→ Convenciones:

```
# En todos los nodos como 'sudo su'.
[root@srv1 ~]# Solo en servidor 'srv1'→ como 'sudo su'.
[root@srv2 ~]# Solo en servidor 'srv2'→ como 'sudo su'.
```

364.4 Network High Availability (weight: 5)

Weight	5
Description	Candidates should be able to configure redundant networking connections and manage VLANs. Furthermore, candidates should have a basic understanding of BGP.

Key Knowledge Areas:

- Understand and configure bonding network interface
- Network bond modes and algorithms (active-backup, blance-tlb, balance-alb, 802.3ad, balance-rr, balance-xor, broadcast)
- Configure switch configuration for high availability, including RSTP
- Configure VLANs on regular and bonded network interfaces
- Persist bonding and VLAN configuration
- Understand the principle of autonomous systems and BGP to manage external redundant uplinks
- Awareness of traffic shaping and control capabilities of Linux

Partial list of the used files, terms and utilities:

- bonding.ko (including relevant module options)
- /etc/network/interfaces
- /etc/sysconfig/networking-scripts/ifcfg-*
- /etc/systemd/network/*.network
- /etc/systemd/network/*.netdev
- nmcli
- /sys/class/net/bonding_masters
- /sys/class/net/bond*/bonding/miimon
- /sys/class/net/bond*/bonding/slaves
- ifenslave
- ip

bond-00 ~ # modprobe bonding

bond-00 ~ # lsmod | grepe bonding

bond-00 ~ # nmcli c s

NAME UUID TYPE DEVICE

enp1s0 4865a23e-32aa-49c4-94cf-2a1f4a7ef3ac ethernet enp1s0

```
enp6s0  1f43b3c1-4e6b-3663-92de-061376dffcdf  ethernet  --
enp7s0  0633a275-8c73-35b8-9c23-93027c361447  ethernet  --
bond-00 ~ # ip -brief addr
lo               UNKNOWN        127.0.0.1/8 ::1/128
enp1s0           UP             192.168.10.160/24 fe80::d9a5:536c:4430:70c5/64
enp6s0           UP
enp7s0           UP
bond-00 ~ # modprobe --first-time bonding
bond-00 ~ # modinfo bonding
filename:       /lib/modules/4.18.0-240.15.1.el8_3.x86_64/kernel/drivers/net/bonding/bonding.ko.xz
author:         Thomas Davis, tadavis@lbl.gov and many others
description:    Ethernet Channel Bonding Driver, v3.7.1
version:        3.7.1
license:        GPL
alias:          rtnl-link-bond
rhelversion:    8.3
srcversion:     CB4A152A6BB7543FE26B59A
depends:
intree:         Y
name:           bonding
vermagic:       4.18.0-240.15.1.el8_3.x86_64 SMP mod_unload modversions
sig_id:         PKCS#7
signer:         Red Hat Enterprise Linux kernel signing key
sig_key:        25:D1:E4:42:29:FE:4A:1B:1A:8F:76:33:4E:12:A1:A7:F6:B4:A2:67
sig_hashalgo:   sha256
signature:      AF:2F:7C:C3:D9:E0:19:19:62:15:6E:C9:F8:1E:C1:BF:D3:AB:0C:C7:
                18:6C:1E:17:89:F9:DE:2E:93:3E:1B:47:B9:EF:C0:6B:1C:E5:73:82:
                53:CB:CB:59:7C:8E:4C:8F:21:1E:53:18:A5:66:D7:AD:56:BD:54:91:
```

24:64:05:67:FD:E1:B2:DD:82:EA:09:BE:E7:50:FD:22:F9:1F:94:8D:
AA:B7:62:5A:91:9C:57:A1:45:6E:37:AE:88:FA:2B:F4:C3:D6:2A:B7:
A5:59:05:A6:88:04:D4:0F:D6:4E:C7:51:DF:00:F6:AD:3D:14:05:A3:
C2:7C:57:0F:0E:E1:8C:2A:73:45:D1:AE:EF:6F:70:B3:89:D0:8F:6C:
CC:A2:82:57:57:7E:34:16:69:95:33:EC:F5:E1:4D:17:0C:98:1C:39:
A9:5F:08:82:96:83:05:42:6C:90:0E:9B:CE:FE:75:77:8F:19:F5:46:
90:31:2E:08:BE:F4:E0:9F:5D:EB:A7:48:C6:49:03:CA:73:45:B1:8F:
A7:FF:82:9A:76:F9:D3:09:1F:DA:47:F4:3B:8B:3A:1A:41:1D:6C:3E:
3F:CC:12:BA:1B:11:B2:0F:18:D1:34:F4:92:23:F5:8A:ED:75:CE:0B:
E1:66:A7:42:3D:9B:76:81:BE:2C:2C:5D:72:CA:55:F4:88:AD:B5:40:
56:E5:44:24:48:E0:E2:F5:1A:7C:96:07:9D:5C:97:9F:E5:5E:3D:AB:
76:E2:E8:B4:23:E0:F5:1F:28:0D:7C:A8:56:3F:7A:98:3E:F9:25:23:
B6:EF:D3:B6:1D:44:E9:AF:83:AC:61:FF:9B:02:28:B6:D5:87:08:4A:
9B:5B:00:9B:79:4A:18:EF:9A:D8:D0:10:55:23:68:C8:26:0C:29:05:
3A:8B:6D:29:96:19:08:FC:44:EC:4F:CF:8A:F6:B5:3C:E4:C0:19:25:
1D:5B:B5:69:EB:58:86:16:89:39:54:7B:EE:EA:9A:C6:4D:1D:F2:E9:
F7:6A:5D:CF

parm: max_bonds:Max number of bonded devices (int)

parm: tx_queues:Max number of transmit queues (default = 16) (int)

parm: num_grat_arp:Number of peer notifications to send on failover event (alias of num_unsol_na) (int)

parm: num_unsol_na:Number of peer notifications to send on failover event (alias of num_grat_arp) (int)

parm: **miimon**:Link check interval in milliseconds (int)

parm: updelay:Delay before considering link up, in milliseconds (int)

parm: **downdelay**:Delay before considering link down, in milliseconds (int)

parm: use_carrier:Use netif_carrier_ok (vs MII ioctls) in miimon; 0 for off, 1 for on (default) (int)

parm: **mode**:Mode of operation; 0 for balance-rr, **1** for active-backup, **2** for balance-xor, **3** for

broadcast, **4** for 802.3ad, **5** for balance-tlb, **6** for balance-alb (charp)

parm: primary:Primary network device to use (charp)

parm: primary_reselect:Reselect primary slave once it comes up; 0 for always (default), 1 for only if speed of primary is better, 2 for only on active slave failure (charp)

parm: lacp_rate:LACPDU tx rate to request from 802.3ad partner; 0 for slow, 1 for fast (charp)

parm: ad_select:802.3ad aggregation selection logic; 0 for stable (default), 1 for bandwidth, 2 for count (charp)

parm: min_links:Minimum number of available links before turning on carrier (int)

parm: xmit_hash_policy:balance-alb, balance-tlb, balance-xor, 802.3ad hashing method; 0 for layer 2 (default), 1 for layer 3+4, 2 for layer 2+3, 3 for encap layer 2+3, 4 for encap layer 3+4 (charp)

parm: arp_interval:arp interval in milliseconds (int)

parm: arp_ip_target:arp targets in n.n.n.n form (array of charp)

parm: arp_validate:validate src/dst of ARP probes; 0 for none (default), 1 for active, 2 for backup, 3 for all (charp)

parm: arp_all_targets:fail on any/all arp targets timeout; 0 for any (default), 1 for all (charp)

parm: fail_over_mac:For active-backup, do not set all slaves to the same MAC; 0 for none (default), 1 for active, 2 for follow (charp)

parm: all_slaves_active:Keep all frames received on an interface by setting active flag for all slaves; 0 for never (default), 1 for always. (int)

parm: resend_igmp:Number of IGMP membership reports to send on link failure (int)

parm: packets_per_slave:Packets to send per slave in balance-rr mode; 0 for a random slave, 1 packet per slave (default), >1 packets per slave. (int)

parm: lp_interval:The number of seconds between instances where the bonding driver sends learning packets to each slaves peer switch. The default is 1. (uint)

→Bonding modes

0- Bond Mode 0 (balance-rr)

Este modalidad de bonding se la llama '**Round-Robin**'. Con este método, los paquetes de red enviados son rotados entre cada una de las tarjetas que forman la interfaz del bond.

Por ejemplo, en un equipo con 4 interfaces de red **eth0**, **eth1**, **eth2** y **eth3** todas ellas como

"esclavas" de la interfaz bond0. Cuando establecemos el bond en modo 0 (Round-Robin), el primer paquete se enviará por la interfaz **eth0**, el segundo por **eth1**, el tercero por **eth2** y el 4 de nuevo por **eth0**.

1- Bond Mode 1 (active-backup)

Con esta modalidad de bonding, solo una de las interfaces está activa mientras que la otra estará en modo de "espera" hasta que la interfaz activa falle por algún motivo. Cuando falla la interfaz de red activa, la otra interfaz que está de backup asume el rol y se pone en activa.

2.- Bond Mode 2 (balance-xor)

Este método **evalúa el origen y el destino** de los paquetes en función de la "**mac address**" para determinar a través de que interfaz de red del bond enviará los paquetes. Este método enviará los paquetes por la misma interfaz que tengan el mismo origen y destino. De esta forma conseguimos una especie de balanceo y tolerancia a fallos.

3- Bond Mode 3 (broadcast)

Transmite todas las tramas por todas las **interfaces slave**. Este modo nos ofrece **tolerancia a fallos**

4- Bond Mode 4 (802.3ad)

Se trata del estandard **IEEE 802.3ad** (Dynamic link aggregation) también llamado "**port trunking**". Permite la definición de agregados ofreciendo alta disponibilidad y un aumento de la velocidad. Para poder configurar este modo necesitamos:

- Soporte de **ethtool** para obtener la velocidad y el modo del interfaz.
- El switch debe soportar el modo. Por ejemplo los CISCO lo soportan con el nombre **port trunking**.

5- Bond Mode 5 (balance-tlb)

Este modo transmite balanceando la carga entre los **slave** en función de la carga de cada **slave**. En el caso que uno falle la **MAC** salta a otro **slave**. Para calcular la carga de cada interfaz es necesario disponer de **ethtool**.

A diferencia de **802.3ad** no se necesario soporte del switch para esta configuración.

Tenemos que tener en cuenta que puede producirse que los paquetes lleguen desordenados, por lo que puede ser peor el remedio que la enfermedad, especialmente en entornos de red complejos.

6- Bond Mode 6 (balance-alb)

Balanceo de carga tanto en el envío como en la recepción. Para el balanceo de carga en la recepción lo que se hace es manipulando los replies ARP para indicar una MAC en concreto de una de las interfaces slave.

Deberemos tener en cuenta que cualquier sistema de seguridad basado en las tablas **ARP** puede quejarse o simplemente tirar el puerto de la interfaz dejando el servidor sin conectividad.

Depending on your requirement, you can set the bonding mode to any of the below 7 modes.

Mode	Policy	How it works	Fault Tolerance	Load balancing
0	Round Robin	packets are sequentially transmitted/received through each interfaces one by one.	Yes	Yes
1	Active Backup	one NIC active while another NIC is a sleep. If the active NIC goes down, another NIC becomes active. only supported in x86 environments.	Yes	No
2	XOR [exclusive OR]	In this mode the, the MAC address of the slave NIC is matched up against the incoming request's MAC and once this connection is established same NIC is used to transmit/receive for the destination MAC.	Yes	Yes
3	Broadcast	All transmissions are sent on all slaves	Yes	No
4	Dynamic Link Aggregation	aggregated NICs act as one NIC which results in a higher throughput, but also provides failover in the case that a NIC fails. Dynamic Link Aggregation requires a switch that supports IEEE 802.3ad.	Yes	Yes
5	Transmit Load Balancing (TLB)	The outgoing traffic is distributed depending on the current load on each slave interface. Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed slave.	Yes	Yes
6	Adaptive Load Balancing (ALB)	Unlike Dynamic Link Aggregation, Adaptive Load Balancing does not require any particular switch configuration. Adaptive Load Balancing is only supported in x86 environments. The receiving packets are load balanced through ARP negotiation.	Yes	Yes

→ Para Debian:

```
iface bond0 inet static
   address 192.168.0.187
   netmask 255.255.255.0
   network 192.168.0.0
   gateway 192.168.0.1
   slaves eth0 eth1
   bond_mode 1
   bond_miimon 100
   bond_downdelay 200
   bond_updelay 200
```

→ Para RedHat:

bond-00 ~ # nmcli connection add type bond con-name bond0 ifname bond0 mode active-backup miimon 100 downdelay 200 updelay 200 ipv4.addrsses 192.168.10.170/24

bond-00 ~ # nmcli c s

NAME UUID TYPE DEVICE

bond0 d27a0f99-5ca0-4b04-aced-9c74b918d8ce bond bond0

enp1s0 4865a23e-32aa-49c4-94cf-2a1f4a7ef3ac ethernet enp1s0

enp6s0 1f43b3c1-4e6b-3663-92de-061376dffcdf ethernet --

enp7s0 0633a275-8c73-35b8-9c23-93027c361447 ethernet --

bond-00 ~ # cat /etc/sysconfig/network-scripts/ifcfg-bond0

BONDING_OPTS="mode=active-backup downdelay=200 miimon=100 updelay=200"

TYPE=Bond

BONDING_MASTER=yes

PROXY_METHOD=none

BROWSER_ONLY=no

BOOTPROTO=static

IPADDR=192.168.10.170

PREFIX=24

DEFROUTE=yes

IPV4_FAILURE_FATAL=no

IPV6INIT=yes

IPV6_AUTOCONF=yes

IPV6_DEFROUTE=yes

IPV6_FAILURE_FATAL=no

IPV6_ADDR_GEN_MODE=stable-privacy

NAME=bond0

UUID=ea0f5923-acc8-428d-baf0-f6ed3f775d78

DEVICE=bond0

ONBOOT=yes

```
bond-00 ~ # nmcli connection add type bond-slave ifname enp6s0 master bond0
bond-00 ~ # nmcli connection add type bond-slave ifname enp7s0 master bond0
bond-00 ~ # nmcli c s
NAME              UUID                                  TYPE      DEVICE
enp1s0            4865a23e-32aa-49c4-94cf-2a1f4a7ef3ac  ethernet  enp1s0
bond0             ea0f5923-acc8-428d-baf0-f6ed3f775d78  bond      bond0
bond-slave-enp6s0  76755ef9-12ab-4176-83ad-5cb4e9f034b5  ethernet  enp6s0
bond-slave-enp7s0  e14e7813-79fe-4587-8f9b-dd8d3ded3e67  ethernet  enp7s0
enp6s0            1f43b3c1-4e6b-3663-92de-061376dffcdf  ethernet  --
enp7s0            0633a275-8c73-35b8-9c23-93027c361447  ethernet  --
bond-00 ~ # ip -brief a
lo               UNKNOWN        127.0.0.1/8 ::1/128
enp1s0           UP             192.168.10.160/24 fe80::d9a5:536c:4430:70c5/64
enp6s0           UP
enp7s0           UP
bond0            UP             192.168.10.170/24 fe80::2152:c01e:84c4:5fcf/64
bond-00 ~ # cat /etc/sysconfig/network-scripts/ifcfg-bond-slave-enp6s0
TYPE=Ethernet
NAME=bond-slave-enp6s0
UUID=76755ef9-12ab-4176-83ad-5cb4e9f034b5
DEVICE=enp6s0
ONBOOT=yes
MASTER=bond0
SLAVE=yes
bond-00 ~ # cat /etc/sysconfig/network-scripts/ifcfg-bond-slave-enp7s0
TYPE=Ethernet
NAME=bond-slave-enp7s0
UUID=e14e7813-79fe-4587-8f9b-dd8d3ded3e67
```

DEVICE=enp7s0

ONBOOT=yes

MASTER=bond0

SLAVE=yes

bond-00 ~ # nmcli connection up bond0

bond-00 ~ # nmcli connection show

NAME UUID TYPE DEVICE

enp1s0 4865a23e-32aa-49c4-94cf-2a1f4a7ef3ac ethernet enp1s0

bond0 eeb12ada-d663-4597-ae29-6ad01a04ed00 bond bond0

bond-slave-enp6s0 37227e09-db2c-45d9-a4e7-cf522dabb5c3 ethernet enp6s0

bond-slave-enp7s0 3967a236-e1ac-4264-aef4-2e0150e3bd0c ethernet enp7s0

enp6s0 1f43b3c1-4e6b-3663-92de-061376dffcdf ethernet --

enp7s0 0633a275-8c73-35b8-9c23-93027c361447 ethernet --

bond-00 ~ # cat /proc/net/bonding/bond0

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

Bonding Mode: fault-tolerance (active-backup)

Primary Slave: None

Currently Active Slave: enp6s0

MII Status: up

MII Polling Interval (ms): 100

Up Delay (ms): 200

Down Delay (ms): 200

Peer Notification Delay (ms): 0

Slave Interface: enp6s0

MII Status: up

Speed: Unknown

Duplex: Unknown

Link Failure Count: 0

Permanent HW addr: 52:54:00:1c:5f:0c

Slave queue ID: 0

Slave Interface: enp7s0

MII Status: up

Speed: Unknown

Duplex: Unknown

Link Failure Count: 0

Permanent HW addr: 52:54:00:6c:a1:ce

Slave queue ID: 0

bond-00 ~ # ip link

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000

link/ether 52:54:00:2c:13:81 brd ff:ff:ff:ff:ff:ff

3: enp6s0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master bond0 state UP mode DEFAULT group default qlen 1000

link/ether 52:54:00:1c:5f:0c brd ff:ff:ff:ff:ff:ff

4: enp7s0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master bond0 state UP mode DEFAULT group default qlen 1000

link/ether 52:54:00:1c:5f:0c brd ff:ff:ff:ff:ff:ff

7: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default qlen 1000

link/ether 52:54:00:1c:5f:0c brd ff:ff:ff:ff:ff:ff

bond-00 ~ # ls /sys/class/net/

bond0 bonding_masters enp1s0 enp6s0 enp7s0 lo

bond-00 ~ # cat /sys/class/net/bonding_masters

bond0

bond-00 ~ # cat /sys/class/net/bond0/bonding/slaves

enp6s0 enp7s0

bond-00 ~ # cat /sys/class/net/bond0/bonding/miimon

100

bond-00 ~ # cat /sys/class/net/bond0/bonding/mode

active-backup 1

bond-00 ~ # cat /proc/net/bonding/bond0

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

Bonding Mode: fault-tolerance (active-backup)

Primary Slave: None

Currently Active Slave: enp6s0

MII Status: up

MII Polling Interval (ms): 100

Up Delay (ms): 200

Down Delay (ms): 200

Peer Notification Delay (ms): 0

Slave Interface: enp6s0

MII Status: up

Speed: Unknown

Duplex: Unknown

Link Failure Count: 0

Permanent HW addr: 52:54:00:1c:5f:0c

Slave queue ID: 0

Slave Interface: enp7s0

MII Status: up

Speed: Unknown

Duplex: Unknown

Link Failure Count: 0

Permanent HW addr: 52:54:00:6c:a1:ce

Slave queue ID: 0

→**ifenslave** is a tool to attach and detach slave network devices to a bonding device. A bonding device will act like a normal

Ethernet network device to the kernel, but will send out the packets via the slave devices using a simple round-robin sched-

uler. This allows for simple load-balancing, identical to "channel bonding" or "trunking" techniques used in switches.

OPTIONS

-a, --all-interfaces

Show information about all interfaces.

-c, --change-active

Change active slave.

-d, --detach

Removes slave interfaces from the bonding device.

-f, --force

Force actions to be taken if one of the specified interfaces appears not to belong to an Ethernet device.

-h, --help

Display a help message and exit.

-u, --usage

Show usage information and exit.

-v, --verbose

Print warning and debug messages.

-V, --version

Show version information and exit.

If not options are given, the default action will be to enslave interfaces.

EXAMPLE

The following example shows how to setup a bonding device and enslave two real Ethernet devices to it:

modprobe bonding

ifconfig bond0 192.168.0.1 netmask 255.255.0.0

ifenslave bond0 eth0 eth1

bond-00 ~ # ip -brief addr

lo UNKNOWN 127.0.0.1/8 ::1/128

enp1s0 UP 192.168.10.160/24 fe80::d9a5:536c:4430:70c5/64

enp6s0 UP

enp7s0 UP

bond0 UP 192.168.10.170/24 fe80::9381:7257:88b8:8dca/64

bond-00 ~ # ifenslave bond0 -dv enp6s0

ifenslave.c:v1.1.0 (December 1, 2003)

o Donald Becker (becker@cesdis.gsfc.nasa.gov).

o Detach support added on 2000/10/02 by Willy Tarreau (willy at meta-x.org).

o 2.4 kernel support added on 2001/02/16 by Chad N. Tindel

(ctindel at ieee dot org).

ABI ver is 2

Slave enp6s0: flags 1843.

Slave 'enp6s0': MTU set to 1500.

bond-00 ~ # ip -brief addr

lo UNKNOWN 127.0.0.1/8 ::1/128

enp1s0 UP 192.168.10.160/24 fe80::d9a5:536c:4430:70c5/64

enp6s0 DOWN

enp7s0 UP

bond0 UP 192.168.10.170/24 fe80::9381:7257:88b8:8dca/64

bond-00 ~ # ifenslave -v bond0 enp6s0

ifenslave.c:v1.1.0 (December 1, 2003)

o Donald Becker (becker@cesdis.gsfc.nasa.gov).

o Detach support added on 2000/10/02 by Willy Tarreau (willy at meta-x.org).

o 2.4 kernel support added on 2001/02/16 by Chad N. Tindel

(ctindel at ieee dot org).

ABI ver is 2

current hardware address of master 'bond0' is 52:54:00:1c:5f:0c, type 1

Interface 'enp6s0': flags set to 1002.

Interface 'enp6s0': address cleared

bond-00 ~ # ip -brief addr

lo UNKNOWN 127.0.0.1/8 ::1/128

enp1s0 UP 192.168.10.160/24 fe80::d9a5:536c:4430:70c5/64

enp6s0 UP

enp7s0 UP

bond0 UP 192.168.10.170/24 fe80::9381:7257:88b8:8dca/64

bond-00 ~ # cat /proc/net/bonding/bond0

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

Bonding Mode: fault-tolerance (active-backup)

Primary Slave: None

Currently Active Slave: enp7s0

MII Status: up

MII Polling Interval (ms): 100

Up Delay (ms): 200

Down Delay (ms): 200

Peer Notification Delay (ms): 0

Slave Interface: enp7s0

MII Status: up

Speed: Unknown

Duplex: Unknown

Link Failure Count: 0

Permanent HW addr: 52:54:00:6c:a1:ce

Slave queue ID: 0

Slave Interface: enp6s0

MII Status: up

Speed: Unknown

Duplex: Unknown

Link Failure Count: 0

Permanent HW addr: 52:54:00:1c:5f:0c

Slave queue ID: 0

bond-00 ~ # nmcli dev mod bond0 +bond.options "primary=enp6s0"

bond-00 ~ # cat /proc/net/bonding/bond0

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

Bonding Mode: fault-tolerance (active-backup)

Primary Slave: enp6s0 (primary_reselect always)

Currently Active Slave: enp6s0

MII Status: up

MII Polling Interval (ms): 100

Up Delay (ms): 200

Down Delay (ms): 200

Peer Notification Delay (ms): 0

Slave Interface: enp7s0

MII Status: up

Speed: Unknown

Duplex: Unknown

Link Failure Count: 0

Permanent HW addr: 52:54:00:6c:a1:ce

Slave queue ID: 0

Slave Interface: enp6s0

MII Status: up

Speed: Unknown

Duplex: Unknown

Link Failure Count: 0

Permanent HW addr: 52:54:00:1c:5f:0c

Slave queue ID: 0

→ VLAN regular and VLAN over a BOND:

→Regular VLAN.

bond-00 ~ # modprobe --first-time 8021q

bond-00 ~ # modinfo 8021q

filename: /lib/modules/4.18.0-240.15.1.el8_3.x86_64/kernel/net/8021q/8021q.ko.xz

version: 1.8

license: GPL

alias: rtnl-link-vlan

rhelversion: 8.3

srcversion: EC9F01F9B4371CEBBBCF2DE

depends: mrp,garp

intree: Y

name: 8021q

vermagic: 4.18.0-240.15.1.el8_3.x86_64 SMP mod_unload modversions

sig_id: PKCS#7

signer: Red Hat Enterprise Linux kernel signing key

```
sig_key:        25:D1:E4:42:29:FE:4A:1B:1A:8F:76:33:4E:12:A1:A7:F6:B4:A2:67
sig_hashalgo:   sha256
signature:      10:80:A8:F1:C4:9F:13:30:2E:CA:A0:31:A8:C7:1E:36:11:FF:D6:6B:
                ED:6D:0F:F2:1C:38:DD:8F:D3:F7:83:B4:BB:69:AB:60:63:95:59:E8:
                A4:10:A5:71:08:4E:D3:67:90:D2:22:FF:5E:40:A2:0B:AB:90:B0:53:
                E9:E1:EB:AE:DF:22:50:43:72:2A:F9:44:3B:AC:C5:CE:5F:20:5D:67:
                12:4D:AF:66:D6:8E:5A:FB:C9:8A:D3:3A:97:81:3E:C1:5B:1B:4A:76:
                EF:E4:04:D8:4E:C6:8C:9A:E4:37:35:25:AC:7B:2B:96:D4:25:EC:63:
                35:DF:FD:49:70:19:29:1C:0B:68:9C:5F:29:E4:87:83:03:31:FE:63:
                EA:87:06:53:9F:78:E9:AA:E8:4D:81:F1:45:CC:1A:4D:05:94:03:2E:
                62:F8:E3:4C:DA:42:74:64:01:DC:36:9B:99:7E:0E:9F:F0:F4:FF:27:
                A8:A4:04:B6:68:88:CA:E0:6D:AA:46:09:E3:AF:3D:83:30:23:91:D0:
                30:D0:3C:9F:CA:56:63:9D:B5:75:2F:5D:97:13:E0:A2:70:63:FC:C9:
                44:E5:AC:29:46:DE:2D:C1:4F:D6:E5:8F:E5:DD:89:BB:78:64:9D:A7:
                C2:AE:1F:E5:DF:C9:9B:B4:87:65:53:55:C2:FE:33:D0:B8:97:B4:F7:
                D1:26:22:30:ED:2C:1C:BF:C4:4A:6F:91:40:D3:64:69:0F:F8:C5:38:
                CC:BE:A9:1F:3E:9B:47:5E:71:E3:D6:68:A1:11:F7:AF:21:27:6E:60:
                74:1C:63:5A:75:A0:8F:FC:4E:41:8E:8A:BB:69:04:16:6E:77:44:72:
                E3:59:46:09:8C:99:01:19:7F:63:A0:54:2F:2F:09:24:24:DE:09:A5:
                F4:E7:21:8A:30:B8:73:6D:6B:AB:74:4B:5C:BD:6C:15:CC:2B:10:E7:
                77:6A:D7:00:EF:DC:AF:4F:53:51:72:22:79:1A:84:90:22:6D:AD:66:
                3A:88:CE:62
```

bond-00 ~ # ip -br a

```
lo              UNKNOWN        127.0.0.1/8 ::1/128
enp1s0          UP             192.168.10.160/24 fe80::d9a5:536c:4430:70c5/64
enp6s0          UP
enp7s0          UP
```

bond-00 ~ # nmcli connection add type vlan con-name VLAN20 dev enp6s0 ipv4.addresses

192.168.20.20/24 id 20 gw4 192.168.20.1

bond-00 ~ # vim /etc/sysconfig/network-scripts/ifcfg-VLAN20

```
VLAN=yes
TYPE=Vlan
PHYSDEV=enp6s0
VLAN_ID=20
REORDER_HDR=yes
GVRP=no
MVRP=no
HWADDR=
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
IPADDR=192.168.20.20
PREFIX=24
GATEWAY=192.168.20.1
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=VLAN20
UUID=d8561ec1-7d91-46fb-8274-0d6e51ac37c9
ONBOOT=yes
```

bond-00 ~ # ip -br a

```
lo               UNKNOWN        127.0.0.1/8 ::1/128
```

```
enp1s0          UP          192.168.10.160/24 fe80::d9a5:536c:4430:70c5/64
enp6s0          UP
enp7s0          UP
enp6s0.20@enp6s0 UP          192.168.20.20/24 fe80::36b7:a5c4:a09b:4ff9/64
bond-00 ~ # nmcli c s
NAME    UUID                                  TYPE      DEVICE
VLAN20  d8561ec1-7d91-46fb-8274-0d6e51ac37c9  vlan      enp6s0.20
enp1s0  4865a23e-32aa-49c4-94cf-2a1f4a7ef3ac  ethernet  enp1s0
enp6s0  1f43b3c1-4e6b-3663-92de-061376dffcdf  ethernet  --
enp7s0  0633a275-8c73-35b8-9c23-93027c361447  ethernet  --
bond-00 ~ # nmcli -p connection show VLAN20
===============================================================================
                Detalles del perfil de conexiones (VLAN20)
===============================================================================
connection.id:                          VLAN20
connection.uuid:                        c5815da4-fcb9-4b04-a82c-944a73c203b3
connection.stable-id:                   --
connection.type:                        vlan
connection.interface-name:              --
connection.autoconnect:                 sí
connection.autoconnect-priority:        0
connection.autoconnect-retries:         -1 (default)
connection.multi-connect:               0 (default)
connection.auth-retries:                -1
connection.timestamp:                   0
connection.read-only:                   no
connection.permissions:                 --
connection.zone:                        --
```

connection.master: --

connection.slave-type: --

…

→ VLAN over BOND.

The active-backup, balance-tlb and balance-alb modes do not require any specific configuration of the switch. Other bonding modes require configuring the switch to aggregate the links. For example, a Cisco switch requires EtherChannel for Modes 0, 2, and 3, but for Mode 4 LACP and EtherChannel are required.

bond-00 ~ # nmcli connection add type bond con-name bond0 ifname bond0 mode active-backup miimon 200 downdelay 200 updelay 200 ipv4.addrsses 192.168.10.170/24 gw4 192.168.10.5

bond-00 ~ # nmcli connection add type bond-slave ifname enp6s0 master bond0

bond-00 ~ # nmcli connection add type bond-slave ifname enp7s0 master bond0

bond-00 ~ # grep -r "ONBOOT=yes" /etc/sysconfig/network-scripts/ | cut -f1 -d":" | xargs grep -E "IPADDR|SLAVE"

/etc/sysconfig/network-scripts/ifcfg-enp1s0:IPADDR=192.168.10.160

/etc/sysconfig/network-scripts/ifcfg-bond0:IPADDR=192.168.10.170

/etc/sysconfig/network-scripts/ifcfg-bond-slave-enp6s0:SLAVE=yes

/etc/sysconfig/network-scripts/ifcfg-bond-slave-enp7s0:SLAVE=yes

bond-00 ~ # ip -br a

lo UNKNOWN 127.0.0.1/8 ::1/128

enp1s0 UP 192.168.10.160/24 fe80::d9a5:536c:4430:70c5/64

enp6s0 UP

enp7s0 UP

bond0 UP 192.168.10.170/24 fe80::b4ee:f9bb:cab5:9b4f/64

bond-00 ~ # ip route

default via 192.168.10.5 dev enp1s0 proto static metric 100

default via 192.168.10.5 dev bond0 proto static metric 300

192.168.10.0/24 dev enp1s0 proto kernel scope link src 192.168.10.160 metric 100

192.168.10.0/24 dev bond0 proto kernel scope link src 192.168.10.170 metric 300

bond-00 ~ # nmcli connection add type vlan con-name VLAN20 dev bond0 ipv4.addresses 192.168.20.180/24 id 20

bond-00 ~ # nmcli c s

```
NAME             UUID                                  TYPE      DEVICE
VLAN20           ace231bf-584d-4922-ab76-e95501a0eee1  vlan      bond0.20
enp1s0           4865a23e-32aa-49c4-94cf-2a1f4a7ef3ac  ethernet  enp1s0
bond0            9237abb5-755c-4da2-b9a6-2f4e594e267e  bond      bond0
bond-slave-enp6s0  c64b1897-8e34-450d-a002-77426eaa5635  ethernet  enp6s0
bond-slave-enp7s0  6b5c3466-ee65-4f94-80e5-52dd6d620a02  ethernet  enp7s0
```

bond-00 ~ # ip -br a

```
lo               UNKNOWN        127.0.0.1/8 ::1/128
enp1s0           UP             192.168.10.160/24 fe80::d9a5:536c:4430:70c5/64
enp6s0           UP
enp7s0           UP
bond0            UP             192.168.10.170/24 fe80::b4ee:f9bb:cab5:9b4f/64
bond0.20@bond0   UP             192.168.20.180/24 fe80::204d:c1a9:446f:acb2/64
```

bond-00 ~ # vim /etc/sysconfig/network-scripts/ifcfg-VLAN20

```
VLAN=yes
TYPE=Vlan
PHYSDEV=bond0
VLAN_ID=20
REORDER_HDR=yes
GVRP=no
MVRP=no
HWADDR=
PROXY_METHOD=none
```

BROWSER_ONLY=no

BOOTPROTO=static

IPADDR=192.168.20.180

PREFIX=24

DEFROUTE=yes

IPV4_FAILURE_FATAL=no

IPV6INIT=yes

IPV6_AUTOCONF=yes

IPV6_DEFROUTE=yes

IPV6_FAILURE_FATAL=no

IPV6_ADDR_GEN_MODE=stable-privacy

NAME=VLAN20

UUID=8707287f-4fd7-4a78-864d-233078afabdc

ONBOOT=yes

→ **Rapid Spanning Tree Protocol (RSTP)** es un protocolo de red de la segunda capa OSI, (nivel de enlace de datos), que gestiona enlaces redundantes. Especificado en IEEE 802.1w, es una evolución del Spanning tree Protocol (STP), reemplazándolo en la edición 2004 del 802.1d. RSTP reduce significativamente el tiempo de convergencia de la topología de la red cuando ocurre un cambio en la topología.

Roles de los puertos RSTP:

- **Raíz** – Es un puerto de envío elegido para la topología Spanning Tree.
- **Designado** – Un puerto de envío elegido para cada segmento de la red.
- **Alternativo** – Un camino alternativo hacia el Puente Raíz. Este camino es distinto al que usan los puertos raíz.
- **Respaldo** – Un camino de respaldo/redundante (de mayor costo) a un segmento donde hay otro puerto ya conectado.
- **Deshabilitado** – Un puerto que no tiene un papel dentro de la operación de Spanning Tree.

Estados de los puertos RSTP:

- **Learning** - Escucha BPDUs y guarda información relevante.
- **Forwarding** - Una vez ejecutado el algoritmo para evitar bucles, los puertos activos pasan a

este estado.

- **Discarding** - No recibe BPDUs por lo cual no se encuentra participando en la instancia activa de STP

Los puertos raíz y designado forman parte de la topología activa. Los puertos alternativo y de respaldo no están incluidos en la topología activa

RSTP monitorea el estado de todas las trayectorias:

- Si una dirección activa se cae, RSTP activa las direcciones redundantes.
- Configura de nuevo la topología de la red adecuadamente.

RSTP se ha convertido en el protocolo preferido para prevenir bucles de capa 2 en topologías que incluyen redundancia. Además de que el 802.1w contiene mejoras, retiene compatibilidad con su antecesor 802.1D dejando algunos parámetros sin cambiar. Por ejemplo, RSTP mantiene el mismo formato de BPDU que STP sólo que cambia el campo de versión, el cual se le asigna el valor de 2.

RSTP también define el concepto de *edge-port*, el cual también se menciona en STP como *PortFast*, en donde el puerto se configura como tal cuando se sabe que nunca será conectado hacia otro switch de manera que pasa inmediatamente al estado de direccionamiento sin esperar los pasos intermedios del algoritmo –etapas de escucha y aprendizaje- los cuales consumen tiempo. Los puertos que no son *edge-ports* pueden ser punto a punto o compartidos. El tipo de enlace es detectado automáticamente, pero puede ser configurado explícitamente para hacer más rápida la convergencia.

Objetivos del RSTP

- Disminuir el tiempo de convergencia cuando un enlace falla.
  - De 30 ó 60 s a milisegundos.
- Soporta redes extendidas.
  - 2048 conexiones o 4096 puertos interconectados en comparación con 256 puertos conectados en STP.
- Compatibilidad con STP.

Configure Spanning Tree Properties

Step 1. Log in to the switch console. The default username and password is cisco/cisco. If you have configured a new username or password, enter the credentials instead.

Note: To learn how to access an SMB switch CLI through SSH or Telnet, click here.

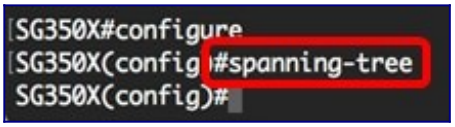
Note: The commands may vary depending on the exact model of your switch. In this example, the SG350X-48MP switch is accessed through Telnet.

Step 2. From the Privileged EXEC mode of the switch, enter the Global Configuration mode by entering the following:

```
SG350X#configure
```

Step 3. To enable the STP functionality on the switch, enter the following:

```
SG350X(config)#spanning tree
```

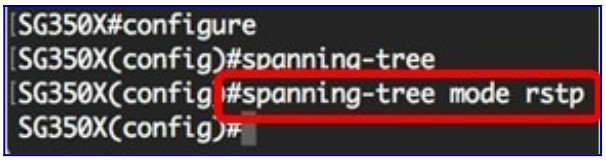


Step 4. To configure the STP protocol to run on the switch, enter the following:

```
SG350X(config)#spanning-tree mode [stp |rstp | mst]
```

The options are:

- stp — Classic STP provides a single path between any two endpoints, eliminating and preventing networking loops.
- rstp — RSTP detects network topologies to provide faster convergence of the spanning tree. This option is enabled by default.
- mst — MSTP is based on RSTP. It detects Layer 2 loops, and attempts to mitigate them by preventing the involved port from transmitting traffic.



Note: In this example, rstp is used.

Step 5. To set the default path cost method, enter the following:

```
SG350X(config)#spanning-tree pathcost method [long | short]
```

The options are:

- long — Specifies the value for port path costs. The range is from one up to 200000000.
- short — Specifies the value for port path costs. The range is from one to 65535.

```
SG350X#configure
SG350X(config)#spanning-tree
SG350X(config)#spanning-tree mode rstp
SG350X(config)#spanning-tree pathcost method long
SG350X(config)#
```

Step 6. To configure the switch STP priority, which is used to determine which bridge is selected as the root bridge, enter the following:

```
SG350X(config)#spanning-tree priority [priority-number]
```

- priority-number — Specifies the bridge priority. The range is from 0 up to 61440.

```
SG350X#configure
SG350X(config)#spanning-tree
SG350X(config)#spanning-tree mode rstp
SG350X(config)#spanning-tree pathcost method long
SG350X(config)#spanning-tree priority 32768
SG350X(config)#
```

Note: In this example, 32768 is used.

Step 7. (Optional) To configure how often the switch broadcasts Hello messages to other devices, enter the following:

```
SG350X(config)#spanning-tree hello-time [seconds]
```

- seconds — Specifies the spanning tree Hello time in seconds. The range is from 1 up to 10 seconds. The default value is 2 seconds.

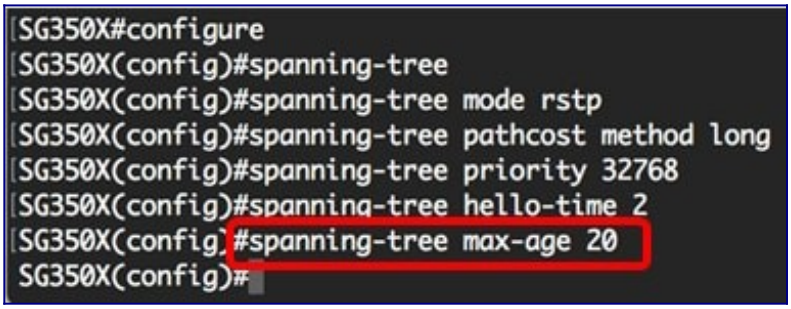
```
SG350X#configure
SG350X(config)#spanning-tree
SG350X(config)#spanning-tree mode rstp
SG350X(config)#spanning-tree pathcost method long
SG350X(config)#spanning-tree priority 32768
SG350X(config)#spanning-tree hello-time 2
SG350X(config)#
```

Note: In this example, the default Hello time of 2 seconds is used.

Step 8. (Optional) To configure the STP maximum age, enter the following:

```
SG350X(config)#spanning-tree max-age [seconds]
```

- seconds — Specifies the spanning tree bridge maximum age in seconds. The range is from six up to 40 seconds. The default value is 20 seconds.

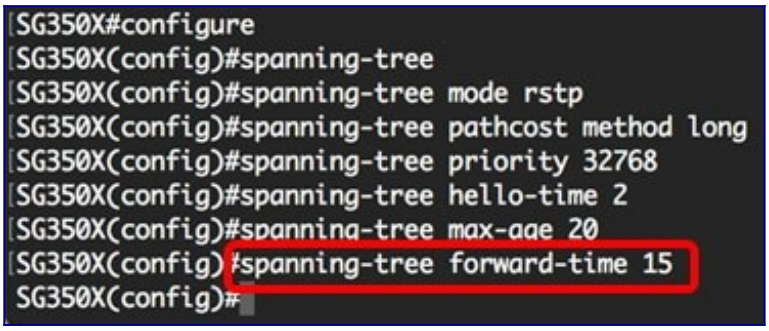


Note: In this example, the default value of 20 seconds is used.

Step 9. (Optional) To configure the STP bridge forward time, which is the amount of time a port remains in the listening and learning states before entering the forwarding state, enter the following:

```
SG350X(config)#spanning-tree forward-time [seconds]
```

- seconds — Specifies the spanning tree forward time in seconds. The range is from four up to 30 seconds. The default value is 15 seconds.

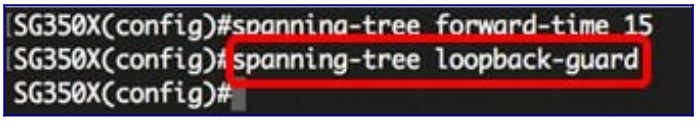


Note: In this example, the default value of 15 seconds is used.

Step 10. (Optional) To enable STP Loopback Guard, enter the following:

```
SG350X(config)#spanning-tree loopback-guard
```

Note: Enabling this feature checks if a root port or an alternate root port receives Bridge Protocol Data Units (BPDUs). In this example, STP Loopback Guard is enabled.



Step 11. Enter the **exit** command to go back to the Privileged EXEC mode:

```
SG350X(config)#exit
```

```
SG350X#configure
SG350X(config)#spanning-tree
SG350X(config)#spanning-tree mode rstp
SG350X(config)#spanning-tree pathcost method long
SG350X(config)#spanning-tree priority 32768
SG350X(config)#spanning-tree hello-time 2
SG350X(config)#spanning-tree max-age 20
SG350X(config)#spanning-tree forward-time 15
SG350X(config)#spanning-tree loopback-guard
SG350X(config)#exit
SG350X#
```

Step 12. (Optional) To display the STP settings on the switch, enter the following:

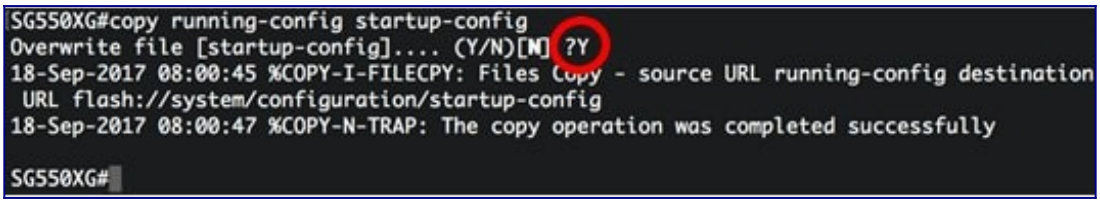
```
SG350X#show spanning-tree
```

```
SG350X(config)#exit
SG350X#show spanning-tree

Spanning tree enabled mode RSTP
Default port cost method:  long
Loopback guard:    Enabled


  Root ID    Priority    32768
             Address     00:eb:d5:5e:09:40
             Cost        40000
             Port        gi1/0/2
             Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
  Bridge ID  Priority    32768
             Address     40:a6:e8:e6:f4:d3
             Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec

  Number of topology changes 5 last change occurred 00:49:25 ago
  Times:  hold 1, topology change 35, notification 2
          hello 2, max age 20, forward delay 15

Interfaces
  Name      State    Prio.Nbr    Cost     Sts   Role PortFast      Type
--------- -------- --------- -------- ------ ---- -------- ----------------
 gi1/0/1  enabled    128.1     20000    Dscr  Altn    No      P2P (RSTP)
 gi1/0/2  enabled    128.2     20000    Frw   Root    No      P2P (RSTP)
 gi1/0/3  enabled    128.3     2000000  Dsbl  Dsbl    No          -
 gi1/0/4  enabled    128.4     20000    Dscr  Altn    No      P2P (RSTP)
More: <space>,  Quit: q or CTRL+Z, One line: <return>
```

Step 13. (Optional) In the Privileged EXEC mode of the switch, save the configured settings to the startup configuration file by entering the following:

```
SG350X#copy running-config startup-config
```

Step 14. (Optional) Press **Y** for Yes or **N** for No on your keyboard once the Overwrite file [startup-config]… prompt appears.



You should now have successfully configured the STP settings on your switch through the CLI.

BIBLIOGRAFIA:

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/networking_guide/ch-configure_network_bonding

https://www.unixmen.com/linux-basics-create-network-bonding-on-centos-76-5/

https://www.thegeekdiary.com/centos-rhel-7-how-to-create-an-interface-bonding-nic-teaming-using-nmcli/

https://www.raulprietofernandez.net/blog/redes/como-configurar-un-bonding-de-interfaces-en-gnu-linux-debian

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/networking_guide/sec-configure_802_1q_vlan_tagging_using_the_command_line_tool_nmcli

https://www.cisco.com/c/en/us/support/docs/smb/switches/cisco-small-business-300-series-managed-switches/smb5760-configure-stp-settings-on-a-switch-through-the-cli.html