1. Introduction

Linux distinguishes between administrative and operational state of an interface. Administrative state is the result of "ip link set dev <dev> up or down" and reflects whether the administrator wants to use the device for traffic.

However, an interface is not usable just because the admin enabled it - ethernet requires to be plugged into the switch and, depending on a site's networking policy and configuration, an 802.1X authentication to be performed before user data can be transferred. Operational state shows the ability of an interface to transmit this user data.

Thanks to 802.1X, userspace must be granted the possibility to influence operational state. To accommodate this, operational state is split into two parts: Two flags that can be set by the driver only, and a RFC2863 compatible state that is derived from these flags, a policy, and changeable from userspace under certain rules.

2. Querying from userspace

Both admin and operational state can be queried via the netlink operation RTM_GETLINK. It is also possible to subscribe to RTMGRP_LINK to be notified of updates. This is important for setting from userspace.

These values contain interface state:

ifinfomsg::if_flags & IFF_UP:
 Interface is admin up
ifinfomsg::if_flags & IFF_RUNNING:
 Interface is in RFC2863 operational state UP or UNKNOWN. This is for backward compatibility, routing daemons, dhcp clients can use this flag to determine whether they should use the interface.
ifinfomsg::if_flags & IFF_LOWER_UP:
 Driver has signaled netif_carrier_on()
ifinfomsg::if_flags & IFF_DORMANT:
 Driver has signaled netif_dormant_on()

TLV IFLA_OPERSTATE

contains RFC2863 state of the interface in numeric representation:

IF_OPER_UNKNOWN (0):
 Interface is in unknown state, neither driver nor userspace has set operational state. Interface must be considered for user data as setting operational state has not been implemented in every driver.
IF_OPER_NOTPRESENT (1):
 Unused in current kernel (notpresent interfaces normally disappear), just a numerical placeholder.
IF_OPER_DOWN (2):
 Interface is unable to transfer data on L1, f.e. ethernet is not plugged or interface is ADMIN down.
IF_OPER_LOWERLAYERDOWN (3):
 Interfaces stacked on an interface that is IF_OPER_DOWN show this

 state (f.e. VLAN).
IF_OPER_TESTING (4):
 Unused in current kernel.
IF_OPER_DORMANT (5):
 Interface is L1 up, but waiting for an external event, f.e. for a
 protocol to establish. (802.1X)
IF_OPER_UP (6):
 Interface is operational up and can be used.

This TLV can also be queried via sysfs.

TLV IFLA_LINKMODE

contains link policy. This is needed for userspace interaction described below.

This TLV can also be queried via sysfs.

3. Kernel driver API

Kernel drivers have access to two flags that map to IFF_LOWER_UP and IFF_DORMANT. These flags can be set from everywhere, even from interrupts. It is guaranteed that only the driver has write access, however, if different layers of the driver manipulate the same flag, the driver has to provide the synchronisation needed.

__LINK_STATE_NOCARRIER, maps to !IFF_LOWER_UP:

The driver uses netif_carrier_on() to clear and netif_carrier_off() to set this flag. On netif_carrier_off(), the scheduler stops sending packets. The name 'carrier' and the inversion are historical, think of it as lower layer.

netif_carrier_ok() can be used to query that bit.

__LINK_STATE_DORMANT, maps to IFF_DORMANT:

Set by the driver to express that the device cannot yet be used because some driver controlled protocol establishment has to complete. Corresponding functions are netif_dormant_on() to set the flag, netif_dormant_off() to clear it and netif_dormant() to query.

On device allocation, networking core sets the flags equivalent to netif_carrier_ok() and !netif_dormant().

Whenever the driver CHANGES one of these flags, a workqueue event is scheduled to translate the flag combination to IFLA_OPERSTATE as follows:

!netif_carrier_ok():
 IF_OPER_LOWERLAYERDOWN if the interface is stacked, IF_OPER_DOWN
 otherwise. Kernel can recognise stacked interfaces because their
 ifindex != iflink.

netif_carrier_ok() && netif_dormant():
 IF_OPER_DORMANT

netif_carrier_ok() && !netif_dormant():
 IF_OPER_UP if userspace interaction is disabled. Otherwise
 IF_OPER_DORMANT with the possibility for userspace to initiate the
 IF_OPER_UP transition afterwards.

4. Setting from userspace

Applications have to use the netlink interface to influence the RFC2863 operational state of an interface. Setting IFLA_LINKMODE to 1 via RTM_SETLINK instructs the kernel that an interface should go to IF_OPER_DORMANT instead of IF_OPER_UP when the combination netif_carrier_ok() && !netif_dormant() is set by the driver. Afterwards, the userspace application can set IFLA_OPERSTATE to IF_OPER_DORMANT or IF_OPER_UP as long as the driver does not set netif_carrier_off() or netif_dormant_on(). Changes made by userspace are multicasted on the netlink group RTMGRP_LINK.

So basically a 802.1X supplicant interacts with the kernel like this:

-subscribe to RTMGRP_LINK
-set IFLA_LINKMODE to 1 via RTM_SETLINK
-query RTM_GETLINK once to get initial state
-if initial flags are not (IFF_LOWER_UP && !IFF_DORMANT), wait until
 netlink multicast signals this state
-do 802.1X, eventually abort if flags go down again
-send RTM_SETLINK to set operstate to IF_OPER_UP if authentication
 succeeds, IF_OPER_DORMANT otherwise
-see how operstate and IFF_RUNNING is echoed via netlink multicast
-set interface back to IF_OPER_DORMANT if 802.1X reauthentication
 fails
-restart if kernel changes IFF_LOWER_UP or IFF_DORMANT flag

if supplicant goes down, bring back IFLA_LINKMODE to 0 and IFLA_OPERSTATE to a sane value.

A routing daemon or dhcp client just needs to care for IFF_RUNNING or waiting for operstate to go IF_OPER_UP/IF_OPER_UNKNOWN before considering the interface / querying a DHCP address.

For technical questions and/or comments please e-mail to Stefan Rompf (stefan at loplof.de).