Softnet Driver Issues

Transmit path guidelines:

1. The ndo_start_xmit method must not return NETDEV_TX_BUSY under any normal circumstances. It is considered a hard error unless there is no way your device can tell ahead of time when it's transmit function will become busy.

   Instead it must maintain the queue properly. For example, for a driver implementing scatter-gather this means:

   ```
   static netdev_tx_t drv_hard_start_xmit(struct sk_buff *skb,
                                          struct net_device *dev)
   {
           struct drv *dp = netdev_priv(dev);

           lock_tx(dp);
           ...
           /* This is a hard error log it. */
           if (TX_BUFFS_AVAIL(dp) <= (skb_shinfo(skb)->nr_frags + 1)) {
                   netif_stop_queue(dev);
                   unlock_tx(dp);
                   printk(KERN_ERR PFX "%s: BUG! Tx Ring full when queue awake!\n",
                          dev->name);
                   return NETDEV_TX_BUSY;
           }

           ... queue packet to card ...
           ... update tx consumer index ...

           if (TX_BUFFS_AVAIL(dp) <= (MAX_SKB_FRAGS + 1))
                   netif_stop_queue(dev);

           ...
           unlock_tx(dp);
           ...
           return NETDEV_TX_OK;
   }
   ```

   And then at the end of your TX reclamation event handling:

   ```
   if (netif_queue_stopped(dp->dev) &&
       TX_BUFFS_AVAIL(dp) > (MAX_SKB_FRAGS + 1))
           netif_wake_queue(dp->dev);
   ```

   For a non-scatter-gather supporting card, the three tests simply become:

   ```
   /* This is a hard error log it. */
   if (TX_BUFFS_AVAIL(dp) <= 0)
   ```

   and:

   ```
   if (TX_BUFFS_AVAIL(dp) == 0)
   ```

   and:

   ```
   if (netif_queue_stopped(dp->dev) &&
       TX_BUFFS_AVAIL(dp) > 0)
           netif_wake_queue(dp->dev);
   ```

2. An ndo_start_xmit method must not modify the shared parts of a cloned SKB.
3. Do not forget that once you return NETDEV_TX_OK from your ndo_start_xmit method, it is your driver's responsibility to free up the SKB and in some finite amount of time.

   For example, this means that it is not allowed for your TX mitigation scheme to let TX packets "hang out" in the TX ring unreclaimed forever if no new TX packets are sent. This error can deadlock sockets waiting for send buffer room to be freed up.

   If you return NETDEV_TX_BUSY from the ndo_start_xmit method, you must not keep any reference to that SKB and you must not attempt to free it up.

Probing guidelines:

1. Any hardware layer address you obtain for your device should be verified. For example, for ethernet check it with linux/etherdevice.h:is_valid_ether_addr()

Close/stop guidelines:

1. After the ndo_stop routine has been called, the hardware must not receive or transmit any data. All in flight packets must be aborted. If necessary, poll or wait for completion of any reset commands.

2. The ndo_stop routine will be called by unregister_netdevice if device is still UP.