Interface bonding

Linux has a bonding driver that provides a method for aggregating multiple network interfaces into a single logical interface. Behavior of bonded interfaces depends on selected bonding mode.
For more details on the supported modes please check Documentation/networking/bonding.txt in Linux's kernel sources.

Configure interface bonding

It is recommended to configure bonding via iproute2 (netlink) or sysfs, the old **ifenslave** control utility is obsolete.

First you will need to load the **bonding** kernel module. You'll need to specify at least the mode you want the bonding to be created in.

```
modprobe bonding mode=balance-alb miimon=100
```

After that you need to set some IP address to the new bonding interface and bring is up

```
ip addr add 10.1.1.2/16 dev bond0
ip link set bond0 up
```

And the last step is to add the network interfaces to the new virtual bonding interface.

```
ip link set eth0 master bond0
ip link set eth1 master bond0
```

Alternative method of creating/manipulating the bonding interfaces via sysfs

```
echo balance-alb > /sys/class/net/bond0/bonding/mode
ip addr add 10.1.1.2/16 dev bond0
ip link set bond0 up
echo 100 > /sys/class/net/bond0/bonding/miimon
echo +eth0 > /sys/class/net/bond0/bonding/slaves
echo +eth1 > /sys/class/net/bond0/bonding/slaves
```

Bonding modes

- **mode 0 / balance-rr** - Round-robin policy: Transmit packets in sequential order from the first available slave through the,last.This mode provides load balancing and fault tolerance.
- **mode 1 / active-backup** - Active-backup policy: Only one slave in the bond is active.,A different slave becomes active if, and only if, the active slave fails., The bond's MAC address is externally visible on only one port (network adapter) to avoid confusing the switch.
- **mode 2 / balance-xor** - XOR policy: Transmit based on the selected transmit,hash policy.,The default policy is a simple [(source MAC address XOR'd with destination MAC address XOR packet

type ID) modulo slave count].,Alternate transmit policies may be,selected via the xmit_hash_policy option, described below. This mode provides load balancing and fault tolerance.
- **mode 3 / broadcast** - Broadcast policy: transmits everything on all slave,interfaces.,This mode provides fault tolerance.
- **mode 4 / 802.3ad** - IEEE 802.3ad Dynamic link aggregation.,Creates,aggregation groups that share the same speed and,duplex settings. Utilizes all slaves in the active,aggregator according to the 802.3ad specification.
- **mode 5 / balance-tlb** - Adaptive transmit load balancing: channel bonding that,does not require any special switch support.In tlb_dynamic_lb=1 mode; the outgoing traffic is,distributed according to the current load (computed,relative to the speed)on each slave., In tlb_dynamic_lb=0 mode; the load balancing based on,current load is disabled and the load is distributed,only using the hash distribution.Incoming traffic is received by the current slave.,If the receiving slave fails, another slave takes over,the MAC address of the failed receiving slave.
- **mode 6 / balance-alb** - Adaptive load balancing: includes balance-tlb plus,receive load balancing (rlb) for IPV4 traffic, and,does not require any special switch support.,The,receive load balancing is achieved by ARP negotiation.,The bonding driver intercepts the ARP Replies sent by,the local system on their way out and overwrites the,source hardware address with the unique hardware, address of one of the slaves in the bond such that,different peers use different hardware addresses for,the server.

mode 4 / 802.3ad and **mode 5 / balance-tlb** have some prerequisites. Check kernel docs for more information

Balancing algorithm modes

xmit_hash_policy option sets the balancing algorithm used.

The default value is layer2.

- **layer2** - Uses XOR of hardware MAC addresses to generate the hash. This algorithm will place all traffic to a particular network peer on the same slave.
- **layer2+3** - Uses XOR of hardware MAC addresses and IP addresses to generate the hash. This algorithm will place all traffic to a particular network peer on the same slave.
- **layer3+4** - This policy uses upper layer protocol information, when available, to generate the hash. This allows for traffic to a particular network peer to span multiple slaves, although a single connection will not span multiple slaves.
- **encap2+3** - This policy uses the same formula as layer2+3 but it relies on skb_flow_dissect to obtain the header fields which might result in the use of inner headers if an encapsulation protocol is used.
- **encap3+4** - This policy uses the same formula as layer3+4 but it relies on skb_flow_dissect to obtain the header fields which might result in the use of inner headers if an encapsulation protocol is used.

Alternative methods

libteam provides an alternative to bonding driver. The main difference is that Team driver kernel part

contains only essential code and the rest of the code (link validation, LACP implementation, decision making, etc.) is run in userspace as a part of teamd daemon.

More information can be found at: libteam.org

Sources

* Originally written by User lamerix

howtos network bonding,network bonding,bonding