NAME

lagg - link aggregation and link failover interface

SYNOPSIS

To compile this driver into the kernel, place the following line in your kernel configuration file:

```
device lagg
```

Alternatively, to load the driver as a module at boot time, place the following line in loader.conf(5):

```
if_lagg_load="YES"
```

DESCRIPTION

The **lagg** interface allows aggregation of multiple network interfaces as one virtual **lagg** interface for the purpose of providing fault-tolerance and high-speed links.

Each **lagg** interface is created at runtime using interface cloning. This is most easily done with the ifconfig(8) **create** command or using the *cloned_interfaces* variable in rc.conf(5).

A **lagg** interface can be created using the **ifconfig lagg***N* **create** command. It can use different link aggregation protocols specified using the **laggproto** *proto* option. Child interfaces can be added using the **laggport** *child-iface* option and removed using the **-laggport** *child-iface* option.

The driver currently supports the aggregation protocols **failover** (the default), **lacp**, **loadbalance**, **roundrobin**, **broadcast**, and **none**. The protocols determine which ports are used for outgoing traffic and whether a specific port accepts incoming traffic. The interface link state is used to validate if the port is active or not.

failover Sends traffic only through the active port. If the master port becomes unavailable, the next active port is used. The first interface added is the master port; any interfaces added after that are used as failover devices.

By default, received traffic is only accepted when it is received through the active port. This constraint can be relaxed by setting the *net.link.lagg.failover_rx_all* sysctl(8) variable to a nonzero value, which is useful for certain bridged network setups.

lacp Supports the IEEE 802.1AX (formerly 802.3ad) Link Aggregation Control Protocol (LACP) and the Marker Protocol. LACP will negotiate a set of aggregable links with the peer in to one or more Link Aggregated Groups. Each LAG is composed of ports of the same speed, set to full-duplex operation. The traffic will be balanced across the ports in the

LAG with the greatest total speed, in most cases there will only be one LAG which contains all ports. In the event of changes in physical connectivity, Link Aggregation will quickly converge to a new configuration.

loadbalance Balances outgoing traffic across the active ports based on hashed protocol header information and accepts incoming traffic from any active port. This is a static setup and does not negotiate aggregation with the peer or exchange frames to monitor the link. The hash includes the Ethernet source and destination address, and, if available, the VLAN tag, and the IP source and destination address.

roundrobin Distributes outgoing traffic using a round-robin scheduler through all active ports and accepts incoming traffic from any active port. Using **roundrobin** mode can cause unordered packet arrival at the client. Throughput might be limited as the client performs CPU-intensive packet reordering.

broadcast Sends frames to all ports of the LAG and receives frames on any port of the LAG.

none This protocol is intended to do nothing: it disables any traffic without disabling the **lagg** interface itself.

The MTU of the first interface to be added is used as the lagg MTU. All additional interfaces are required to have exactly the same value.

The **loadbalance** and **lacp** modes will use the RSS hash from the network card if available to avoid computing one, this may give poor traffic distribution if the hash is invalid or uses less of the protocol header information. Local hash computation can be forced per interface by setting the **-use_flowid** ifconfig(8) flag. The default for new interfaces is set via the *net.link.lagg.default_use_flowid* sysctl(8).

When creating a **lagg** interface, the **laggtype** can be specified as either **ethernet** or **infiniband**. If neither is specified then the default is **ethernet**.

EXAMPLES

Create a link aggregation using LACP with two bge(4) Gigabit Ethernet interfaces:

```
# ifconfig bge0 up
# ifconfig bge1 up
# ifconfig lagg0 create
# ifconfig lagg0 laggproto lacp laggport bge0 laggport bge1 \
        192.168.1.1 netmask 255.255.255.0
```

Create a link aggregation using ROUNDROBIN with two bge(4) Gigabit Ethernet interfaces and set a stride of 500 packets per interface:

```
# ifconfig bge0 up
# ifconfig bge1 up
# ifconfig lagg0 create
# ifconfig lagg0 laggproto roundrobin laggport bge0 laggport bge1 \
        192.168.1.1 netmask 255.255.255.0
# ifconfig lagg0 rr_limit 500
```

The following example uses an active failover interface to set up roaming between wired and wireless networks using two network devices. Whenever the wired master interface is unplugged, the wireless failover device will be used:

```
# ifconfig em0 ether 00:11:22:33:44:55 up
# ifconfig wlan0 create wlandev ath0 ssid my_net up
# ifconfig lagg0 create
# ifconfig lagg0 laggproto failover laggport em0 laggport wlan0 \
        192.168.1.1 netmask 255.255.255.0
```

(Note the MAC address of the wired device is forced to match that of the wireless device, ‘00:11:22:33:44:55’ in this example, as some common wireless devices will not allow MAC addresses to be changed.)

The following example shows how to create an infiniband failover interface.

```
# ifconfig ib0 up
# ifconfig ib1 up
# ifconfig lagg0 create laggtype infiniband
# ifconfig lagg0 laggproto failover laggport ib0 laggport ib1 \
        1.1.1.1 netmask 255.255.255.0
```

Configure two ethernets for failover with static IP in /etc/rc.conf(5):

```
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto failover laggport bge0 laggport bge1 \
        10.1.29.21/24"
ifconfig_bge0="up"
ifconfig_bge1="up"
```

SEE ALSO

ng_one2many(4), rc.conf(5), ifconfig(8), sysctl(8)

HISTORY

The **lagg** device first appeared in FreeBSD 6.3.

AUTHORS

The **lagg** driver was written under the name **trunk** by Reyk Floeter <*reyk@openbsd.org*>. The LACP implementation was written by YAMAMOTO Takashi for NetBSD.

BUGS

There is no way to configure LACP administrative variables, including system and port priorities. The current implementation always performs active-mode LACP and uses 0x8000 as system and port priorities.