

PPP and SLIP

When your modem makes funny
noises...

PPP / SLIP



- **PPP**
 - ◆ Where is PPP used
 - ◆ What is the task of LCP
 - ◆ What is the task of NCP
- **SLIP**
 - ◆ Why SLIP died out

Introduction (1)



- **Goal of PPP**
 - ◆ **Convey datagrams over a serial link**
 - ◆ **Both synchronous or asynchronous serial links are supported**
 - ◆ **Both bit or byte oriented transmissions are supported**
- **Basically, PPP consists of**
 - ◆ **One Link Control Protocol (LCP)**
 - ◆ **Several Network Control Protocols (NCPs)**

The Point-to-Point Protocol (PPP) provides a standard method for transporting multi-protocol datagrams over point-to-point links. PPP is comprised of three main components:

1. A method for encapsulating multi-protocol datagrams.
2. A Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection.
3. A family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols.

Introduction (2)



- **HDLC is basis for encapsulation**
 - ♦ Only framing and error detection necessary
 - ♦ Only simple unnumbered information frames (UI)
- **PPP supports full-duplex links only (!)**
- **PPP Frame = Datagram + 2-8 bytes extra header**
 - ♦ Extra header consists of HDLC header and PPP header
- **Byte Stuffing: Data dependent overhead!**

Overhead

Only 8 additional octets are necessary to form the encapsulation when used with the default HDLC framing. In environments where bandwidth is at a premium, the encapsulation and framing may be shortened to 2 or 4 octets.

Byte Stuffing

If the flag byte (126) occurs in the data field it has to be escaped using the escape byte 125, while byte 126 is transmitted as a two byte sequence (125, 94) and the escape byte itself is transmitted as (125, 93).



- **Link Control Protocol (LCP)**
 - ♦ Setup, configure, test and terminate PPP connection
 - ♦ Supports various environments
- **LCP negotiates**
 - ♦ Encapsulation format options
 - ♦ Maximal packet sizes
 - ♦ Identification and authentication of peers (!)
 - ♦ Determination of proper link functionality

In order to be sufficiently versatile to be portable to a wide variety of environments, PPP provides a Link Control Protocol (LCP). The LCP is used to automatically agree upon the encapsulation format options, handle varying limits on sizes of packets, authenticate the identity of its peer on the link, determine when a link is functioning properly and when it is defunct, detect a looped-back link and other common misconfiguration errors, and terminate the link.



- **Network Control Protocols (NCPs)**
 - ◆ **Helper to establish various network protocols**
 - ◆ **IP uses "IPCP"**
- **Typical tasks**
 - ◆ **Assignment and management of IP addresses**
 - ◆ **Compression and authentication**

Point-to-Point links tend to exacerbate many problems with the current family of network protocols. For instance, assignment and management of IP addresses, which is a problem even in LAN environments, is especially difficult over circuit-switched point-to-point links (such as dial-up modem servers). These problems are handled by a family of Network Control Protocols (NCPs), which each manage the specific needs required by their respective network-layer protocols.

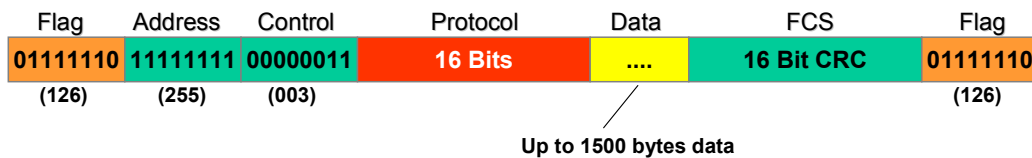
NCPs have been developed for all important network layer protocols such as IP, which uses the IP Control Protocol (IPCP).

There are also NCPs designed to enable compression and authentication.

Data Link Layer: HDLC



- **Address 11111111 means "all stations"**
 - ♦ PPP does not assign individual station addresses
- **Only the control field 00000011 is used**
 - ♦ Unnumbered Information (UI) command
- **Protocol field identifies datagram**
 - ♦ Already part of PPP, not HDLC (!)



(C) Herbert Haas 2005/03/11

7

Protocol: The True PPP Field

The most important field is the protocol field, which has two octets and its value identifies the datagram encapsulated in the Information field of the packet.

PPP Header Compression

If protocol field compression is enabled, the protocol field is reduced from 2 to 1 byte. Since the first two bytes are always constant, that is the address byte (always 255) and the control byte (always 003), PPP also supports address-and-control-field-compression, which omits these bytes.

Byte Stuffing

If the flag byte (126) occurs in the data field it has to be escaped using the escape byte 125, while byte 126 is transmitted as a two byte sequence (125, 94) and the escape byte itself is transmitted as (125, 93).

Protocol Field



0xxx – 3xxx	L3 protocol type
4xxx – 7xxx	L3 protocol type without associated NCPs
8xxx – bxxx	Associated NCPs for protocols in range 0xxx – 3xxx
cxxx – fxxx	LCP, PAP, CHAP, ...

		Important Examples	
0021	IP	c021	Link Control Protocol (LCP)
002b	Novell IPX	c023	Password Auth. Protocol (PAP)
002d	Van Jacobson Compressed TCP/IP	c025	Link Quality Report
002f	Van Jacobson Uncompressed TCP/IP	c223	Challenge Handshake Auth. Protocol (CHAP)
8021	IP-NCP (IPCP)		
802b	IPX-NCP (IPXCP)		

Protocol Field Values

Protocol field values in the "0****" to "3****" range identify the network-layer protocol of specific packets, and values in the "8****" to "b****" range identify packets belonging to the associated Network Control Protocols (NCPs), if any. Protocol field values in the "4****" to "7****" range are used for protocols with low volume traffic which have no associated NCP. Protocol field values in the "c****" to "f****" range identify packets as link-layer Control Protocols (such as LCP).

All these numbers are controlled by the IANA (see RFC-1060).

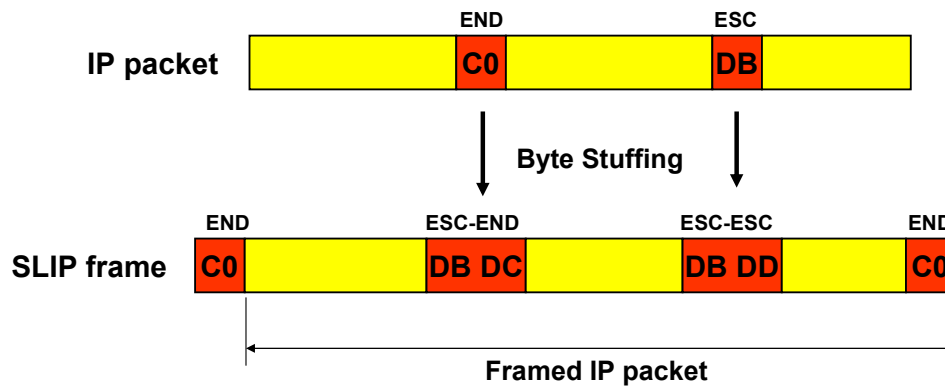
SLIP



- **Serial Line IP (SLIP)**
 - ♦ Asynchronous, character oriented, 8 bit, no parity
 - ♦ Simple layer 2 frame format for serial lines
- **Only provides framing**
- **Only encapsulates IP packets**
- **No flow control with XON/XOFF possible.**
- **Used in earlier "dial In" modem connections.**

SLIP has its origins in the 3COM UNET TCP/IP implementation from the early 1980's. It is merely a packet framing protocol: SLIP defines a sequence of characters that frame IP packets on a serial line, and nothing more. It provides no addressing, packet type identification, error detection/correction or compression mechanisms. Because the protocol does so little, though, it is usually very easy to implement.

SLIP Framing



C0 ... 192
DB ... 219
DC ... 220
DD ... 221

The SLIP frame format is very simple. Actually the frame is only enclosed using a framing byte (C0) at each end. Using byte stuffing this special flag is masked inside the frame.

SLIP Disadvantages



- **IP addresses must be preconfigured**
 - ♦ No dynamic assignment
- **No protocol (type) field**
 - ♦ Only defined to transport IP packets
- **No Frame Check Sequence (FCS)**
 - ♦ Higher layers must care!
 - ♦ But higher layers just use checksums (CRC would be better)
- **Inconstant overhead**
 - ♦ Depends on data pattern!

There are several features that many users would like SLIP to provide which it doesn't. SLIP is just a very simple protocol designed quite a long time ago when these problems were not really important issues. The following are commonly perceived shortcomings in the existing SLIP protocol.

Both computers in a SLIP link need to know each other's IP addresses for routing purposes. Also, when using SLIP for hosts to dial-up a router, the addressing scheme may be quite dynamic and the router may need to inform the dialing host of the host's IP address. SLIP currently provides no mechanism for hosts to communicate addressing information over a SLIP connection.

SLIP has no type field. Thus, only one protocol can be run over a SLIP connection, so in a configuration of two DEC computers running both TCP/IP and DECnet, there is no hope of having TCP/IP and DECnet share one serial line between them while using SLIP. Error detection is not absolutely necessary at the SLIP level because any IP application should detect damaged packets.

The protocol overhead introduced by SLIP is data dependent. In worst cases the overhead is very high. Consider the case of a transmitted image which has very often the color value of 219.

Compressed SLIP



- **CSLIP utilizes Van Jacobson header compression**
 - ◆ 40 byte TCP/IP header to 3-5 bytes
 - ◆ Up to 16 CSLIP connections per link
- **Useful for small TCP/IP packets**

Compressed SLIP (CSLIP) is a common enhancement of today's SLIP implementation. CSLIP uses Van Jacobson header compression algorithm, which relies on the fact that most of the IP and TCP header remains constant during a connection. The current state of a TCP/IP connection is maintained by each endpoint of a link to enable compression and reconstruction of the TCP/IP header.

Up to 16 TCP connections can be handled on the same link by CSLIP.