Network Working Group
Request for Comments: 2823
Category: Experimental

J. Carlson
Sun Microsystems, Inc.
P. Langner
Lucent Technologies Microelectronics Group
E. Hernandez-Valencia
J. Manchester
Lucent Technologies

May 2000

PPP over Simple Data Link (SDL) using SONET/SDH with ATM-like framing

Status of this Memo

Copyright Notice

Abstract

The Point-to-Point Protocol (PPP) [1] provides a standard method for transporting multi-protocol datagrams over point-to-point links, and RFCs 1662 [2] and 2615 [3] provide a means to carry PPP over Synchronous Optical Network (SONET) [4] and Synchronous Digital Hierarchy (SDH) [5] circuits. This document extends these standards to include a new encapsulation for PPP called Simple Data Link (SDL) [6]. SDL provides a very low overhead alternative to HDLC-like encapsulation, and can also be used on SONET/SDH links.

Applicability

This specification is intended for those implementations that use PPP over high speed point-to-point circuits, both with so-called "dark fiber" and over public telecommunications networks. Because this enhanced PPP encapsulation has very low overhead and good hardware scaling characteristics, it is anticipated that significantly higher throughput can be attained when compared to other possible SONET/SDH payload mappings, and at a significantly lower cost for line termination equipment.

SDL is defined over other media types and for other data link protocols, but this specification covers only the use of PPP over SDL on SONET/SDH.

The use of SDL requires the presentation of packet length information in the SDL header. Thus, hardware implementing SDL must have access to the packet length when generating the header, and where a router's input link does not have this information (that is, for non-SDL input links), the router may be required to buffer the entire packet before transmission. "Worm-hole" routing is thus at least problematic with SDL, unless the input links are also SDL. This, however, does not appear to be a great disadvantage on modern routers due to the general requirement of length information in other parts of the system, notably in queuing and congestion control strategies such as Weighted Fair Queuing [7] and Random Early Detect [8].

This document is not a replacement for the existing HDLC-like framing mandated by RFC 2615 [3]. Instead, the authors intend to gain implementation experience with this technique for operational and performance evaluation purposes, and would like to hear from others either considering or using the protocol as described in this document. Please see Section 14 of this document for contact information.

Table of Contents

1. Introduction

The Path Signal Label (SONET/SDH overhead byte named C2; referred to as PSL in this document) is intended to indicate the type of data carried on the path. This data, in turn, is referred to as the SONET Synchronous Payload Envelope (SPE) or SDH Administrative Unit Group (AUG). The experimental PSL value of decimal 207 (CF hex) is currently [3] used to indicate that the SPE contains PPP framed using RFC 1662 Octet Synchronous (O-S) framing and transmission without scrambling, and the value 22 (16 hex) is used to indicated PPP framed using O-S framing and transmission with ATM-style X^43+1 scrambling.

This document describes a method to enable the use of SDL framing for PPP over SONET/SDH, and describes the framing technique and requirements for PPP. While O-S framing on SONET/SDH has a fixed seven octet overhead per frame plus a worst-case overhead of 100% of all data octets transmitted, SDL has a fixed eight octet per frame overhead with zero data overhead. Unlike O-S framing, SDL also provides positive indication of link synchronization.

Note: This document describes two new SONET/SDH Path Signal Label (PSL) values; 23 (17 hex) for SDL with the proposed self synchronous scrambler and 25 (19 hex) for SDL with the proposed set-reset scrambler. These values have been allocated by ANSI T1X1.5 and ITU-T SG-15 for use with SDL over SONET and SDH, and will appear in subsequent updates of T1.105 (Table 8) and Recommendation G.707 (Table 7).

2. Compliance

In this document, the words that are used to define the significance of each particular requirement are capitalized.

These words are:

* "MUST"

  This word means that the item is an absolute requirement of the specification.

* "MUST NOT"

  This phrase means that the item is an absolute prohibition of the specification.

* "SHOULD"

  This word means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

* "SHOULD NOT"

  This phrase means that there may exist valid reasons in particular circumstances to apply this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

* "MAY"

  This word means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

An implementation is not compliant if it fails to satisfy one or more of the MUST or MUST NOT requirements for this protocol. An implementation that satisfies all of the MUST, MUST NOT, SHOULD, and SHOULD NOT requirements for this protocol is said to be "unconditionally compliant". One that satisfies all the MUST and MUST NOT requirements but not all the SHOULD or SHOULD NOT requirements is said to be "conditionally compliant".

3. Physical Layer Requirements

PPP treats SONET/SDH transport as octet-oriented synchronous links. No provision is made to transmit partial octets. Also, SONET/SDH links are full-duplex by definition.

3.1. Payload Types

Only synchronous payloads STS-1 and higher are considered in this document. Lower speed synchronous, such as VT1.5-SPE/VC-11, and plesiochronous payload mappings, such as T1 and T3, are defined for SONET/SDH and for the SDL algorithm itself, but, since HDLC-like framing is defined for PPP on those media, PPP over SDL is not defined.

SDL is separately defined as a PPP transport for use on raw fiber without SONET/SDH framing for use as an alternative to bit-synchronous HDLC. Please see the separate work-in-progress for details.

3.2. Control Signals

The PPP over SONET/SDH mapping allows the use of the PSL as a control signal. Not all equipment, however, is capable of setting or detecting this value, and any use must take this into account. Equipment employing only SDL MUST be capable of transmitting PSL with value 23, and MAY also be capable of transmitting PSL with value 25, but need not be capable of detecting the peer's value or capable of changing its own value.

There are two methods to enable SDL, an LCP-negotiated method and a prior-arrangement method. The former allows for easier configuration and compatibility with existing equipment, while the latter allows general use with separate SONET/SDH transmission equipment with PSL limitations. Both types of implementations will freely interoperate given the procedures below.

LCP-negotiated systems MUST be capable of changing their transmitted PSL value and detecting the peer's value. Equipment without these features MUST NOT support LCP negotiation of SDL.

When SDL is negotiated by LCP, LCP negotiation MUST be started with the PSL value initially set to 22 or 207 and the corresponding non-SDL O-S PPP encapsulation MUST be used. The SDL LCP option is then placed in the LCP Configure-Request messages transmitted. On reception of LCP Configure-Request with an SDL LCP option or when the peer's transmitted PSL value is received as 23 (or 25), the implementation MUST shut down LCP by sending a Down event to its state machine, then switch its transmitted PSL value to 23 (or 25), switch encapsulation mode to SDL, wait for SDL synchronization, and then restart LCP by sending an Up event into LCP. Otherwise, if the peer does not transmit PSL value 23 (or 25) and it does not include the SDL LCP option in its LCP Configure-Request messages, then operation using non-SDL O-S PPP encapsulation continues. If the received PSL value subsequently received reverts from 23 (or 25) to any other value, then this is treated as a Down event into the LCP state machine, and an Up event MUST be generated if the new value is recognized as a valid PPP framing mode.

When SDL is enabled by prior arrangement, the PSL SHOULD be transmitted as 23 (or 25). Any other value may also be used by prior external arrangement with the peer, although the values 22 and 207 are discouraged. (Such use is enforced by an administrator, and is outside the scope of this specification.) When SDL is enabled by prior arrangement, the SDL LCP option SHOULD NOT be negotiated by the peers.

An implementation-specific configuration option SHOULD exist to enable the use of prior-arrangement versus LCP-negotiated modes. This option SHOULD be presented to an administrator, and SHOULD default to LCP-negotiated if the hardware permits. Otherwise, if the hardware implementation precludes non-SDL modes of operation, then it MUST default to prior-arrangement mode.

The LCP-negotiated method of operation is compatible with the current version of G.783 [12]. This method may not be compatible, however, with some non-intrusive SDH path monitoring equipment based on obsolete versions of G.783. The change in PSL value indicated by the LCP negotiation method will cause this equipment to declare an alarm condition on the path. For this reason, the prior-arrangement method MUST be used on any SDH network that is using such monitoring equipment.

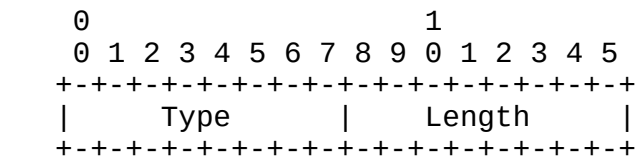
3.3. Synchronization Modes

Unlike O-S encapsulation, SDL provides a positive indication that it has achieved synchronization with the peer. An SDL PPP implementation MUST provide a means to temporarily suspend PPP data transmission (both user data and negotiation traffic) if synchronization loss is detected. An SDL PPP implementation SHOULD also provide a configurable timer that is started when SDL is initialized and restarted on the loss of synchronization, and is terminated when link synchronization is achieved. If this timer expires, implementation-dependent action should be taken to report the hardware failure.

3.4. Simple-Data-Link LCP Option

A new LCP Configuration Option is used to request Simple Data Link (SDL) [6] operation for the PPP link.

A summary of the Simple-Data-Link Configuration Option format for the Link Control Protocol (LCP) is shown below. The fields are transmitted from left to right.



Type

29

Length

2

This option is used only as a hint to the peer that SDL over SONET/SDH operation is preferred by the sender. If the current encapsulation mode is not SDL, then the only appropriate response to reception of this option by an SDL speaker is to then switch the encapsulation mode to SDL (as detailed in the section above) and restart LCP. Non SDL-speakers SHOULD instead send LCP Configure-Reject for the option.

If either LCP Configure-Nak or LCP Configure-Reject is received for this option, then the next transmitted LCP Configure-Request MUST NOT include this option. If LCP Configure-Ack with this option is received, it MUST NOT be treated as a request to switch into SDL mode. If the received LCP Configure-Request message does not contain an SDL LCP option, an implementation MUST NOT send an unsolicited Configure-Nak for the option.

(An implementation of SDL that is already in SDL framing mode and receives this option in an LCP Configure-Request message MAY, both for clarity and for convergence reasons, elect to send LCP Configure-Ack. It MUST NOT restart LCP nor change framing modes in this case.)

3.5. Framing

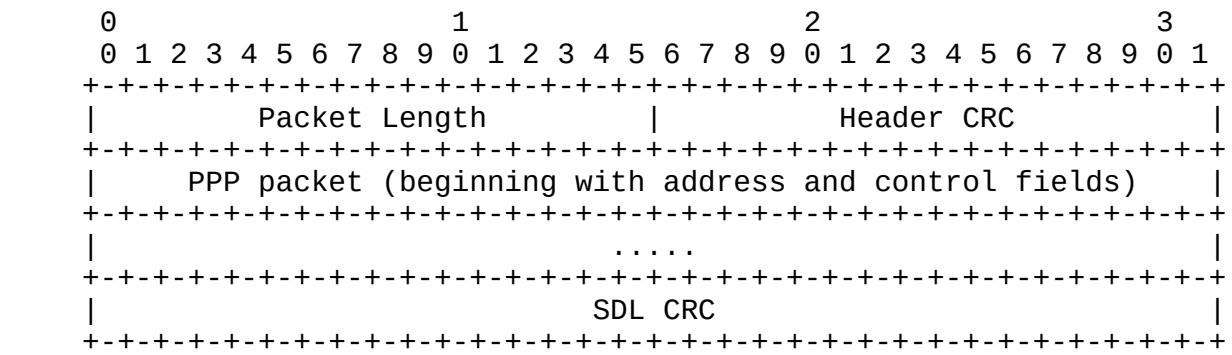
The PPP frames are located by row within the SPE payload. Because frames are variable in length, the frames are allowed to cross SPE boundaries. Bytes marked as "overhead" or "fixed stuff" in SONET/SDH documentation for concatenated streams are not used as payload bytes.

With reference to the Lucent SDL specification [6] when SDL framing for PPP is employed, the SDL "Datagram Offset" feature is set to the value 4. This corresponds to the fixed overhead value 4 in the

description below. The "A" and "B" messages are never used. These optional features of SDL are not described in this document, but are rather described in Lucent's SDL specification.

Fixing the Datagram Offset value described in the Lucent documentation to 4 allows a PPP MRU/MTU up to 65536 using SDL.

SDL framing is in general accomplished by the use of a four octet header on the packet. This fixed-length header allows the use of a simple framer to detect synchronization as described in section 3.7. For use with PPP, this fixed-length header precedes each PPP/HDLC packet as follows:



The four octet length header is DC balanced by exclusive-OR (also known as "modulo 2 addition") with the hex value B6AB31E0. This is the maximum transition, minimum sidelobe, Barker-like sequence of length 32. No other scrambling is done on the header itself.

Packet Length is an unsigned 16 bit number in network byte order. Unlike the PPP FCS, the Header CRC is a CRC-16 generated with initial value zero and transmitted in network byte order. The PPP packet is scrambled, begins with the address and control fields, and may be any integral octet length (i.e., it is not padded unless the Self Describing Padding option is used). The Packet CRC is also scrambled, and has a mode-dependent length (described below), and is located only on an octet boundary; no alignment of this field may be assumed.

When the Packet Length value is 4 or greater, the distance in octets between one message header and the next in SDL is the sum of 8 plus the Packet Length field. The value 8 represents a fixed overhead of 4 octets plus the fixed length of the Packet CRC field. When the Packet Length is 0, the distance to the next header is 4 octets. This is the idle fill header. When the Packet Length is 1 to 3, the

distance to the next header is 12 octets. These headers are used for special SDL messages used only with optional scrambling and management modes. See section 5 for details of the messages.

General SDL, like PPP, allows the use of no CRC, ITU-T CRC-16, or ITU-T CRC-32 for the packet data. However, because the Packet Length field does not include the CRC length, synchronization cannot be maintained if the CRC type is changed per RFC 1570 [9], because frame-to-frame distance is, as described above, calculated including the CRC length. Thus, this PPP over SDL specification fixes the CRC type to CRC-32 (four octets), and all SDL implementations MUST reject any LCP FCS Alternatives Option [9] requested by the peer when in SDL mode.

PPP over SDL implementations MAY allow a configuration option to set different CRC types for use by prior arrangement. Any such configurable option MUST default to CRC-32, and MUST NOT include LCP negotiation of FCS Alternatives.

Setting the SDL Datagram Offset value to 4 accounts for the 4 octet SDL header overhead. With the SDL Datagram Offset set to 4, the value placed in the Packet Length field is exactly the length in octets of the PPP frame itself, including the address and control fields but not including the CRC field (the RFC 1662 PPP FCS field is not used with SDL). Note again that the Datagram Offset is just an arithmetic value; it does not occupy bits in the message itself.

Because Packet Lengths below 4 are reserved, the Packet Length MUST be 4 or greater for any legal PPP packet. PPP packets with fewer octets, which are not possible without address/control or protocol field compression, MUST be padded to length 4 for SDL.

Inter-packet time fill is accomplished by sending the four octet length header with the Packet Length set to zero. No provision is made for intra-packet time fill.

By default, an independent, self-synchronous x^43+1 scrambler is used on the data portion of the message including the 32 bit CRC. This is done in exactly the same manner as with the ATM x^43+1 scrambler on an ATM channel. The scrambler is not clocked when SDL header bits are transmitted. Thus, the data scrambling MAY be implemented in an entirely independent manner from the SDL framing, and the data stream may be prescrambled before insertion of SDL framing marks.

Optionally, by prior arrangement, SDL links MAY use a set-reset scrambler as described in section 6. If this option is provided, it MUST be configurable by the administrator, and the option MUST default to the self-synchronous scrambler.

3.6. Framing Example

To help clarify this structure, the following example may be helpful. First we have an LCP Configure-Request message that we wish to transmit over SDL:

FF 03 C0 21 01 01 00 04

Next, we create an SDL header for the length of this packet (8 octets), a header CRC, and an SDL CRC.

00 08 81 08 FF 03 C0 21 01 01 00 04 D1 F5 21 5E

Finally, we DC-balance the header with the barker-like sequence:

B6 A3 B0 E8 FF 03 C0 21 01 01 00 04 D1 F5 21 5E

Note that the final length of the message is 8 (original message length) plus 4 (fixed datagram offset value) plus 4 (fixed CRC length), or 16 octets.

3.7. Synchronization Procedure

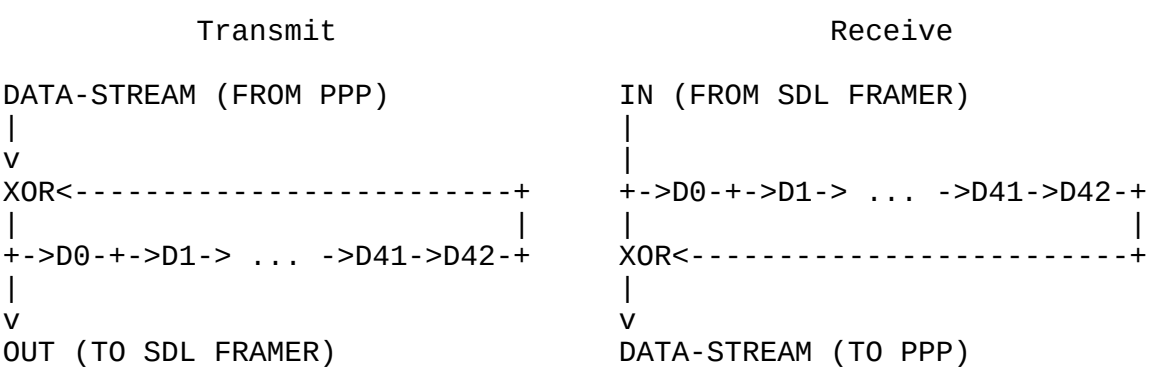
The link synchronization procedure is similar to the I.432 section 4.5.1.1 ATM HEC delineation procedure [10], except that the SDL messages are variable length. The machine starts in HUNT state until a four octet sequence in the data stream with a valid CRC-16 is found. (Note that the CRC-16 single-bit error correction technique described in section 3.10 is not employed until the machine is in in SYNCH state. The header must have no bit errors in order to leave HUNT state.) Such a valid sequence is a candidate SDL header. On finding the valid sequence, the machine enters PRESYNCH state. Any one invalid SDL header in PRESYNCH state returns the link to HUNT state.

If a second valid SDL header is seen after entering PRESYNCH state, then the link enters SYNCH state and PPP transmission is enabled. If an invalid SDL header is detected, then the link is returned to HUNT state without enabling PPP transmission.

Once the link enters SYNCH state, the SDL header single bit error correction logic is enabled (see section 3.10). Any unrecoverable header CRC error returns the link to HUNT state, disables PPP transmission, and disables the error correction logic.

3.8. Scrambler Operation

The transmit and receive scramblers are shift registers with 43 stages that MAY be initialized to all-ones when the link is initialized. Synchronization is maintained by the data itself.



Each XOR is an exclusive-or gate; also known as a modulo-2 adder. Each Dn block is a D-type flip-flop clocked on the appropriate data clock.

The scrambler is clocked once after transmission or reception of each bit of payload and before the next bit is applied as input. Bits within an octet are, per SONET/SDH practice, transmitted and received MSB-first.

3.9. CRC Generation

The CRC-16 and CRC-32 generator polynomials used by SDL are the ITU-T polynomials [11]. These are:

$$x^{16}+x^{12}+x^{5}+1$$

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$$

The SDL Header CRC and the CRC-16 used for each of the three special messages (scrambler state, message A, and message B; see section 5) are all generated using an initial remainder value of 0000 hex.

The optional CRC-16 on the payload data (this mode is not used with PPP over SDL except by prior arrangement) uses the initial remainder value of FFFF hex for calculation and the bits are complemented before transmission. The final CRC remainder, however, is transmitted in network byte order, unlike the regular PPP FCS. If the CRC-16 algorithm is run over all of the octets including the appended CRC itself, then the remainder value on intact packets will

always be E2F0 hex. Alternatively, an implementation may stop CRC calculation before processing the appended CRC itself, and do a direct comparison.

The CRC-32 on the payload data (used for PPP over SDL) uses the initial remainder value of FFFFFFFF hex for calculation and the bits are complemented before transmission. The CRC, however, is transmitted in network byte order, most significant bit first, unlike the optional PPP 32 bit FCS, which is transmitted in reverse order. The remainder value on intact packets when the appended CRC value is included in the calculation is 38FB2284.

C code to generate these CRCs is found in section 8.1.

3.10. Error Correction

The error correction technique is based on the use of a Galois number field, as with the ATM HEC correction. In a Galois number field, f(a+b) = f(a) + f(b). Since the CRC-16 used for SDL forms such a field, we can state that CRC(message+error) = CRC(message) + CRC(error). Since the CRC-16 remainder of a properly formed message is always zero, this means that, for the N distinct "error" strings corresponding to a single bit error, there are N distinct CRC(error) values, where N is the number of bits in the message.

A table look-up is thus applied to the CRC-16 residue after calculation over the four octet SDL header to correct bit errors in the header and to detect multiple bit errors. For the optional set-reset scrambler, a table look-up is similarly applied to the CRC-16 residue after calculation over the eight octet scrambler state message to correct bit errors and to detect multiple bit errors. (This second correction is also used for the special SDL A and B messages, which are not used for PPP over SDL.)

Note: No error correction is performed for the payload.

Note: This error correction technique is used only when the link has entered SYNCH state. While in HUNT or PRESYNCH state, error correction should not be performed, and only messages with syndrome 0000 are accepted. If the calculated syndrome does not appear in this table, then an unrecoverable error has occurred. Any such error in the SDL header will return the link to HUNT state.

Since the CRC calculation is started with zero, the two tables can be merged. The four octet table is merely the last 32 entries of the eight octet table.

Eight octet (64 bit) single bit error syndrome table (in hexadecimal):

```
FD81 F6D0 7B68 3DB4 1EDA 0F6D 8FA6 47D3
ABF9 DDEC 6EF6 377B 93AD C1C6 60E3 B861
D420 6A10 3508 1A84 0D42 06A1 8B40 45A0
22D0 1168 08B4 045A 022D 8906 4483 AA51
DD38 6E9C 374E 1BA7 85C3 CAF1 ED68 76B4
3B5A 1DAD 86C6 4363 A9A1 DCC0 6E60 3730
1B98 0DCC 06E6 0373 89A9 CCC4 6662 3331
9188 48C4 2462 1231 8108 4084 2042 1021
```

Thus, if the syndrome 6EF6 is seen on an eight octet message, then the third bit (hex 20) of the second octet is in error. Similarly, if 48C4 is seen on an eight octet message, then the second bit (hex 40) in the eighth octet is in error. For a four octet message, the same two syndromes would indicate a multiple bit error for 6EF6, and a single bit error in the second bit of the fourth octet for 48C4.

Note that eight octet messages are used only for the optional set-reset scrambling mode, described in section 6.

Corresponding C code to generate this table is found in section 8.2.

4. Performance Analysis

There are five general statistics that are important for framing algorithms. These are:

MTTF	Mean time to frame
MTTS	Mean time to synchronization
PFF	Probability of false frame
PFS	Probability of false synchronization
PLF	Probability of loss of frame

The following sections summarize each of these statistics for SDL. Details and mathematic development can be found in the Lucent SDL documentation [6].

4.1. Mean Time To Frame (MTTF)

This metric measures the amount of time required to establish correct framing in the input data. This may be measured in any convenient units, such as seconds or bytes. For SDL, the relevant measurement is in packets, since fragments of packets are not useful.

In order to calculate MTTF, we must first determine how often the frame detection state machine is "unavailable" because it failed to detect the next incoming SDL frame in the data stream.

Since the probability of a false header detection using CRC-16 in random data is 2^-16 and this rate is large compared to the allowable packet size, it is worthwhile to run multiple parallel frame-detection state machines. Each machine starts with a different candidate framing point in order to reduce the probability of falsely detecting user data as a valid frame header.

The results for this calculation, given maximal 64KB packets and slightly larger than Internet average 354 byte packets, are:

Number of Framers	Unavailability 64KB packets	Unavailability 354 byte pkts
1	3.679E-1	5.373E-3
2	3.083E-2	1.710E-6
3	2.965E-3	9.712E-10
4	2.532E-4	4.653E-13

Using these values, MTTF can be calculated as a function of the Bit Error Rate (BER). These plots show a characteristically flat region for all BERs up to a knee, beyond which the begins to rise sharply. In all cases, this knee point has been found to occur at a BER of approximately 1E-4, which is several orders of magnitude above that observed on existing SONET/SDH links. The flat rate values are summarized as:

Number of Framers	Flat region 64KB packets	Flat region 354 bytes
1	3.58	1.52
2	1.595	1.5
3	1.52	1.5
4	1.5	1.5

Thus, for common packet sizes in an implementation with two parallel framers using links with a BER of 1E-4 or better, the MTTF is approximately 1.5 packets. This is also the optimal time, since it represents initiating framing at an average point half-way into one packet, and achieving good framing after seeing exactly one correctly framed packet.

4.2. Mean Time To Synchronization (MTTS)

The MTTS for SDL with a self-synchronous scrambler is the same as the MTTF, or 1.5 packets.

The MTTS for SDL using the optional set-reset scrambler is one half of the scrambling state transmission interval (in packets) plus the MTTF. For insertion at the default rate of one per eight packets, the MTTS is 5.5 packets.

(The probability of receiving a bad scrambling state transmission should also be included in this calculation. The probability of random corruption of this short message is shown in the SDL document [6] to be small enough that it can be neglected for this calculation.)

4.3. Probability of False Frame (PFF)

The PFF is 2.328E-10 (2^-32), since false framing requires two consecutive headers with falsely correct CRC-16.

4.4. Probability of False Synchronization (PFS)

The PFS for SDL with the self-synchronous scrambler is the same as the PFF, or 2.328E-10 (2^-32).

The PFS for SDL with the set-reset scrambler is 5.421E-20 (2^-64), and is calculated as the PFF above multiplied by the probability of a falsely detected scrambler state message, which itself contains two independent CRC-16 calculations.

4.5. Probability of Loss of Frame (PLF)

The PLF is a function of the BER, and for SDL is approximately the square of the BER multiplied by 500, which is the probability of two or more bit errors occurring within the 32 bit SDL header. Thus, at a BER of 1E-5, the PLF is 5E-8.

5. The Special Messages

When the SDL Packet Length field has any value between 0000 and 0003, the message following the header has a special, pre-defined length. The 0 value is a time-fill on an idle link, and no other data follows. The next octet on the link is the first octet of the next SDL header.

The values 1 through 3 are defined in the following subsections. These special messages each consist of a six octet data portion followed by another CRC-16 over that data portion, as with the SDL header, and this CRC is used for single bit error correction.

5.1. Scrambler State

The special value of 1 for Packet Length is reserved to transfer the scrambler state from the transmitter to the receiver for the optional set-reset scrambler. In this case, the SDL header is followed by six octets (48 bits) of scrambler state. Neither the scrambler state nor the CRC are scrambled.

5.2. A/B Message

The special values of 2 and 3 for Packet Length are reserved for "A" and "B" messages, which are also six octets in length followed by two octets of CRC-16. Each of these eight octets are scrambled. No use for these messages with PPP SDL is defined. These messages are reserved for use by link maintenance protocols, in a manner analogous to ATM's OAM cells.

6. The Set-Reset Scrambler Option

PPP over SDL uses a self-synchronous scrambler. SDL implementations MAY also employ a set-reset scrambler to avoid some of the possible inherent problems with self-synchronous scramblers.

6.1. The Killer Packet Problem

Scrambling in general solves two problems. First, SONET and SDH interfaces require a minimum density of bit transitions in order to maintain hardware clock recovery. Since data streams frequently contain long runs of all zeros or all ones, scrambling the bits using a pseudo-random number sequence breaks up these patters. Second, all link-layer synchronization mechanisms rely on detecting long-range patterns in the received data to detect framing.

Self-synchronous scramblers are an easy way to partially avoid these problems. One problem that is inherent with self-synchronous, however, is that long user packets from malicious sites can make use of the known properties of these scramblers to inject either long strings of zeros or other synchronization-destroying patterns into the link. For public networks, where the data presented to the network is usually multiplexed (interleaved) with multiple unrelated streams, the clocking problem does not pose a significant threat to the public network. It does, however, pose a threat to the PPP-speaking device, and it poses a threat to long lines that are unchannelized.

Such carefully constructed packets are called "killer packets".

6.2. SDL Set-Reset Scrambler

An alternative to the self-synchronous scrambler is the externally synchronized or "set-reset" scrambler. This is a free-running scrambler that is not affected by the patterns in the user data, and therefore minimizes the possibility that a malicious user could present data to the network that mimics an undesirable data pattern.

The option set-reset scrambler defined for SDL is an x^48+x^28+x^27+x+1 independent scrambler initialized to all ones when the link enters PRESYNCH state and reinitialized if the value ever becomes all zero bits. As with the self-synchronous scrambler, all octets in the PPP packet data following the SDL header through the final packet CRC are scrambled.

This mode MAY be detected automatically. If a scrambler state message is received (as described in the following section), an SDL implementation that includes the set-reset scrambler option may switch from self-synchronous into set-reset mode automatically. An SDL implementation that does not include the set-reset scrambler MUST NOT send scrambler state messages.

6.3. SDL Scrambler Synchronization

As described in the previous section, the special value of 1 for Packet Length is reserved to transfer the scrambler state from the transmitter to the receiver. In this case, the SDL header is followed by six octets (48 bits) of scrambler state plus two octets of CRC-16 over the scrambler state. None of these eight octets are scrambled.

SDL synchronization consists of two components, link and scrambler synchronization. Both must be completed before PPP data flows on the link.

If a valid SDL header is seen in PRESYNCH state, then the link enters SYNCH state, and the scrambler synchronization sequence is started. If an invalid SDL header is detected, then the link is returned to HUNT state, and PPP transmission is suspended.

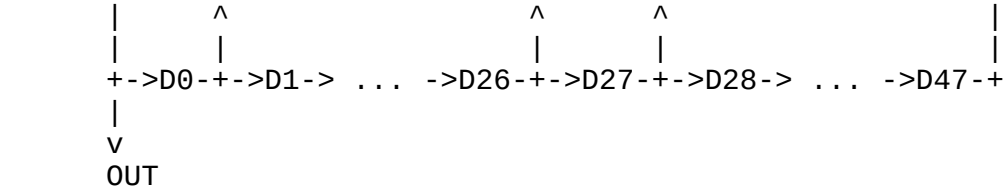
When scrambler synchronization is started, a scrambler state message is sent (Packet Length set to 1 and six octets of scrambler state in network byte order follow the SDL header). When a scrambler synchronization message is received from the peer, PPP transmission is enabled.

Scrambler state messages are periodically transmitted to keep the peers in synchronization. A period of once per eight transmitted packets is suggested, and it SHOULD be configurable. Excessive packet CRC errors detected indicates an extended loss of synchronization and should trigger link resynchronization.

On reception of a scrambler state message, an SDL implementation MUST compare the received 48 bits of state with the receiver's scrambler state. If any of these bits differ, then a synchronization slip error is declared. After such an error, the next valid scrambler state message received MUST be loaded into the receiver's scrambler, and the error condition is then cleared.

6.4. SDL Scrambler Operation

The transmit and receive scramblers are shift registers with 48 stages that are initialized to all-ones when the link is initialized. Each is refilled with all one bits if the value in the shift register ever becomes all zeros. This scrambler is not reset at the beginning of each frame, as is the SONET/SDH X^7+X^6+1 scrambler, nor is it modified by the transmitted data, as is the ATM self-synchronous scrambler. Instead it is kept in synchronization using special SDL messages.



Each XOR is an exclusive-or gate; also known as a modulo-2 adder. Each Dn block is a D-type flip-flop clocked on the appropriate data clock.

The scrambler is clocked once after transmission of each bit of SDL data, whether or not the transmitted bit is scrambled. When scrambling is enabled for a given octet, the OUT bit is exclusive-ored with the raw data bit to produce the transmitted bit. Bits within an octet are transmitted MSB-first.

Reception of scrambled data is identical to transmission. Each received bit is exclusive-ored with the output of the separate receive data scrambler.

To generate a scrambler state message, the contents of D47 through D0 are snapshot at the point where the first scrambler state bit is sent. D47 is transmitted as the first bit of the output. The first octet transmitted contains D47 through D40, the second octet D39 through D32, and the sixth octet D7 through D0.

The receiver of a scrambler state message MUST first run the CRC-16 check and correct algorithm over this message. If the CRC-16 message check detects multiple bit errors, then the message is dropped and is not processed further.

Otherwise, it then should compare the contents of the entire receive scrambler state D47:D0 with the corrected message. (By pipelining the receiver with multiple clock stages between SDL Header error-correction block and the descrambling block, the receive descrambler will be on the correct clock boundary when the message arrives at the descrambler. This means that the decoded scrambler state can be treated as immediately available at the beginning of the D47 clock cycle into the receive scrambler.)

If any of the received scrambler state bits is different from the corresponding shift register bit, then a soft error flag is set. If the flag was already set when this occurs, then a synchronization slip error is declared. This error SHOULD be counted and reported through implementation-defined network management procedures. When the receiver has this soft error flag set, any scrambler state message that passes the CRC-16 message check without multiple bit errors is clocked directly into the receiver’s state register after the comparison is done, and the soft error flag is then cleared. Otherwise, while uncorrectable scrambler state messages are received, the soft error flag state is maintained.

(The intent of this mechanism is to reduce the likelihood that a falsely corrected scrambler state message with multiple bit errors can corrupt the running scrambler state.)

7. Configuration Details

7.1. Default LCP Configuration

The LCP synchronous configuration defaults apply to SONET/SDH links.

The following Configuration Options are recommended:

Magic Number
No Address and Control Field Compression
No Protocol Field Compression
No FCS alternatives (32-bit FCS default)

This configuration means that PPP over SDL generally presents a 32-bit aligned datagram to the network layer. With the address, control, and protocol field intact, the PPP overhead on each packet is four octets. If the SDL framer presents the SDL packet header to the PPP input handling in order to communicate the packet length (the Lucent implementation does not do this, but other hardware implementations may), this header is also four octets, and alignment is preserved.

7.2. Modification of the Standard Frame Format

Since SDL does take the place of HDLC as a transport for PPP, it is at least tempting to remove the HDLC-derived overhead. This is not done for PPP over SDL in order to preserve the message alignment and to allow for the future possibility interworking with other services (e.g., Frame Relay).

By prior external arrangement or via LCP negotiation, any two SDL implementations MAY agree to omit the address and control fields or implement protocol field compression on a link. Such use is not described by this document and MUST NOT be the default on any SDL implementation.

8. Implementation Details

8.1. CRC Generation

The following unoptimized code generates proper CRC-16 and CRC-32 values for SDL messages. Note that the polynomial bits are numbered in big-endian order for SDL CRCs; bit 0 is the MSB.

```
typedef unsigned char u8;
typedef unsigned short u16;
typedef unsigned long u32;

#define POLY16  0x1021
#define POLY32  0x04C11DB7

u16
crc16(u16 crcval, u8 cval)
{
    int i;

    crcval ^= cval << 8;
    for (i = 8; i--; )
        crcval = crcval & 0x8000 ? (crcval << 1) ^ POLY16 :
            crcval << 1;
    return crcval;
```

```
}

u32
crc32(u32 crcval, u8 cval)
{
    int i;

    crcval ^= cval << 24;
    for (i = 8; i--; )
        crcval = crcval & 0x80000000 ? (crcval << 1) ^ POLY32 :
            crcval << 1;
    return crcval;
}

u16
crc16_special(u8 *buffer, int len)
{
    u16 crc;

    crc = 0;
    while (--len >= 0)
        crc = crc16(crc,*buffer++);
    return crc;
}

u16
crc16_payload(u8 *buffer, int len)
{
    u16 crc;

    crc = 0xFFFF;
    while (--len >= 0)
        crc = crc16(crc,*buffer++);
    return crc ^ 0xFFFF;
}

u32
crc32_payload(u8 *buffer, int len)
{
    u32 crc;

    crc = 0xFFFFFFFFul;
    while (--len >= 0)
        crc = crc32(crc,*buffer++);
    return crc ^ 0xFFFFFFFFul;
}
```

```
void
make_sdl_header(int packet_length, u8 *buffer)
{
    u16 crc;

    buffer[0] = (packet_length >> 8) & 0xFF;
    buffer[1] = packet_length & 0xFF;
    crc = crc16_special(buffer,2);
    buffer[0] ^= 0xB6;
    buffer[1] ^= 0xAB;
    buffer[2] = ((crc >> 8) & 0xFF) ^ 0x31;
    buffer[3] = (crc & 0xFF) ^ 0xE0;
}
```

8.2. Error Correction Tables

To generate the error correction table, the following implementation may be used. It creates a table called sdl_error_position, which is indexed on CRC residue value. The tables can be used to determine if no error exists (table entry is equal to FE hex), one correctable error exists (table entry is zero-based index to errored bit with MSB of first octet being 0), or more than one error exists, and error is uncorrectable (table entry is FF hex). To use for eight octet messages, the bit index from this table is used directly. To use for four octet messages, the index is treated as an unrecoverable error if it is below 32, and as bit index plus 32 if it is above 32.

The program also prints out the error syndrome table shown in section 3.10. This may be used as part of a "switch" statement in a hardware implementation.

```
u8 sdl_error_position[65536];

/* Calculate new CRC from old^(byte<<8) */
u16
crc16_t8(u16 crcval)
{
    u16 f1,f2,f3;

    f1 = (crcval>>8) | (crcval<<8);
    f2 = (crcval>>12) | (crcval&0xF000) | ((crcval>>7)&0x01E0);
    f3 = ((crcval>>3) & 0x1FE0) ^ ((crcval<<4) & 0xF000);
    return f1^f2^f3;
}
```

```
void
generate_error_table(u8 *bptab, int nbytes)
{
    u16 crc;
    int i, j, k;

    /* Marker for no error */
    bptab[0] = 0xFE;

    /* Marker for >1 error */
    for (i = 1; i < 65536; i++ )
        bptab[i] = 0xFF;

    /* Mark all single bit error cases. */
    printf("Error syndrome table:\n");
    for (i = 0; i < nbytes; i++) {
        putchar(' ');

        for (j = 0; j < 8; j++) {
            crc = 0;
            for (k = 0; k < i; k++)
                  crc = crc16_t8(crc);
            crc = crc16_t8(crc ^ (0x8000>>j));
            for (k++; k < nbytes; k++)
                  crc = crc16_t8(crc);
            bptab[crc] = (i * 8) + j;
            printf(" %04X",crc);
        }
        putchar('\n');
    }
}

int
main(int argc, char **argv)
{
    u8 buffer[8] = {
        0x01,0x55,0x02,0xaa,
        0x99,0x72,0x18,0x56
    };
    u16 crc;
    int i;

    generate_error_table(sdl_error_position,8);

    /* Run sample message through check routine. */
    crc = 0;
    for (i = 0; i < 8; i++)
        crc = crc16_t8(crc ^ (buffer[i]<<8));
```

```
        /* Output is 0000 64 -- no error encountered. */
        printf("\nError test:  CRC %04X, bit position %d\n",
          crc,sdl_error_position[crc]);
    }
```

9. Security Considerations

The reliability of public SONET/SDH networks depends on well-behaved traffic that does not disrupt the synchronous data recovery mechanisms. This document describes framing and scrambling options that are used to ensure the distribution of transmitted data such that SONET/SDH design assumptions are not likely to be violated.

10. References

[1] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[2] Simpson, W., Editor, "PPP in HDLC-like Framing", STD 51, RFC 1662, July 1994.

[3] Malis, A. and W. Simpson, "PPP over SONET/SDH", RFC 2615, June 1999.

[4] "American National Standard for Telecommunications - Synchronous Optical Network (SONET) Payload Mappings," ANSI T1.105.02-1995.

[5] ITU-T Recommendation G.707, "Network Node Interface for the Synchronous Digital Hierarchy (SDH)," March 1996.

[6] Doshi, B., Dravida, S., Hernandez-Valencia, E., Matragi, W., Qureshi, M., Anderson, J., Manchester, J.,"A Simple Data Link Protocol for High Speed Packet Networks", Bell Labs Technical Journal, pp. 85-104, Vol.4 No.1, January-March 1999.

[7] Demers, A., S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," ACM SIGCOMM volume 19 number 4, pp. 1-12, September 1989.

[8] Floyd, S. and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, August 1993.

[9] Simpson, W., Editor, "PPP LCP Extensions", RFC 1570, January 1994.

[10] ITU-T Recommendation I.432.1, "B-ISDN User-Network Interface - Physical Layer Specification: General Characteristics," February 1999.

[11] ITU-T Recommendation V.41, "Code-independent error-control system," November 1989.

[12] ITU-T Recommendation G.783, "Characteristics of synchronous digital hierarchy (SDH) equipment functional blocks," April 1997.

11. Acknowledgments

PPP over SONET was first proposed by Craig Partridge (BBN) and is documented by Andrew Malis and William Simpson as RFC 2615.

Much of the material in this document was supplied by Lucent.

Other length-prefixed forms of framing for PPP have gone before SDL, such as William Simpson's "PPP in Ether-like Framing" expired draft.

12. Working Group and Chair Address

The working group can be contacted via the mailing list (ietf-ppp@merit.edu; send mail to ietf-ppp-request@merit.edu to subscribe), or via the current chair:

Karl Fox
Extant, Inc.
3496 Snouffer Road, Suite 100
Columbus, Ohio 43235

EMail: karl@extant.net

13. Intellectual Property Notices

obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

14. Authors' Addresses

James Carlson
Sun Microsystems, Inc.
1 Network Drive MS UBUR02-212
Burlington MA 01803-2757

Phone: +1 781 442 2084
Fax: +1 781 442 1677
EMail: james.d.carlson@sun.com

Paul Langner
Lucent Technologies Microelectronics Group
555 Union Boulevard
Allentown PA 18103-1286

EMail: plangner@lucent.com

Enrique J. Hernandez-Valencia
Lucent Technologies
101 Crawford Corners Rd.
Holmdel NJ 07733-3030

EMail: enrique@lucent.com

James Manchester
Lucent Technologies
101 Crawford Corners Rd.
Holmdel NJ 07733-3030

EMail: sterling@hotair.hobl.lucent.com

15. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.