ОНЛАЙН-ОБРАЗОВАНИЕ

Linux и сеть

Транспортный уровень

Александр Румянцев
Алексей Цыкунов

Чудеса отладки сетей в linux

https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt

1. **ARP отвечает по любому интерфейсу**

   Лечим так:

```
sysctl -w net.ipv4.conf.all.arp_filter = 1
sysctl -w net.ipv4.conf.all.arp_ignore = 2
```

2. Проверка обратного адреса. Если пакет пришел не с того интерфейса, на который указывает FIB (таблица маршрутизации), то пакет отбрасывается. При ассиметричном роутинге надо отключать (но только в таком случае!)

```
sysctl -w net.ipv4.conf.<интерфейс>.rp_filter = 2
```

3. По-умолчанию включена поддержка Zeroconf.

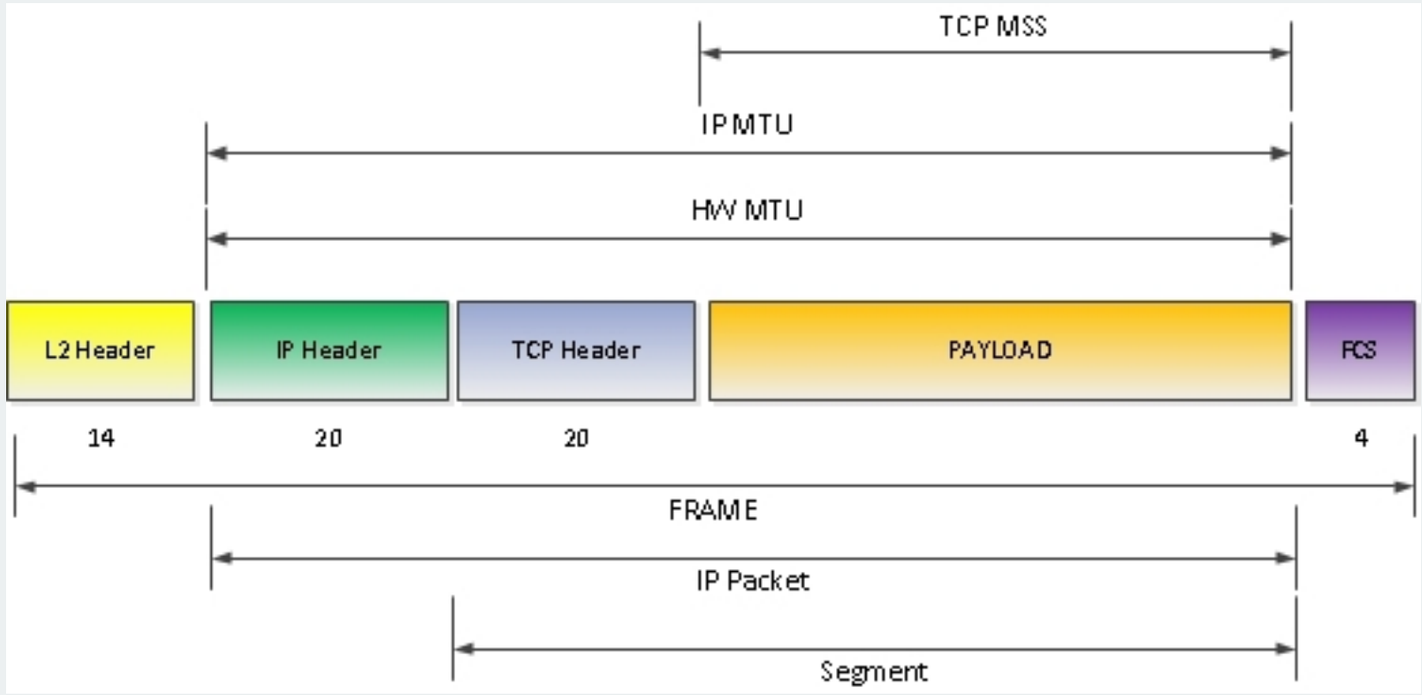
Лечим:

```
echo NOZEROCONF=yes >> /etc/sysconfig/network
```

З.Ы. этот файл должен существовать, иначе не взлетит

IP MTU / TCP MSS

В большинстве случаев имеется в виду IP MTU
Статья на тему: https://habr.com/post/226807/



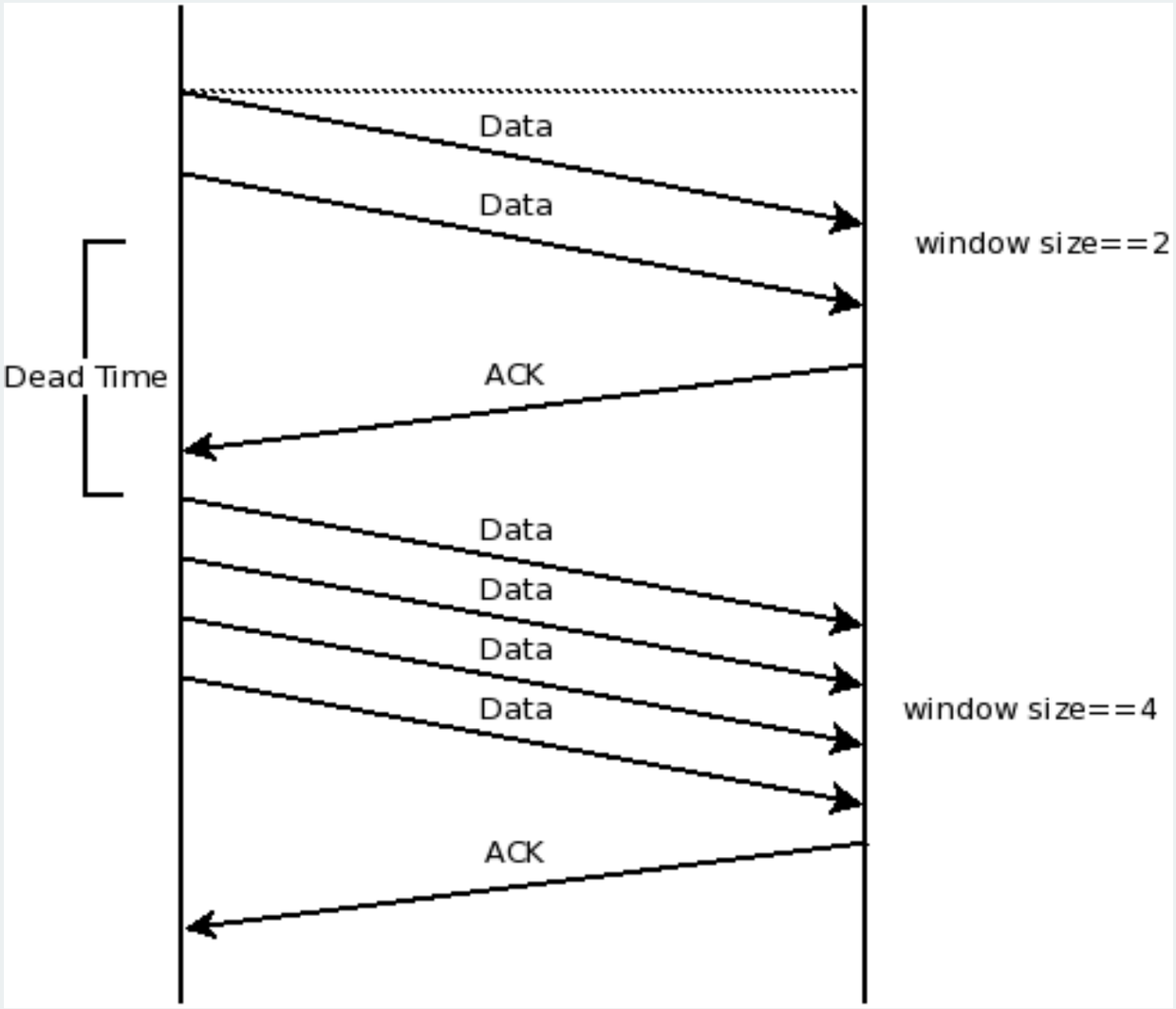
IP MTU / TCP MSS

Вручную проверяем подбором значения размера пакета. Linux всегда ставит DF.
`ping -s <size>`

Ядро само определяет и запоминает PATH MTU (алгоритм PMTUD,
потому и всегда DF установлен) для каждого элемента FIB
Но иногда ему надо помогать.

1. **Выставить MTU на интерфейсе**
2. **Уменьшить TCP MSS**
   `iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu`

TCP Window Size / Window Scaling



TCP Window Size / Window Scaling

https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt

На очень плохих каналах можно сразу отключить

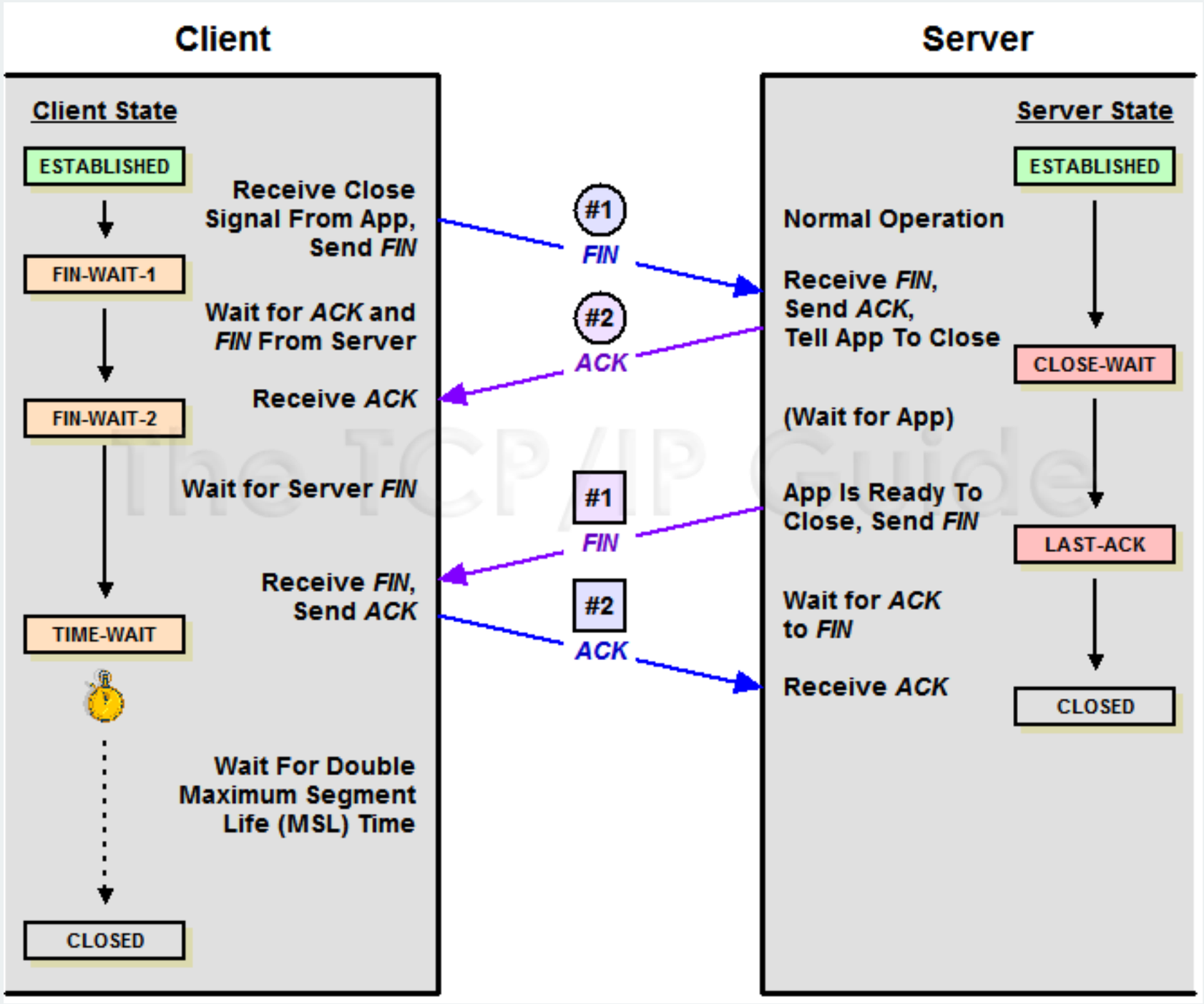
```
net.ipv4.tcp_window_scaling = 0
```

Косвено на window scaling влияет параметр `/proc/sys/net/ipv4/tcp_rmem`, который надо настраивать на системе

Socket backlog

https://linux.die.net/man/2/listen

`/proc/sys/net/core/somaxconn` - максимальное количество соединений в системе, оно же максимальный бэк-лог
`/proc/sys/net/ipv4/tcp_max_syn_backlog` - максимальное число одновременных попыток соединений


Client
Server
Client State
ESTABLISHED
Receive Close Signal From App, Send FIN
FIN-WAIT-1
Wait for ACK and FIN From Server
FIN-WAIT-2
Receive ACK
Wait for Server FIN
Receive FIN, Send ACK
TIME-WAIT
Wait For Double Maximum Segment Life (MSL) Time
CLOSED
#1
FIN
#2
ACK
#1
FIN
#2
ACK
Server State
ESTABLISHED
Normal Operation
Receive FIN, Send ACK, Tell App To Close
CLOSE-WAIT
(Wait for App)
App Is Ready To Close, Send FIN
LAST-ACK
Wait for ACK to FIN
Receive ACK
CLOSED

TIME_WAIT

https://vincent.bernat.im/en/blog/2014-tcp-time-wait-state-linux

Период ожидания потерявшихся пакетов, когда сокет еще не закрыт. По-умолчанию = 2*MSL = 1 минута. На высоконагруженных системах можно уменьшить до 10-15 секунд, меньше не стоит.

`/proc/sys/net/ipv4/tcp_fin_timeout`

Если вы всё равно испытываете трудности, то стоит навесить еще адресов для приёма и обработки трафика

Есть еще
`net.ipv4.tcp_tw_reuse = 0` **- переиспользовать сокеты для исходящих соединений**

На балансировщиках стоит еще увеличить
`/proc/sys/net/ipv4/ip_local_port_range` **- диапазон портов, доступных для исходящих соединений.**

VLAN

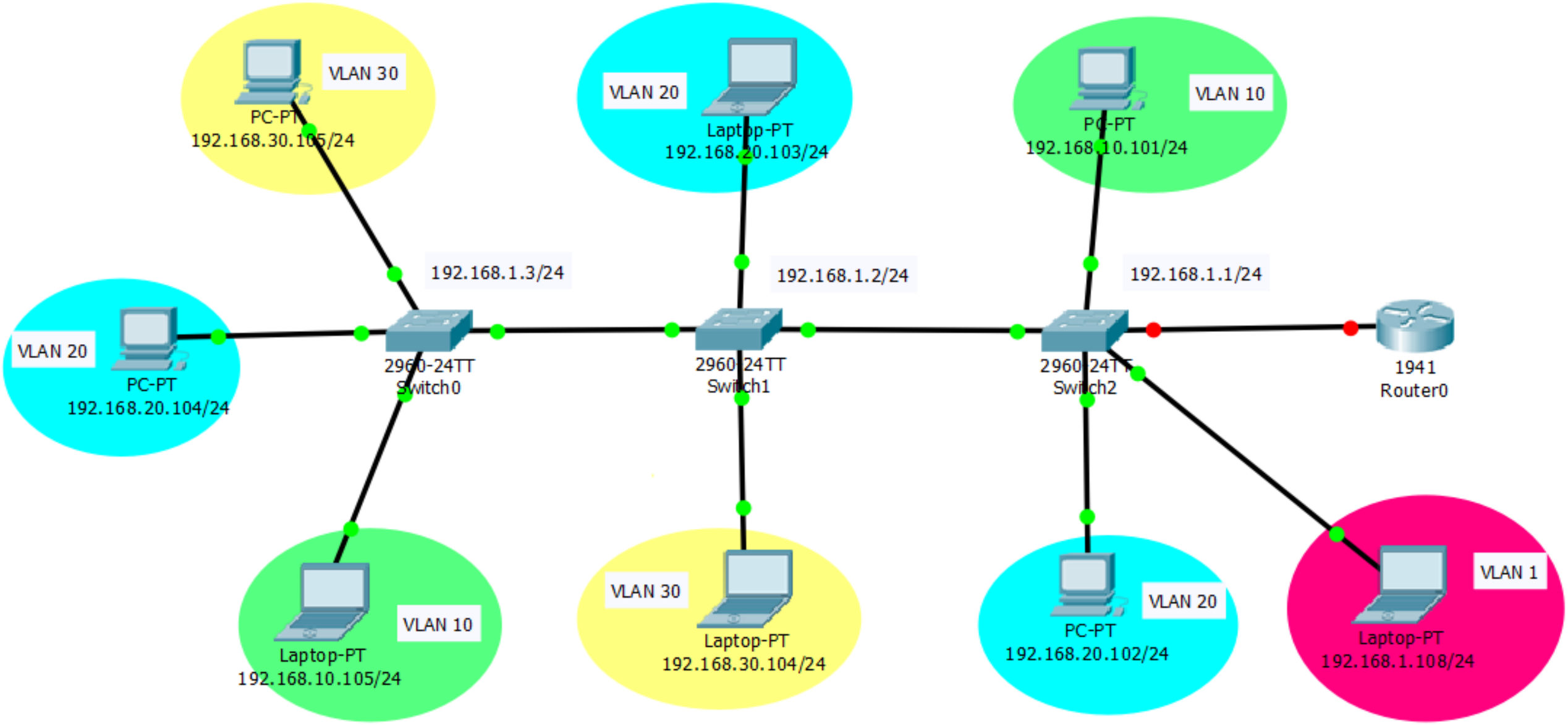
Перед администраторами linux и сетей очень часто возникает задача изоляции сегментов друг от друга. Их можно решить физическим разделением оборудования, но это не всегда приемлемо, т.к. количество портов ограничено, а стоимость оборудования не нулевая (и карт и свичей). На помощь приходят VLAN - они позволяют виртуализировать логические сети на одном оборудовании, как гипервизор позволяет запускать несколько виртуальных машин на одном компьютере. Существует несколько подходов к организации vlan на L2:

- 802.1q (Vlan-tagging)
- Port grouping

dot1q

Стандарт 802.1q приносит нам понятие режима порта:
- **native/access**
- **trunk/tagged**
- **mixed**
- **native vlan**

Обычно сервера подключаются в access-режиме и тогда все прозрачно для администратора, но это дает возможности пользоваться vlan’ами. Полностью возможности технологии раскрываются, когда порт переводится в trunk/tagged режим и появляется возможность получать несколько VLAN’ов по одному порту за счет добавления vlan-тега. Это требует дополнительной конфигурации, как со стороны оборудования, так и со стороны OS.


VLAN 30
PC-PT
192.168.30.105/24
VLAN 20
Laptop-PT
192.168.20.103/24
VLAN 10
PC-PT
192.168.10.101/24
192.168.1.3/24
192.168.1.2/24
192.168.1.1/24
VLAN 20
PC-PT
192.168.20.104/24
2960-24TT
Switch0
2960-24TT
Switch1
2960-24TT
Switch2
1941
Router0
VLAN 10
Laptop-PT
192.168.10.105/24
VLAN 30
Laptop-PT
192.168.30.104/24
VLAN 20
PC-PT
192.168.20.102/24
VLAN 1
Laptop-PT
192.168.1.108/24

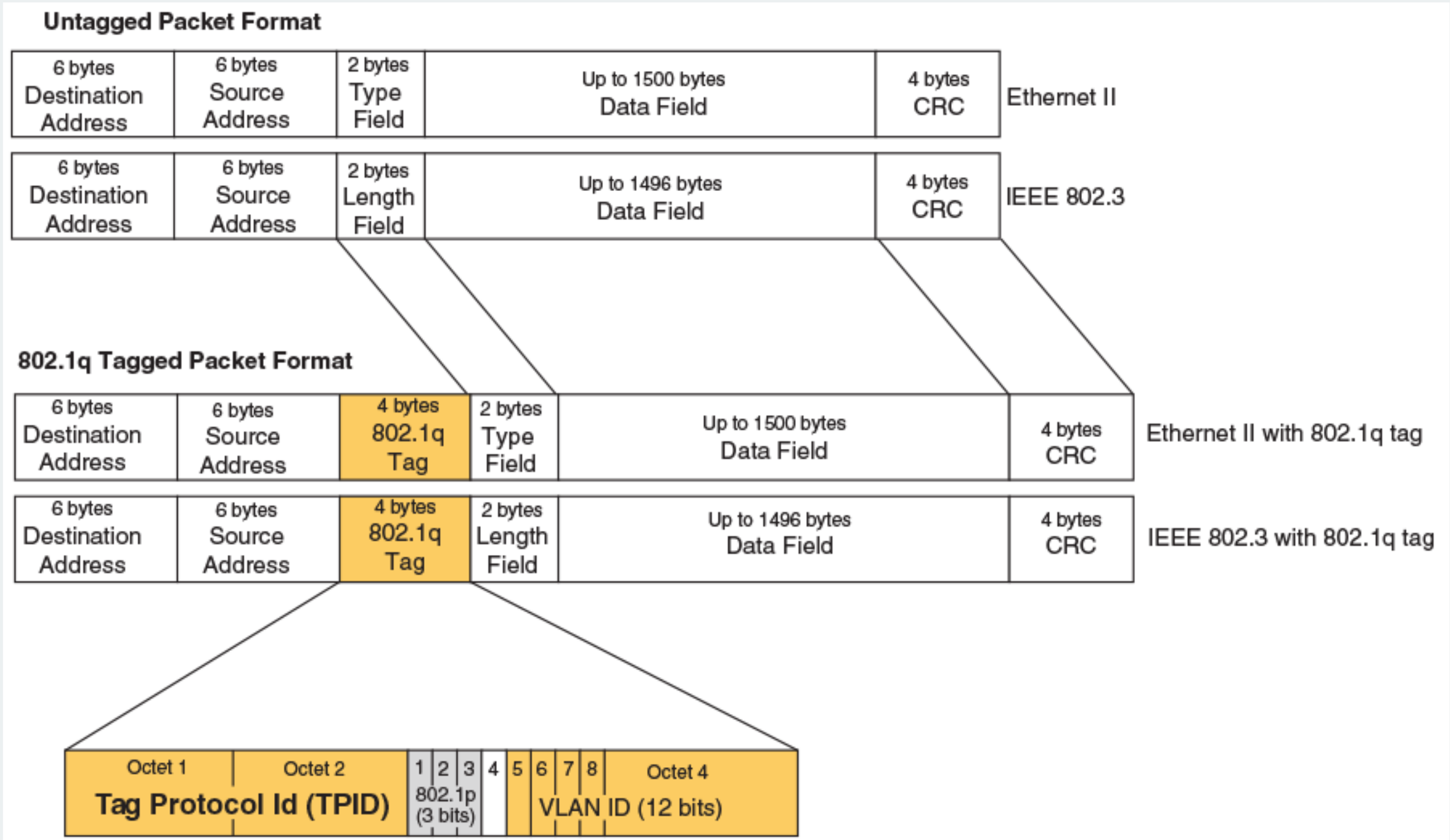
dot1q

Заголовок 802.1q размером 32 бита добавляется внутрь заголовка ethernet фрейма и увеличивает размер фрейма на 4 октета (32 бит), важно чтобы сетевое оборудование понимало это. Под идентификатор vlan'а выделено 12 бит (max 4096).

При посылке через vlan-интерфейс и передаче в родительский интерфейс к фрейму добавляется 802.1q заголовок и контрольная сумма фрейма пересчитывается. Это L2 информация, поэтому 802.1q это технология 2го уровня и не маршрутизируется.

При приеме фрейма на интерфейс система смотрит наличие у него 802.1q заголовка и, если нету, отправляет его на физический интерфейс (native vlan), а если есть - в соответствующий vlan-интерфейс.

VLAN для L3 называется IP-VPN и реализуется через технологию MPLS.


Untagged Packet Format
6 bytes Destination Address
6 bytes Source Address
2 bytes Type Field
Up to 1500 bytes Data Field
4 bytes CRC
Ethernet II
6 bytes Destination Address
6 bytes Source Address
2 bytes Length Field
Up to 1496 bytes Data Field
4 bytes CRC
IEEE 802.3
802.1q Tagged Packet Format
6 bytes Destination Address
6 bytes Source Address
4 bytes 802.1q Tag
2 bytes Type Field
Up to 1500 bytes Data Field
4 bytes CRC
Ethernet II with 802.1q tag
6 bytes Destination Address
6 bytes Source Address
4 bytes 802.1q Tag
2 bytes Length Field
Up to 1496 bytes Data Field
4 bytes CRC
IEEE 802.3 with 802.1q tag
Octet 1
Octet 2
Tag Protocol Id (TPID)
1 2 3
802.1p (3 bits)
4
5 6 7 8
Octet 4
VLAN ID (12 bits)

dot1q

Родительский интерфейс

```
[root@Net-R0 network-scripts]# cat ifcfg-enp0s10
ONBOOT=yes
BOOTPROTO=none
DEVICE=enp0s10
NM_CONTROLLED=no
```

Конфигурация vlan-интерфейса:

```
[root@Net-R0 network-scripts]# cat ifcfg-enp0s10.2
ONBOOT=yes
VLAN=yes
BOOTPROTO=static
TYPE=Ethernet
DEVICE=enp0s10.2
NM_CONTROLLED=no
IPADDR=172.16.2.1
PREFIX=24
```

macvlan

https://sreeninet.wordpress.com/2016/05/29/macvlan-and-ipvlan

Множество mac-адресов на интерфейсе, под каждый - свой дочерний интерфейс

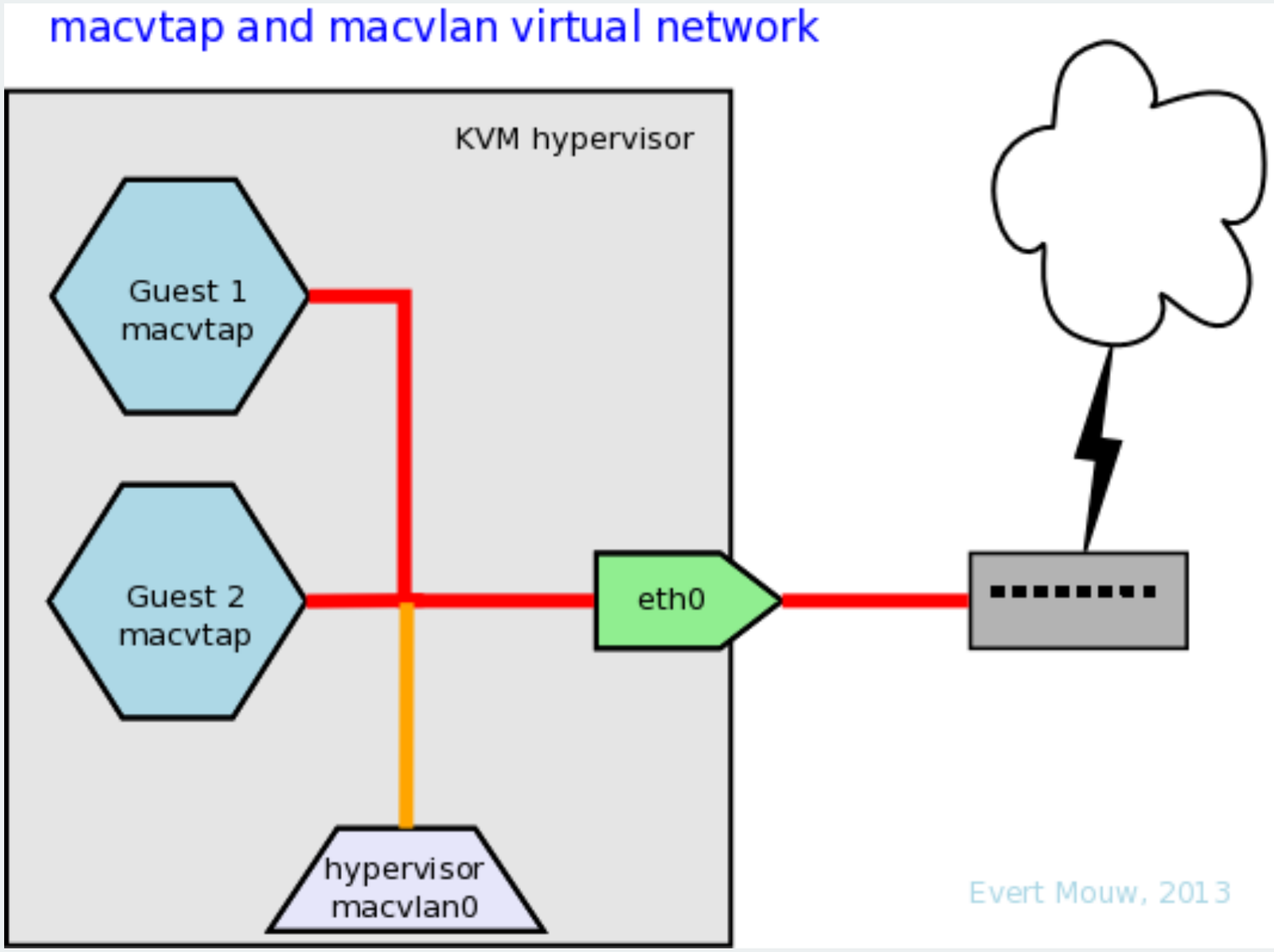
Конфигурация

```
ip link add mymacvlan1 link eth0 type macvlan mode bridge
ip link add mymacvlan2 link eth0 type macvlan mode bridge
ip link set mymacvlan1 up
ip link set mymacvlan1 up
```

RH и CentOS не поддерживают из коробки такую технологию, но есть возможность написать собственный хук поднятия интерфейса

```
/sbin/ifup-pre-local
/sbin/ifdown-pre-local
```


macvlan


macvtap and macvlan virtual network
KVM hypervisor
Guest 1
macvtap
Guest 2
macvtap
eth0
hypervisor
macvlan0
Evert Mouw, 2013

bonding

https://www.kernel.org/doc/Documentation/networking/bonding.txt

Объединение сетевых интерфейсов. Есть различные режимы:

#	name	FT	PERF	LB-OUT	LB-IN
0	balance-rr	✅	✅	✅	❌
1	active/backup	✅	❌	❌	❌
2	balance-xor	✅	✅	✅	❌
3	broadcast	✅	❌	❌	❌
4	802.3ad (LACP)	✅	✅	✅	✅
5	balance-tlb	✅	✅	✅	❌
6	balance-alb	✅	✅	✅	✅?

bonding

```
modprobe --first-time bonding
nmcli con show
nmcli con del eb9184a3-1cab-4b84-9bdf-3df397f25b2b
nmcli con del a2565d1e-773d-4251-833f-fa871e9d0c91
nmcli con add type bond con-name mybond0 ifname bond0 mode active-active
nmcli con mod mybond0 ipv4.addresses 192.168.1.149/24
nmcli con mod mybond0 ipv4.gateway 192.168.1.1
nmcli con mod mybond0 ipv4.method manual
nmcli con add type bond-slave con-name bond0-em1 ifname em1 master bond0
nmcli con add type bond-slave con-name bond0-em2 ifname em2 master bond0
nmcli con up mybond0
nmcli con show
```

bonding

```
# cat ifcfg-bond0
DEVICE=bond0
ONBOOT=yes
TYPE=Bond
BONDING_MASTER=yes
IPADDR=192.168.255.2
PREFIX=30
GATEWAY=192.168.255.1
BOOTPROTO=static
USERCTL=no
BONDING_OPTS="mode=1 miimon=100
fail_over_mac=1"
```

```
# cat ifcfg-enp0s9
DEVICE=enp0s9
ONBOOT=yes
SLAVE=yes
MASTER=bond0
BOOTPROTO=none
```

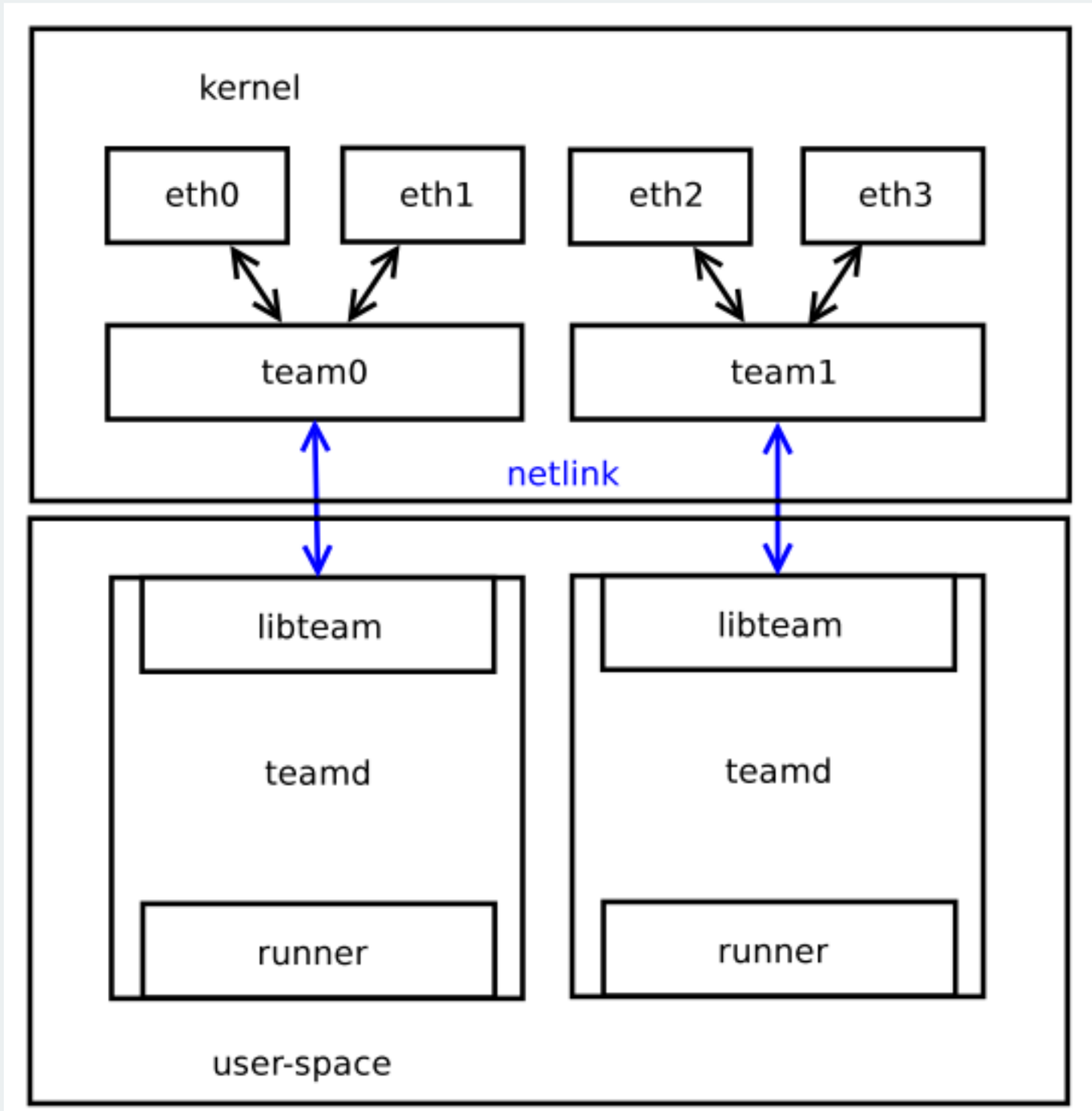
```
# cat ifcfg-enp0s10
DEVICE=enp0s10
ONBOOT=yes
SLAVE=yes
MASTER=bond0
BOOTPROTO=none
```

teaming

Альтернатива бондингу

утилиты и демон

- **teamd**
- **teamctl**
- **teamnl**



teaming

```
DEVICE=team0
DEVICETYPE=Team
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.11.1
PREFIX=24
TEAM_CONFIG='{"runner": {"name": "activebackup"}, "link_watch": {"name":
"ethtool"}}'
```

```
DEVICE=eth1
HWADDR=D4:85:64:01:46:9E
DEVICETYPE=TeamPort
ONBOOT=yes
TEAM_MASTER=team0
TEAM_PORT_CONFIG='{"prio": 100}'
```


Спасибо за внимание!

Вопросы?