

Блог IT-KB (<https://blog.it-kb.ru/>)

[Вики \(https://wiki.it-kb.ru\)](https://wiki.it-kb.ru)[Форум \(https://forum.it-kb.ru\)](https://forum.it-kb.ru)[О нас ▼](#)

Поиск

Настройка Network Bonding с LACP между CentOS Linux 7.2 и коммутатором Cisco 3560G

20.06.2016

4 795 Просмотров



(<https://blog.it-kb.ru/wp-content/uploads/2016/06/image-133.png>) На серверах имеющих несколько сетевых интерфейсов каждый отдельно взятый интерфейс можно использовать под какую-то выделенную задачу, например отдельный интерфейс под трафик управления хостом и отдельный интерфейс под трафик периодического резервного копирования. Может возникнуть мысль о том, что такая конфигурация имеет свои плюсы, например, позволяет максимально жёстко разграничить утилизацию интерфейсов под особенности разных задач. Но на этом плюсы, пожалуй, и заканчиваются. Ибо при таком подходе может получиться так, что один интерфейс будет постоянно чем-то чрезмерно нагружен, а другой интерфейс будет периодически полностью простаивать. К тому же, в такой схеме каждый из физических интерфейсов будет выступать как конкретная сущность, при выходе из строя которой будет происходить остановка того или иного сетевого сервиса, жёстко связанного с этой сущностью. С точки зрения повышения доступности и балансировки нагрузки, более эффективным методом использования нескольких сетевых интерфейсов сервера является объединение этих интерфейсов в логические сущности с использованием технологий **Network Bonding** и **Network Teaming**.

В этой заметке на конкретном примере мы рассмотрим то, как с помощью технологии **Network Bonding** в ОС **CentOS Linux 7.2** можно организовать объединение физических сетевых интерфейсов сервера в логический сетевой интерфейс (**bond**), а уже поверх него создать виртуальные сетевые интерфейсы под разные задачи и сетевые службы с изоляцией по разным **VLAN**. Агрегирование каналов между коммутатором и сервером будет выполняться с помощью протокола **Link Aggregation Control Protocol (LACP)** регламентированного документом **IEEE 802.3ad**. Использование LACP, с одной стороны, обеспечит высокую доступность агрегированного канала, так как выход из строя любого из линков между сервером и коммутатором не приведёт к потери доступности сетевых сервисов сервера, и с другой стороны, обеспечит равномерную балансировку нагрузки между физическими сетевыми интерфейсами сервера. Настройка в нашем примере будет выполняться как на стороне коммутатора (на примере коммутатора **Cisco Catalyst WS-C3560G**), так и на стороне Linux-сервера с двумя сетевыми интерфейсами (на примере сервера **HP ProLiant DL360 G5** с контроллерами **Broadcom NetXtreme II BCM5708/HP NC373i**)

Настройка конфигурации LACP на коммутаторе Cisco

Активация механизмов **LACP** на коммутаторе **Cisco** сводится к созданию виртуальной группы портов

channel-group и включению в эту группу нужных нам портов коммутатора.

Очищаем существующую конфигурацию портов коммутатора, которые будем включать в channel-group (в нашем примере это будут порты 21 и 22):

```
switch# configure terminal  
switch(config)# default interface range Gi0/21-22
```

Создаём channel-group и включаем в неё нужные нам порты коммутатора, предварительно эти отключив порты:

```
switch(config)# interface range Gi0/21-22  
switch(config-if-range)# shutdown  
switch(config-if-range)# channel-group 2 mode active  
Creating a port-channel interface Port-channel 2  
switch(config-if-range)# no shutdown  
switch(config-if-range)# exit
```

Вносим изменения в свойства появившегося виртуального интерфейса Port-channel2:

```
switch(config)# interface Port-channel2  
switch(config-if)# description LACP channel for KOM-AD01-VM32  
switch(config-if)# switchport trunk encapsulation dot1q  
switch(config-if)# switchport trunk allowed vlan 1,2,4  
switch(config-if)# switchport mode trunk  
switch(config-if)# exit
```

Изменения в свойствах первого порта-участника channel-group вносим минимальные, так как основные параметры наследуются портом с виртуального интерфейса channel-group:

```
switch(config)# interface GigabitEthernet0/21  
switch(config-if)# description LACP channel for KOM-AD01-VM32 (port enp3s0)  
switch(config-if)# channel-protocol lacp  
switch(config-if)# exit
```

Вносим изменения в свойства второго порта-участника channel-group:

```
switch(config)# interface GigabitEthernet0/22  
switch(config-if)# description LACP channel for KOM-AD01-VM32 (port enp5s0)  
switch(config-if)# channel-protocol lacp
```

Не забываем сохранить изменения:

```
switch(config-if)# end  
switch# write
```

Проверяем итоговую конфигурацию портов:

```
switch# show running-config

...
interface Port-channel2
  description LACP Channel for KOM-AD01-VM32
  switchport trunk encapsulation dot1q
  switchport trunk allowed vlan 1,2,4
  switchport mode trunk
...
interface GigabitEthernet0/21
  description LACP channel for KOM-AD01-VM32 (port enp3s0)
  switchport trunk encapsulation dot1q
  switchport trunk allowed vlan 1,2,4
  switchport mode trunk
  channel-protocol lacp
  channel-group 2 mode active
...
interface GigabitEthernet0/22
  description LACP channel for KOM-AD01-VM32 (port enp5s0)
  switchport trunk encapsulation dot1q
  switchport trunk allowed vlan 1,2,4
  switchport mode trunk
  channel-protocol lacp
  channel-group 2 mode active
...
```

Проверяем статус подключения портов:

```
switch# show interfaces status

...
Gi0/21  LACP channel for K  suspended  trunk  a-full a-1000 10/100/1000BaseTX
Gi0/22  LACP channel for K  suspended  trunk  a-full a-1000 10/100/1000BaseTX
...
Po2     LACP channel for K  notconnect  unassigned  auto  auto
...
```

До тех пор, пока на стороне нашего Linux-сервера не будет настроен LACP, статус портов будет отображаться как **suspended/notconnect**, а после настройки статус должен будет измениться на **connected** (это можно будет проверить позже):

```
switch# show interfaces status

...
Gi0/21  LACP channel for K  connected  trunk  a-full a-1000 10/100/1000BaseTX
Gi0/22  LACP channel for K  connected  trunk  a-full a-1000 10/100/1000BaseTX
...
Po2     LACP channel for K  connected  trunk  a-full a-1000
...
```

Проверяем внутренний статус LACP:

```
switch# show lacp internal

Flags:  S - Device is requesting Slow LACPDUs
        F - Device is requesting Fast LACPDUs
        A - Device is in Active mode           P - Device is in Passive mode
...
Channel group 2

Port      Flags  State  LACP port  Admin  Oper  Port  Port
Gi0/21    SA     susp   32768      0x2    0x2    0x15  0x7D
Gi0/22    SA     susp   32768      0x2    0x2    0x16  0x7D
...
```

Здесь аналогично, до тех пор, пока на стороне нашего Linux-сервера не будет настроен LACP, статус LACP будет соответствующий, а после настройки статус должен будет измениться:

```
switch# show lacp internal

...
Channel group 2

Port      Flags  State  LACP port  Admin  Oper  Port  Port
Gi0/21    SA     bndl   32768      0x2    0x2    0x15  0x3D
Gi0/22    SA     bndl   32768      0x2    0x2    0x16  0x3D
...
```

Теперь можем перейти к настройке **Network Bonding с LACP** на стороне Linux-сервера.

Настройка Network Bonding с LACP в CentOS Linux 7.2

Проверим наличие модуля поддержки **bonding** (команда должна отработать без ошибок):

```
# modinfo bonding
```

В ниже представленном примере мы рассмотрим сценарий настройки логического **bond**-интерфейса **bond0** (**master**-интерфейс), членами которого будут два имеющихся в сервере физических интерфейса **enp3s0** и **enps5s0** (**slave**-интерфейсы). А затем на созданном **bond**-интерфейсе мы создадим дочерний **VLAN**-интерфейс, имеющий доступ к изолированной тегированной сети с определённым номером **VLAN**. Всю настройку мы будем выполнять путём прямой правки файлов интерфейсов.

Создадим файл с конфигурацией нового агрегирующего интерфейса **bond0**:

```
# nano /etc/sysconfig/network-scripts/ifcfg-bond0
```

Наполним файл параметрами бонда bond0:

```
DEVICE=bond0
NAME=bond0
TYPE=Bond
BONDING_MASTER=yes
IPV6INIT=no
MTU=9000
ONBOOT=yes
USERCTL=no
NM_CONTROLLED=no
BOOTPROTO=none
BONDING_OPTS="mode=802.3ad xmit_hash_policy=layer2+3 lacp_rate=1 miimon=100"
```

Опции **BONDING_OPTS** можно передавать и через файл /etc/modprobe.d/bonding.conf, но вместо этого лучше использовать файлы ifcfg-bond*. Исключением здесь является лишь глобальный параметр **max_bonds**. Это замечание описано в документе Create a Channel Bonding Interface (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/sec-Network_Bonding_Using_the_Command_Line_Interface.html#sec-Create_a_Channel_Bonding_Interface).

Получить информацию о всех возможных опциях **BONDING_OPTS** и вариантах их значений можно в документе Using Channel Bonding (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/sec-Using_Channel_Bonding.html).

В нашем примере используется несколько опций:

- **mode=802.3ad**, который определяет режим работы bond-a (bonding policy). В нашем случае выбран режим с использованием протокола LACP. Этот режим для корректной работы должен поддерживаться и коммутатором, к которому подключаются сетевые интерфейсы bond-a.
- **lacp_rate=fast**, который заставляет bonding-интерфейс отсылать пакеты LACPDU каждую секунду, в то время как значение по умолчанию slow, которое определяет 30-секундный интервал.
- **xmit_hash_policy=layer2+3**, который определяет режим вычисления хешей при организации балансировки нагрузки между интерфейсами бонда. Эта опция используется только для режимов работы бонда (ранее описанный mode) поддерживающих балансировку нагрузки, таких как balance-xor и 802.3ad. Значение layer2+3 говорит о том, что для вычисления хешей будут использоваться как MAC адреса получателей/отправителей пакета, так и их IP адреса, если это возможно. Значение по умолчанию для этой опции – layer2, что определяет вычисление хеша только по MAC-адресам.
- **miimon=100**, который определяет количество миллисекунд для мониторинга состояния линка с помощью механизмов MII, который, к слову говоря, должен поддерживаться физическим сетевым контроллером. Значение опции miimon по умолчанию – 0, что значит, что данный функционал не используется.

Чтобы проверить то, может ли наш сетевой контроллер работать с MII, можно выполнить:

```
# ethtool enp3s0 | grep "Link detected:"
```

```
Link detected: yes
```

Значение **yes** в данном случае говорит о поддержке MII.

Если вы используете режим **active-backup** bonding, то есть рекомендация (https://access.redhat.com/node/28421/Configuring_VLAN_devices_over_a_bonded_interface) настраивать дополнительную опцию bond-интерфейса: **fail_over_mac=1**. Как я понял, это нужно для того, чтобы использовать для bond в качестве основного MAC-адреса, MAC-адрес backup-интерфейса, что позволит сократить время потери соединений в процессе fail-over.

Создадим файл с конфигурацией **slave**-интерфейсов, то есть физических Ethernet-портов, которые будут участниками bond0 (файл скорее всего уже существует, так что просто отредактируем его под свою задачу):

```
# nano /etc/sysconfig/network-scripts/ifcfg-enp3s0
```

Содержимое файла

```
DEVICE=enp3s0
NAME=bond0-slave0
TYPE=Ethernet
MTU=9000
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
USERCTL=no
NM_CONTROLLED=no
```

Второй **slave**-интерфейс настраиваем по аналогии:

```
# nano /etc/sysconfig/network-scripts/ifcfg-enp5s0
```

Параметры файла аналогичные за исключением параметров **DEVICE** и **NAME**

```
DEVICE=enp5s0
NAME=bond0-slave1
TYPE=Ethernet
MTU=9000
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
USERCTL=no
NM_CONTROLLED=no
```

Если используется служба **NetworkManager**, то перезагрузим её конфигурацию с учётом созданных/изменённых файлов ifcfg-*:

```
# nmcli con reload
```

Перезапускаем **slave**-интерфейсы:

```
# ifdown enp3s0 && ifup enp3s0
# ifdown enp5s0 && ifup enp5s0
```

Если отдельное отключение/включение интерфейсов приводит к ошибкам (это возможно в случае если ранее подключения контролировались службой NetworkManager), то перезапускаем сеть полностью:

```
# service network restart
```

Убедимся в том, что созданные интерфейсы успешно запущены со статусом **UP** и заданными нами настройками, например обратим внимание на размер MTU.

```
# ip link show
...
2: enp3s0: mtu 9000 qdisc mq master bond0 state UP mode DEFAULT qlen 1000
   link/ether 00:1f:28:0a:f7:f4 brd ff:ff:ff:ff:ff:ff
3: enp5s0: mtu 9000 qdisc mq master bond0 state UP mode DEFAULT qlen 1000
   link/ether 00:1f:28:0a:f7:f4 brd ff:ff:ff:ff:ff:ff
4: bond0: mtu 9000 qdisc noqueue state UP mode DEFAULT
   link/ether 00:1f:28:0a:f7:f4 brd ff:ff:ff:ff:ff:ff
```

Итак, bond создан и запущен в работу, и теперь переходим к созданию виртуальных интерфейсов с поддержкой VLAN поверх нашего bond-a.

Проверим наличие модуля поддержки **VLAN** (команда должна отработать без ошибок)

```
# modprobe 8021q
```

Под каждый отдельный VLAN-интерфейс создаём конфигурационные файлы в каталоге `/etc/sysconfig/network-scripts` по типу `ifcfg-bond<номер бонда>.<номер VLAN-a>`. Например создадим конфигурационный файл для bond-интерфейса `bond0` с **VLAN 2**

```
# nano /etc/sysconfig/network-scripts/ifcfg-bond0.2
```

Содержимое файла:

```
DEVICE=bond0.2
NAME=bond0-vlan2
BOOTPROTO=none
ONBOOT=yes
USERCTL=no
NM_CONTROLLED=no
VLAN=yes
MTU=1500
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
IPADDR=10.1.0.32
PREFIX=24
GATEWAY=10.1.0.1
DNS1=10.1.0.9
DNS2=10.1.1.8
DOMAIN=holding.com
```

Попробуем поднять созданный VLAN-интерфейс или опять-же просто перезапускаем сеть:

```
# ifup ifcfg-bond0.2
# service network restart
```

Чтобы разнообразные сетевые службы, используемые на нашем сервере приняли во внимание изменение сетевой конфигурации, можно перезапустить их, либо просто перезагрузить сервер.

Убедимся в том, что VLAN-интерфейс поднялся с указанными нами настройками:

```
# ip link show
...

5: bond0.2@bond0: mtu 1500 qdisc noqueue state UP mode DEFAULT
    link/ether 00:1f:28:0a:f7:f4 brd ff:ff:ff:ff:ff:ff
```

Аналогичным образом мы можем создать на сервере столько выделенных виртуальных VLAN интерфейсов, сколько потребуется для разделения трафика между разными задачами и сетевыми службами исполняемыми на сервере.

Проверяем статус работы `bond0`:


```
# cat /proc/net/bonding/bond0
```

Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation

Transmit Hash Policy: layer2+3 (2)

MII Status: up

MII Polling Interval (ms): 100

Up Delay (ms): 0

Down Delay (ms): 0

802.3ad info

LACP rate: fast

Min links: 0

Aggregator selection policy (ad_select): stable

Active Aggregator Info:

Aggregator ID: 1

Number of ports: 2

Actor Key: 9

Partner Key: 2

Partner Mac Address: 00:1c:f8:57:bd:50

Slave Interface: enp3s0

MII Status: up

Speed: 1000 Mbps

Duplex: full

Link Failure Count: 0

Permanent HW addr: 00:1f:28:0a:f7:f4

Slave queue ID: 0

Aggregator ID: 1

Actor Churn State: none

Partner Churn State: none

Actor Churned Count: 0

Partner Churned Count: 0

details actor lacp pdu:

system priority: 65535

port key: 9

port priority: 255

port number: 1

port state: 63

details partner lacp pdu:

system priority: 32768

oper key: 2

port priority: 32768

port number: 21

port state: 61

Slave Interface: enp5s0

MII Status: up

Speed: 1000 Mbps

Duplex: full

Link Failure Count: 0

Permanent HW addr: 00:1f:29:0a:f7:f8

```
Slave queue ID: 0
Aggregator ID: 1
Actor Churn State: none
Partner Churn State: none
Actor Churned Count: 0
Partner Churned Count: 0
details actor lacp pdu:
  system priority: 65535
  port key: 9
  port priority: 255
  port number: 2
  port state: 63
details partner lacp pdu:
  system priority: 32768
  oper key: 2
  port priority: 32768
  port number: 22
  port state: 61
```

Если в секции **802.3ad info** значение параметра **Partner Mac Address** не содержит MAC-адреса коммутатора, а вместо этого видно значение 00:00:00:00:00:00, это значит то, что обмен пакетами LACPDU с коммутатором по какой-то причине не выполняется (либо на стороне коммутатора неправильно настроен LACP, либо коммутатор вовсе его не поддерживает)

На этом этапе агрегированный LACP линк между коммутатором и сервером можно считать настроенным. На стороне коммутатора убедимся в том, что вывод ранее упомянутых команд изменился:

```
switch# show interfaces status
switch# show lacp internal
```

Теперь можно переходить к заключительному этапу – проверкам нашего LACP соединения на предмет устойчивости соединения при отказе любого из портов на стороне коммутатора/сервера а также на предмет корректности балансировки нагрузки.

Проверка высокой доступности соединения

Теперь попробуем проверить устойчивость нашего LACP-соединения. Для этого запустим с любой сторонней системы, например с рабочей станции администратора, **ping** IP-адреса **bond**-интерфейса сервера

```
ping -t 10.1.0.32
```

Подключимся к коммутатору Cisco и выполним отключение одного из портов, входящих в группу channel-group, обеспечивающей со стороны коммутатора LACP-соединение.

```
switch# configure terminal
switch(config)# int Gi0/21
switch(config-if)# shutdown
```

Убедимся в том, что запущенная утилита `ping` не отражает факт потери пакетов. Если всё в порядке, включим отключенный порт коммутатора, а затем отключим второй порт входящий в группу `channel-group`:

```
switch(config-if)# no shutdown
switch(config-if)# exit
switch(config)# int Gi0/22
switch(config-if)# shutdown
switch(config-if)# no shutdown
switch(config-if)# end
```

Снова проверим, что `ping` не имеет потерь пакетов. Если всё в порядке, включим отключенный порт коммутатора и будем считать, что наше LACP-соединение Cisco `channel-group` <-> Linux `bond` успешно отрабатывает ситуацию потери соединения на любом из интерфейсов коммутатора, входящих в `channel-group`. Аналогичную проверку можно выполнить и со стороны Linux сервера попеременно отключая интерфейсы входящие в `bond`:

```
# ifdown enp3s0
# ifup enp3s0
# ifdown enp5s0
# ifup enp5s0
```

Здесь между включением первого интерфейса и отключением второго для чистоты эксперимента нужно сделать небольшую паузу (5-7 секунд), чтобы LACP-линк смог полностью перестроиться для того, чтобы не было потерь пакетов. Если же эту паузу не сделать, то несколько пакетов может таки потеряться, но потом соединение всё-же будет автоматически восстановлено.

Проверка балансировки нагрузки соединения

Чтобы проверить факт того, что трафик действительно распределяется по интерфейсам `bond`-а, воспользуемся немного более сложной процедурой тестирования, для которой используем три компьютера, то есть сам наш сервер КОМ-AD01-VM32 (IP: 10.1.0.32) и любые два Linux-компьютера, например КОМ-AD01-SRV01 (IP: 10.1.0.11) и КОМ-AD01-SRV02 (IP: 10.1.0.12). Установим на все три компьютера две утилиты - **iperf** (для тестирования скорости) и **nload** (для отображения нагрузки интерфейсов в реальном времени).

Начнем с проверки **входящего трафика**.

Установим на проверяемом сервере КОМ-AD01-VM32 утилиты **iperf** и **nload** а затем на время теста выключим **брандмауэр** (либо можете создать разрешающее правило для порта, который будет

слушать запущенный серверный **iperf**, по умолчанию это **TCP 5001**). Запустим утилиту **iperf** в режиме сервера (с ключом **-s**), то есть ориентированную на приём трафика:

```
# yum -y install iperf nload
# service firewalld stop
# iperf -s

-----
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
-----
```

Здесь же, но в отдельной консоли, запустим утилиту **nload**, которая в реальном времени будет показывать нам статистические значения нагрузки на интерфейсы. В качестве параметров укажем названия интерфейсов, за нагрузкой которых нужно следить:

```
# nload enp3s0 enp5s0
```

Затем выполняем отсылку трафика с помощью **iperf** (в режиме клиента) с компьютеров КОМ-AD01-SRV01 и КОМ-AD01-SRV02 (запустим одинаковую команду одновременно на обоих этих компьютерах):

```
# iperf -c 10.1.0.32 -t 600 -i 2 -b 100m
```

В качестве параметров **iperf** укажем следующие значения:

- **-c** – работа в режиме клиента (отсылка трафика)
- 10.1.0.32 - IP адрес сервера, на который клиент **iperf** будет посылать трафик (КОМ-AD01-VM32)
- **-t** 600 – время, в течение которого будет выполняться отсылка трафика (600 секунд)
- **-i** 2 - интервал в секундах для вывода статистической информации об отсылке трафика на экран (каждые 2 секунды)
- **-b** 100m – размер пропускной способности сети, которую будет пытаться задействовать **iperf** при отсылке трафика (в нашем случае 100 мегабит/с)

После того, как **iperf** в режиме клиента начнёт работу на обоих серверах, переходим на проверяемый сервер КОМ-AD01-VM32 и наблюдаем за статистикой **nload**. На начальном экране увидим статистику по входящему и исходящему трафику по первому интерфейсу (**enp3s0**). Обращаем внимание на значения по входящему трафику:

```
Device enp3s0 (1/2):
```

```
=====
Incoming:
##### Curr: 98.64 MBit/s
##### Avg: 99.14 MBit/s
##### Min: 0.00 Bit/s
##### Max: 197.12 MBit/s
##### Ttl: 120.64 GByte
...
```

Чтобы увидеть статистику по следующему интерфейсу (enp5s0) просто нажмём **Enter**:

```
Device enp5s0 (2/2):
```

```
=====
Incoming:
##### Curr: 97.93 MBit/s
##### Avg: 97.97 MBit/s
##### Min: 0.00 Bit/s
##### Max: 196.68 MBit/s
##### Ttl: 40.56 GByte
...
```

Как видим, скорость на обоих интерфейсах распределена равномерно и равна примерно 100 мегабит/с, значит балансировка входящего трафика работает успешно.

Далее, по аналогичной методике, протестируем механизм балансировки нагрузки для **трафика исходящего** с нашего подопытного сервера KOM-AD01-VM32. Для этого **iperf** будет запускаться на компьютерах KOM-AD01-SRV01/KOM-AD01-SRV02 в режиме сервера, принимающего трафик. Запустим одинаковую команду на обоих этих компьютерах, не забывая при этом про необходимость открытия порта для сервера **iperf**:

```
# iperf -s
```

А на компьютере KOM-AD01-VM32 утилиту **iperf** запускаем в режиме клиента, отправляющего трафик на компьютеры KOM-AD01-SRV01/KOM-AD01-SRV02, используя при этом запуск из двух разных консолей:

```
# iperf -c 10.1.0.11 -t 600 -i 2 -b 100m
# iperf -c 10.1.0.12 -t 600 -i 2 -b 100m
```

Запускаем на компьютере KOM-AD01-VM32 третью консоль и наблюдаем за ситуацией в **nload**

```
# nload enp3s0 enp5s0
```

На этот раз обращаем внимание на исходящий трафик на первом интерфейсе...

```
Device enp3s0 (1/2):
```

```
=====
```

```
...
```

```
Outgoing:
```

```
##### Curr: 102.88 MBit/s
##### Avg: 51.36 MBit/s
##### Min: 0.00 Bit/s
##### Max: 111.24 MBit/s
##### Ttl: 1.53 GByte
```

...и исходящий трафик на втором интерфейсе...

```
Device enp5s0 (2/2):
```

```
=====
```

```
...
```

```
Outgoing:
```

```
##### Curr: 101.16 MBit/s
##### Avg: 59.73 MBit/s
##### Min: 5.14 kBit/s
##### Max: 110.71 MBit/s
##### Ttl: 1.89 GByte
```

Как видим, трафик распределён по разным интерфейсам нашего bond-a, значит балансировка исходящего трафика работает успешно.

Если ранее на время теста отключался брандмауэр, то по окончании теста не забываем включить его обратно:

```
# service firewalld start
```


На этом всё.

Дополнительные источники информации:

- Red Hat Enterprise Linux 7 Networking Guide - Chapter 4. Configure Network Bonding (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/ch-Configure_Network_Bonding.html)
- RedHat Knowledgebase - How to use bonded interfaces in specific applications? [доступ ограничен] (https://access.redhat.com/node/28421/Configuring_VLAN_devices_over_a_bonded_interface)
- kb.h1host.ru — Настройка объединения портов (bonding) Cisco IOS и CentOS LACP (<http://kb.h1host.ru/article/view?id=111>)
- Irek Fasihov — Пример настройки сети в Linux CentOS 6.5 связкой Bond+VLAN's (<http://www.theirek.com/blog/2014/03/22/nastroika-sieti-v-linux-centos-sviazkoi-bond-plus-vlans>)

Поделиться ссылкой на эту запись:

👍 Pocket (<https://blog.it-kb.ru/2016/06/20/network-bonding-with-vlan-and-802-3ad-lacp-on-centos-linux-7-2-and-lag-channel-group-on-switch-cisco-catalyst-ws-c3560q-with-testing-load-balancing-and-high-availability/?share=pocket&nb=1>)

 Pinterest (<https://blog.it-kb.ru/2016/06/20/network-bonding-with-vlan-and-802-3ad-lacp-on-centos-linux-7-2-and-lag-channel-group-on-switch-cisco-catalyst-ws-c3560g-with-testing-load-balancing-and-high-availability/?share=pinterest&nb=1>)

Отделяем трафик
резервного копирования
System Center 2012 DPM
(<https://blog.it-kb.ru/2013/05/25/system-center-2012-dpm-network-on-separate-nic/>)
25.05.2013

В "Microsoft System Center"

CentOS Linux 7.2 и HP Array
Config Utility (ACU) для
управления устаревшими
контроллерами HP Smart
Array (<https://blog.it-kb.ru/2016/08/26/install-hp-array-config-utility-acu-on-centos-linux-7-2-for-hp-smart-array-controllers-as-6400-management/>)

26.08.2016

В "CentOS"

Настройка CentOS Linux 7.2
на сервере HP ProLiant
DL360 G5. Установка HP
System Management Tools
(<https://blog.it-kb.ru/2016/06/07/centos-linux-7-2-on-hp-proliant-dl360-g5-install-hp-system-management-tools-add-repo-sh-script-snmp-hp-health-system-management-homepage-smh-hpssa-vca/>)

07.06.2016

В "CentOS"

Опубликовано в : CentOS (<https://blog.it-kb.ru/category/centos/>) , Cisco (<https://blog.it-kb.ru/category/cisco/>) , Linux (<https://blog.it-kb.ru/category/linux/>) , Networking (<https://blog.it-kb.ru/category/networking/>) , Red Hat Enterprise Linux Server (<https://blog.it-kb.ru/category/red-hat-enterprise-linux-server/>) , Virtualization (<https://blog.it-kb.ru/category/virtualization/>)

Метки : 802.1q (<https://blog.it-kb.ru/tag/802-1q/>) , 802.3ad (<https://blog.it-kb.ru/tag/802-3ad/>) , 8021q (<https://blog.it-kb.ru/tag/8021q/>) , 8023ad (<https://blog.it-kb.ru/tag/8023ad/>) , BCM5708 (<https://blog.it-kb.ru/tag/bcm5708/>) , Bond (<https://blog.it-kb.ru/tag/bond/>) , Bonding (<https://blog.it-kb.ru/tag/bonding/>) , Broadcom (<https://blog.it-kb.ru/tag/broadcom/>) , C3560 (<https://blog.it-kb.ru/tag/c3560/>) , Catalyst (<https://blog.it-kb.ru/tag/catalyst/>) , CentOS (<https://blog.it-kb.ru/tag/centos/>) , CentOS 7 (<https://blog.it-kb.ru/tag/centos-7/>) , Channel-Group (<https://blog.it-kb.ru/tag/channel-group/>) , Cisco (<https://blog.it-kb.ru/tag/cisco/>) , DL360 (<https://blog.it-kb.ru/tag/dl360/>) , dot1q (<https://blog.it-kb.ru/tag/dot1q/>) , ethtool (<https://blog.it-kb.ru/tag/ethtool/>) , G5 (<https://blog.it-kb.ru/tag/g5/>) , High Availability (<https://blog.it-kb.ru/tag/high-availability/>) , HP (<https://blog.it-kb.ru/tag/hp/>) , ifcfg (<https://blog.it-kb.ru/tag/ifcfg/>) , IOS (<https://blog.it-kb.ru/tag/ios/>) , iperf (<https://blog.it-kb.ru/tag/iperf/>) , LACP (<https://blog.it-kb.ru/tag/lacp/>) , LAG (<https://blog.it-kb.ru/tag/lag/>) , Linux (<https://blog.it-kb.ru/tag/linux/>) , Load Balancing (<https://blog.it-kb.ru/tag/load-balancing/>) , MTU (<https://blog.it-kb.ru/tag/mtu/>) , NC373i (<https://blog.it-kb.ru/tag/nc373i/>) , Networking (<https://blog.it-kb.ru/tag/networking/>) , NetworkManager (<https://blog.it-kb.ru/tag/networkmanager/>) , NetXtreme (<https://blog.it-kb.ru/tag/netxtreme/>) , NIC Bonding (<https://blog.it-kb.ru/tag/nic-bonding/>) , nload (<https://blog.it-kb.ru/tag/nload/>) , nmcli (<https://blog.it-kb.ru/tag/nmcli/>) , Port-Channel (<https://blog.it-kb.ru/tag/port-channel/>) , ProLiant (<https://blog.it-kb.ru/tag/proliant/>) , RHEL (<https://blog.it-kb.ru/tag/rhel/>) , RHEL 7 (<https://blog.it-kb.ru/tag/rhel-7/>) , Trunk (<https://blog.it-kb.ru/tag/trunk/>) , Virtualization (<https://blog.it-kb.ru/tag/virtualization/>) , VLAN (<https://blog.it-kb.ru/tag/vlan/>)

Всего комментариев: 5

Комментировать



Сергей / 20.06.2016 16:15

Почему во всех инструкциях по данному вопросу, в настройках сетевых интерфейсов в CentOS выставляют такой MTU аж 9000 с какой целью ? Объясните пожалуйста. Спасибо



Алексей Максимов (<https://blog.it-kb.ru/author/blogroot/>) / Автор записи 20.06.2016 16:27

Потому что bond-интерфейс по сути является корневым для виртуальных интерфейсов, которые будут далее создаваться на его основе. Среди этих интерфейсов более верхнего уровня нередко используются интерфейсы под такие задачи как, например, периодическое полное резервное копирование или, например, миграция виртуальных машин. Так вот для таких интерфейсов принято использовать MTU максимально возможного размера. Поэтому на корневом bond-интерфейсе, ровно как и на его физических slave-интерфейсах, выставляется увеличенный размер MTU, чтобы его можно было потом использовать на виртуальных интерфейсах более высокого уровня. А для интерфейсов верхнего уровня не требующих большого MTU (например интерфейс управления хостом) выставляется более скромное значение MTU, так как там в большинстве случаев поддержка Jumbo Frame не требуется.

Ответить (<https://blog.it-kb.ru/2016/06/20/network-bonding-with-vlan-and-802-3ad-lacp-on-centos-linux-7-2-and-lag-channel-group-on-switch-cisco-catalyst-ws-c3560g-with-testing-load-balancing-and-high-availability/?replytocom=26375#respond>)



Сергей / 20.06.2016 16:28

Спасибо.

Ответить (<https://blog.it-kb.ru/2016/06/20/network-bonding-with-vlan-and-802-3ad-lacp-on-centos-linux-7-2-and-lag-channel-group-on-switch-cisco-catalyst-ws-c3560g-with-testing-load-balancing-and-high-availability/?replytocom=26376#respond>)



Юрий Быков (<http://tplus.net>) / 20.06.2016 17:09

В вики кратко говорится почему обычно именно 9000 - <https://ru.wikipedia.org/wiki/Jumbo-%D0%BA%D0%B0%D0%B4%D1%80> (<https://ru.wikipedia.org/wiki/Jumbo-%D0%BA%D0%B0%D0%B4%D1%80>)

Ответить (<https://blog.it-kb.ru/2016/06/20/network-bonding-with-vlan-and-802-3ad-lacp-on-centos-linux-7-2-and-lag-channel-group-on-switch-cisco-catalyst-ws-c3560g-with-testing-load-balancing-and-high-availability/?replytocom=26377#respond>)

Обратная ссылка: Развёртывание и настройка oVirt 4.0. Часть 4. Fencing как средство повышения доступности хостов и виртуальных машин | Блог IT-KB (<https://blog.it-kb.ru/2016/09/16/install-ovirt-4-0-part-4-about-ssh-soft-fencing-and-hard-fencing-over-hp-proliant-ilo2-power-managment-agent-and-test-of-high-availability/>) / 16.09.2016 14:57

Добавить комментарий

Введите свой комментарий...

**Социальные
ссылки**

Email: Blog@IT-KB.RU
(<mailto:Blog@IT-KB.RU>)

**Купи свой
домен!**

Алексей Максимов

Продвинуть

([http://www.liveintern
et.ru/stat/blog.it-
kb.ru](http://www.liveinternet.ru/stat/blog.it-kb.ru))

(<http://blog.it-kb.ru/feed/>)

(https://twitter.com/Blog_IT_KB)

(<https://www.facebook.com/blog.it.kb>)

(<https://plus.google.com/115134238896229141625>)

(<http://blog-it-kb.tumblr.com/>)

(<http://vk.com/blogitkb>)

© Блог IT-KB (<https://blog.it-kb.ru/>) All Rights Reserved. Theme zAlive by zenoven (<http://www.zenoven.com/>).

Контактная форма (<https://blog.it-kb.ru/cform/>) Команда авторов (<https://blog.it-kb.ru/about-this-blog/>)