Synchronous PPP and Cisco HDLC Programming Guide

Alan Cox

alan@redhat.com

Synchronous PPP and Cisco HDLC Programming Guide

by Alan Cox

Table of Contents

Chapter 1. Introduction

The syncppp drivers in Linux provide a fairly complete implementation of Cisco HDLC and a minimal implementation of PPP. The longer term goal is to switch the PPP layer to the generic PPP interface that is new in Linux 2.3.x. The API should remain unchanged when this is done, but support will then be available for IPX, compression and other PPP features

Chapter 2. Known Bugs And Assumptions

PPP is minimal

The current PPP implementation is very basic, although sufficient for most wan usages.

Cisco HDLC Quirks

Currently we do not end all packets with the correct Cisco multicast or unicast flags. Nothing appears to mind too much but this should be corrected.

Chapter 3. Public Functions Provided

sppp_input

Name

`sppp_input` — receive and process a WAN PPP frame

Synopsis

```
void sppp_input  (struct net_device * dev); struct sk_buff *
skb);
```

Arguments

dev

The device it arrived on

skb

The buffer to process

Description

This can be called directly by cards that do not have timing constraints but is normally called from the network layer after interrupt servicing to process frames queued via `netif_rx`.

We process the options in the card. If the frame is destined for the protocol stacks then it requeues the frame for the upper level protocol. If it is a control from it is processed and discarded here.

sppp_close

Name

`sppp_close` — close down a synchronous PPP or Cisco HDLC link

Synopsis

```
int sppp_close  (struct net_device * dev);
```

Arguments

dev

The network device to drop the link of

Description

This drops the logical interface to the channel. It is not done politely as we assume we will also be dropping DTR. Any timeouts are killed.

sppp_open

Name

`sppp_open` — open a synchronous PPP or Cisco HDLC link

Synopsis

```
int sppp_open  (struct net_device * dev);
```

Arguments

dev

Network device to activate

Description

Close down any existing synchronous session and commence from scratch. In the PPP case this means negotiating LCP/IPCP and friends, while for Cisco HDLC we simply need to start sending keepalives

sppp_reopen

Name

`sppp_reopen` — notify of physical link loss

Synopsis

```
int sppp_reopen  (struct net_device * dev);
```

Arguments

dev

Device that lost the link

Description

This function informs the synchronous protocol code that the underlying link died (for example a carrier drop on X.21)

We increment the magic numbers to ensure that if the other end failed to notice we will correctly start a new session. It happens do to the nature of telco circuits is that you can lose carrier on one endonly.

Having done this we go back to negotiating. This function may be called from an interrupt context.

sppp_change_mtu

Name

`sppp_change_mtu` — Change the link MTU

Synopsis

```
int sppp_change_mtu  (struct net_device * dev); int new_mtu);
```

Arguments

`dev`

Device to change MTU on

`new_mtu`

New MTU

Description

Change the MTU on the link. This can only be called with the link down. It returns an error if the link is up or the mtu is out of range.

sppp_do_ioctl

Name

`sppp_do_ioctl` — Ioctl handler for ppp/hdlc

Synopsis