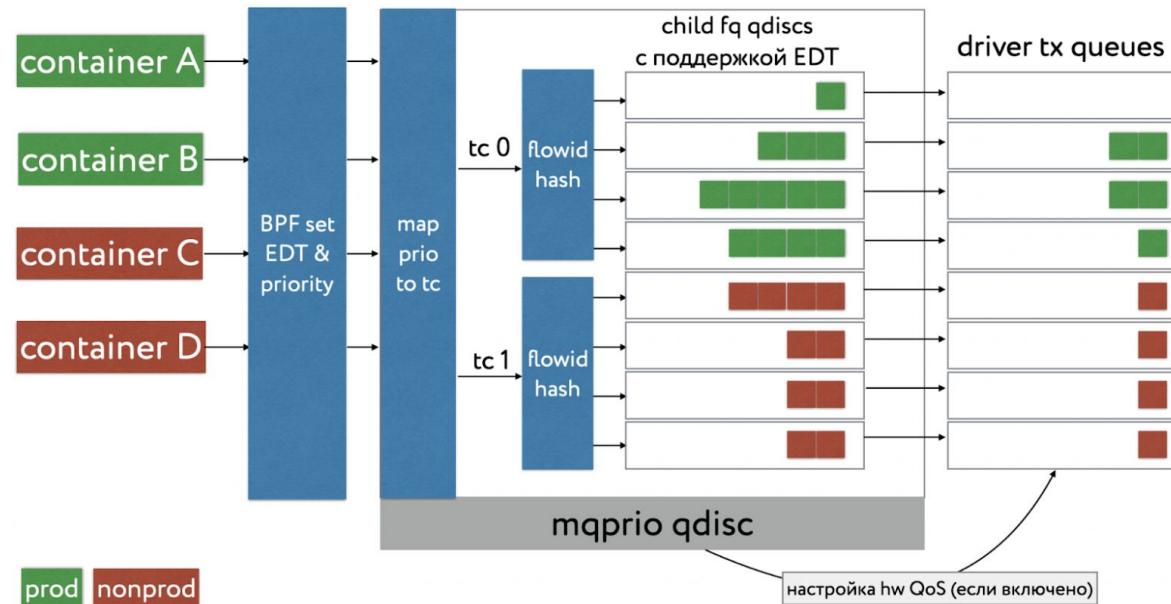
- Уровень работы:
  - **eBPF:** Работает на уровне ядра Linux, позволяя выполнять программы в контексте ядра для мониторинга, трассировки и фильтрации сетевого трафика.
  - **DPDK:** Работает на уровне пользовательского пространства, предоставляя набор библиотек и драйверов для высокопроизводительной обработки пакетов, обходя сетевой стек ядра.
- Цель:
  - **eBPF:** Основная цель — расширение функциональности ядра без изменения его кода, улучшение мониторинга и безопасности.
  - **DPDK:** Основная цель — максимизация пропускной способности и минимизация задержек при обработке сетевых пакетов.
- Использование ресурсов:
  - **eBPF:** Использует ресурсы ядра и встроенные хуки для выполнения задач.
  - **DPDK:** Использует выделенные ядра процессора и прямой доступ к сетевым интерфейсам для достижения высокой производительности.

# eBPF в One-cloud



Как мы оптимизировали сетевой шейпер Linux в облаке с помощью eBPF  
<https://habr.com/ru/companies/odnoklassniki/articles/572206>

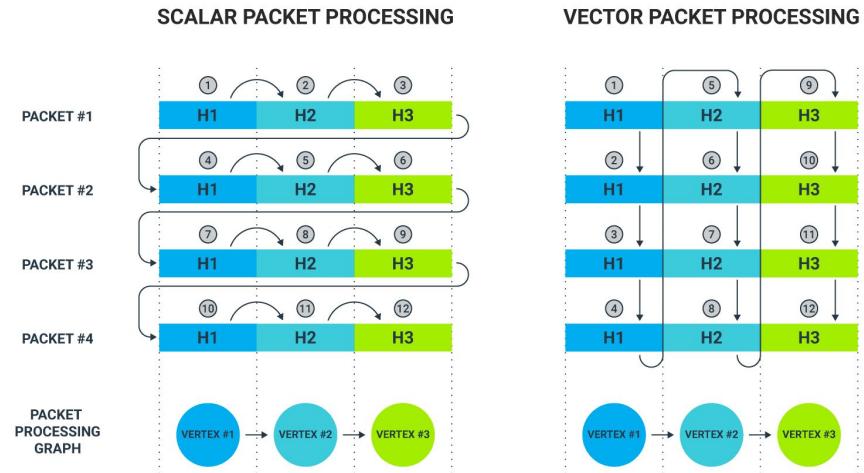
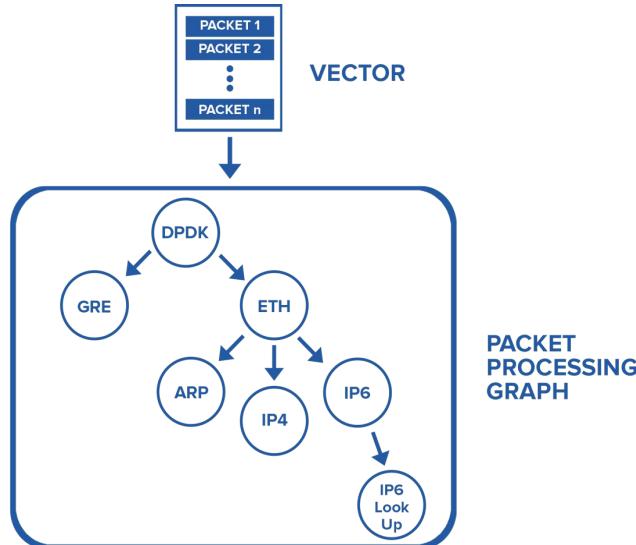
# eBPF в One-cloud



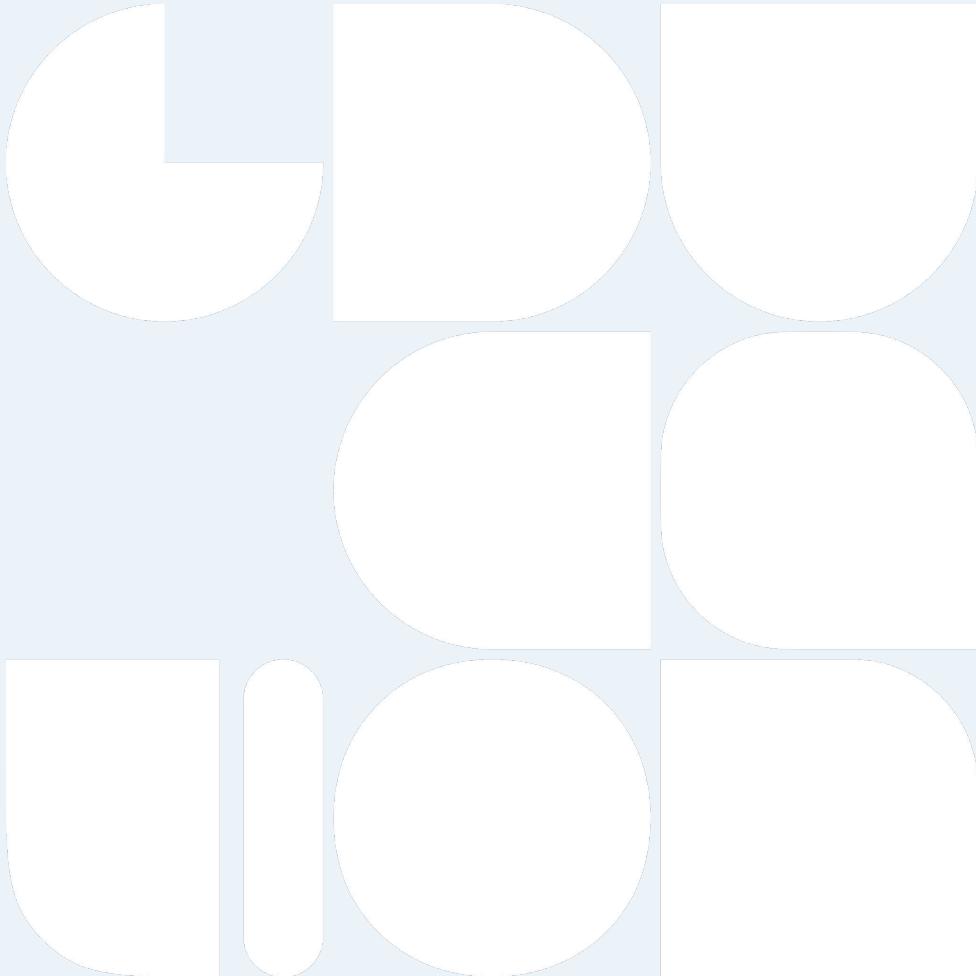
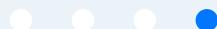
Как мы оптимизировали сетевой шейпер Linux в облаке с помощью eBPF  
<https://habr.com/ru/companies/odnoklassniki/articles/572206>

# VPP

## Vector Packet Processing



# Контейнеры и оркестраторы



# Базовая концепция

- Изоляция процессов/сетей и т.д
- Управление ресурсами

# Базовая концепция

- Изоляция процессов/сетей и т.д
- Управление ресурсами

Кстати, чем отличается контейнеризация от виртуализации?

# Docker

- **Host**
  - сеть хоста
- **Bridge**
  - виртуальный бридж (свитч), куда “подключены” контейнеры
- **IPvlan**
  - виртуальное L3 устройство в сети хоста
- **MACvlan**
  - виртуальное L2 устройство в сети хоста
- **Overlay**
  - VXLAN

# Kubernetes

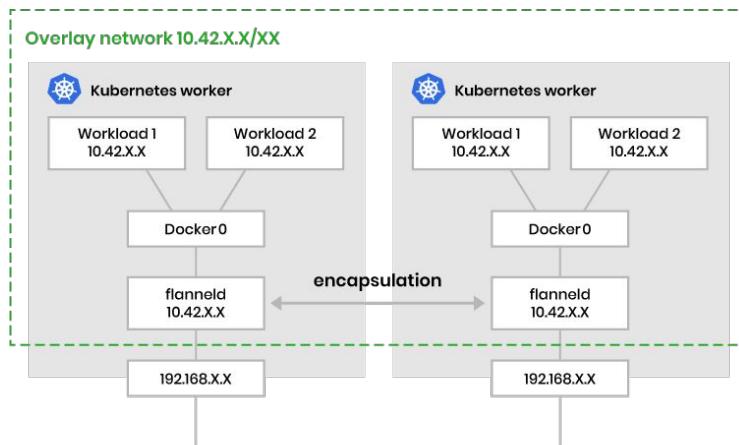
- Сетевые плагины, реализующие CNI (container-network interface)

<https://kubernetes.io/ru/docs/concepts/cluster-administration/addons/#networking-and-network-policy>

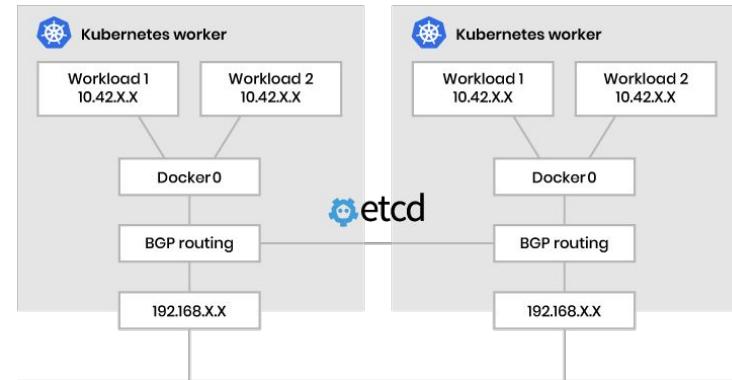
- Концепции примерно те же что в Docker
- Существенное отличие в том, что контейнеров много на разных хостах и им нужен интерконнект
- Всё это под управлением Kubernetes

# Kubernetes

- Encapsulated Networks



- Unencapsulated Networks



# Подведём итоги:

1. Передача данных — это процесс обмена информацией между устройствами в сети, который обеспечивается различными протоколами уровня 4 (UDP, TCP, QUIC), каждый из которых имеет свои особенности и области применения.
2. Уровень 3 (L3) включает протоколы IPv4 и IPv6, которые отвечают за маршрутизацию данных в сети. Маршрутизацию в сети обеспечивают протоколы вроде BGP и OSPF. Уровни 1 и 2 (L1/L2) включают физический уровень и Ethernet, обеспечивающие физическую передачу данных и управление доступом к среде передачи.
3. Настройка и диагностика сети в Linux требует знания ключевых команд и утилит, что позволяет эффективно управлять и устранять неполадки в сети.
4. Современные технологии, такие как SDN, NFV, VXLAN, SR-IOV, DPDK и eBPF, играют важную роль в развитии сетевой инфраструктуры, обеспечивая гибкость, масштабируемость и высокую производительность.



# Вопросы?



# ДЗ

## 1. Админское

- На двух ВМ создать виртуальное устройство dummy0 и настроить на нём /32 адрес из приватного диапазона
  - i. [1] Статически (файлом конфигурации в systemd-networkd/NetworkManager)
  - ii. [0.5] Динамически (командами ip...)
- Настроить маршрутизацию между /32
  - i. [1] bird + BGP
  - ii. [0.5] Статические маршруты
- Обменяться данными между /32 адресами этих ВМ

## 2. Программистское

- [3] Написать программу для DNS резолва (A/AAAA записи) через произвольный DNS сервер.  
Сетевое взаимодействие организовать через RAW сокет (**SOCK\_RAW**). Сформировать UDP пакет, отправить и получить ответ.

# ДЗ output

1. Конфиги/команды настройки
2. Исходники
3. Результаты

```
[root@vpc4 ~]# ip a sh dev dummy0
3: dummy0: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether fe:a1:ed:51:f0:18 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.2/32 scope global noprefixroute dummy0
        valid_lft forever preferred_lft forever
```

```
[root@vpc4 ~]# birdc show route table master4
BIRD 2.15.1 ready.
Table master4:
0.0.0.0/0          unicast [kernel1 12:36:13.771] * (10)
    via 172.16.43.2 on ens160
172.16.43.0/24      unicast [direct1 12:36:13.771] * (240)
    dev ens160
192.168.0.2/32      unicast [direct1 12:36:13.771] * (240)
    dev dummy0
192.168.0.1/32      unicast [vpc3 12:36:15.770] * (100) [AS65501i]
    via 172.16.43.130 on ens160
```

```
[root@vpc4 ~]# birdc show proto
BIRD 2.15.1 ready.
Name      Proto      Table      State   Since      Info
device1   Device     ---        up       12:36:13.767
direct1   Direct     ---        up       12:36:13.767
kernel1   Kernel     master4    up       12:36:13.767
vpc3      BGP        ---        up       12:36:15.724  Established
```

```
[root@vpc4 ~]# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=1.23 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.876 ms
```

```
[root@vpc4 ~]# ip r
default via 172.16.43.2 dev ens160 proto dhcp src 172.16.43.131 metric 100
172.16.43.0/24 dev ens160 proto kernel scope link src 172.16.43.131 metric 100
192.168.0.1 via 172.16.43.130 dev ens160 proto bird metric 32
192.168.0.2 dev dummy0 proto bird scope link metric 32
```

```
[root@vpc4 ~]# systemctl stop bird
```

```
[root@vpc4 ~]# ip r
default via 172.16.43.2 dev ens160 proto dhcp src 172.16.43.131 metric 100
172.16.43.0/24 dev ens160 proto kernel scope link src 172.16.43.131 metric 100
```

```
[root@vpc4 ~]# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
^C
--- 192.168.0.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 1999ms
```

Спасибо за  
внимание!

