                    PPP Stac LZS Compression Protocol

Status of this Memo

   This memo provides information for the Internet community. This memo
   does not specify an Internet standard of any kind. Distribution of
   this memo is unlimited.

Abstract

   The Point-to-Point Protocol (PPP) [1] provides a standard method for
   transporting multi-protocol datagrams over point-to-point links.

   The PPP Compression Control Protocol [2] provides a method to
   negotiate and utilize compression protocols over PPP encapsulated
   links.

   This document describes the use of the Stac LZS data compression
   algorithm, with single or multiple compression histories, for
   compressing PPP encapsulated packets.

Table of Contents

1. Introduction

   Starting with a sliding window compression history, similar to LZ1
   [3], Stac Electronics developed a new, enhanced compression algorithm
   identified as Stac LZS. The LZS algorithm is optimized to compress
   all file types as efficiently as possible. Even string matches as
   short as two octets are effectively compressed.

   The Stac LZS compression algorithm supports both single compression
   history communication and multiple compression history communication.

   A single compression history will require the minimum amount of
   memory to implement, but may not provide as much compression as a
   multiple history implementation.

   Often, many streams of information are interleaved over the same
   link. Each virtual link will transmit data that is independent of
   other virtual links. Using multiple compression histories can
   improve the compression ratio of a communication link by associating
   separate compression histories with separate virtual links of
   communication.

1.1. Licensing

   Source and object licenses are available on a non-discriminatory
   basis. Hardware implementations are also available. Contact Stac
   Electronics at the address and phone number listed with the author's
   address for further information.

1.2. Specification of Requirements

   In this document, several words are used to signify the requirements
   of the specification. These words are often capitalized.

   MUST   This word, or the adjective "required", means that the
               definition is an absolute requirement of the specification.

   MUST NOT   This phrase means that the definition is an absolute
               prohibition of the specification.

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

   MAY   This word, or the adjective "optional", means that this
               item is one of an allowed set of alternatives. An
               implementation which does not include this option MUST be
               prepared to interoperate with another implementation which
               does include the option.

2. LZS Packets

   Before any LZS packets may be communicated, PPP must reach the
   Network-Layer Protocol phase.

   When the Compression Control Protocol (CCP) has reached the Opened
   state, and LZS is negotiated as the primary compression algorithm,
   exactly one Stac LZS datagram is encapsulated in the PPP Information
   field, where the PPP Protocol field indicates type hex 00FD
   (compressed datagram) or type hex 00FB (Individual link compressed
   datagram). Type hex 00FD is used when compression is negotiated over
   a single physical link or when compression is negotiated over a
   single bundle consisting of multiple physical links. Type hex 00FB
   is used when compression is negotiated separately over individual
   physical links to the same destination. For more information, please
   refer to PPP Compression Control Protocol.

   When CCP has not successfully reached the Opened state, or LZS is not
   the primary compression algorithm, exactly one LZS datagram is
   encapsulated in the PPP Information field, where the PPP Protocol
   field indicates type hex 4021 (Stac LZS).

      Note that in the latter case, use of LZS is terminated by the PPP
      LCP Protocol-Reject. The default format is used: a single history
      with no History Number field and no Check Value field (as if the

negotiated history count were 1).

The maximum length of the Stac LZS datagram transmitted over a PPP
link is the same as the maximum length of the Information field of a
PPP encapsulated packet.

Prior to compression, the uncompressed data begins with the PPP
Protocol ID Field. Protocol-Field-Compression MAY be used on this
value, if it has been successfully negotiated for the link.

The PPP Protocol ID Field is followed by the original Information
field. The length of the uncompressed data field is limited only by
the allowed size of the compressed data field and the higher protocol
layers.

PPP Link Control Protocol packets MUST NOT be sent within Stac LZS
packets. PPP Network Control Protocol packets MUST NOT be sent
within Stac LZS packets.

2.1. Padding

The LZS Information field always ends with the last compressed data
byte (also known as the <end marker>), which is used to disambiguate
padding. This allows trailing bits as well as octets to be
considered padding.

2.2 Zero Deletion/Insertion

When the sender does not add Padding [1], any trailing zero octets
MAY be removed prior to transmission. A single trailing zero octet
MUST be appended upon receipt, after removal of any framing FCS.

2.3. Reliability and Sequencing

When no Compression History is kept, the algorithm does not depend on
a reliable link, and does not require that packets be delivered in
sequence. However, per packet compression results in a lower
compression ratio than it could be on a stream.

Some reasons for resetting the history on a per packet basis include:

  - The link has a high error rate.
  - The resources of the transmitter or receiver limit the ability
     to maintain a compression history between packets.

When more than 1 Compression History is negotiated, the packet
sequence MUST be preserved within specific History Numbers. There is
no sequence requirement between different History Numbers.

When one or more compression histories is negotiated on the link, the
implementation MUST implement either a lower layer reliable link
protocol, or keep the compressor and decompressor histories in
synchronization, or both.

To maintain history synchronization, the implementation MUST use the
Reset-Request and Reset-Ack messages of the Compression Control
Protocol and MUST use an Option 17 check mode value of sequence
numbers (and MAY implement other check mode values other than none).
In this case the Data field of the CCP Reset-Request and Reset-Ack
MUST contain the two octet History Number to be reset, most
significant octet first.

If neither of these conditions are met on the data link, then the
compression histories MUST be reset after transmitting each datagram.

The transmitter MAY clear a Compression History at any time. The
receiver is implicitly notified of this event, and the decompression
history will automatically be affected.

The transmitter MUST reset a history after a CCP Reset-Request for
the given History Number.

2.3.1 Reset-Request and Reset-Ack Packet Formats

   A summary of the CCP Reset-Request and Reset-Ack packet formats
   for Stac LZS compressed links are shown below. The fields are
   transmitted from left to right.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Data              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
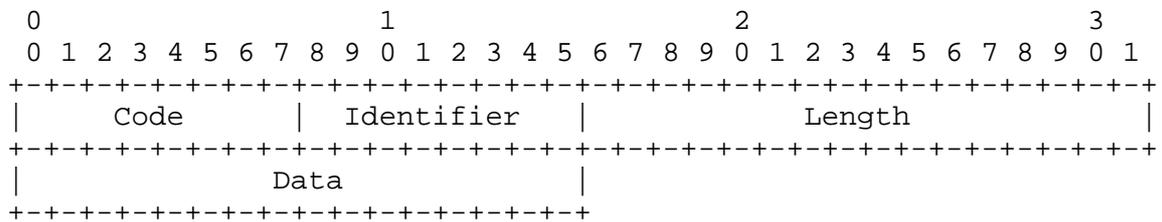
Code

   14 for Reset-Request;

   15 for Reset-Ack.

Identifier

   On transmission, the Identifier field MUST be changed whenever the
   content of the Data field changes, and whenever a valid reply has

been received for a previous request. For retransmissions, the
Identifier MAY remain unchanged.

On reception, the Identifier field of the Reset-Request is copied
into the Identifier field of the Reset-Ack packet.

Data

The Data field contains the two octet History Number of the
compression history that is to be reset, most significant octet
first. This History Number value is 1 when no history number is
present.

2.4. Data Expansion

The maximum expansion of Stac LZS is 12.5%.

A Maximum Receive Unit (MRU) MAY be negotiated that is 12.5% larger
than the size of a normal packet. Then, packets can always be sent
compressed regardless of expansion.

When the expansion plus compression header exceeds the size of the
peer's MRU for the link, the PPP packet MUST be sent without
compression, in the original PPP packet form with the "native" PPP
Protocol ID number. The transmitter MUST reset the affected history.

If it is detected that most packets are expanding (for example, due
to the use of already compressed data), then the transmitter SHOULD
stop sending compressed packets, and reset the appropriate history.
Data compression MAY be resumed on this data link later.

2.5. Packet Format

A summary of the Stac LZS packet format is shown below. The fields
are transmitted from left to right.
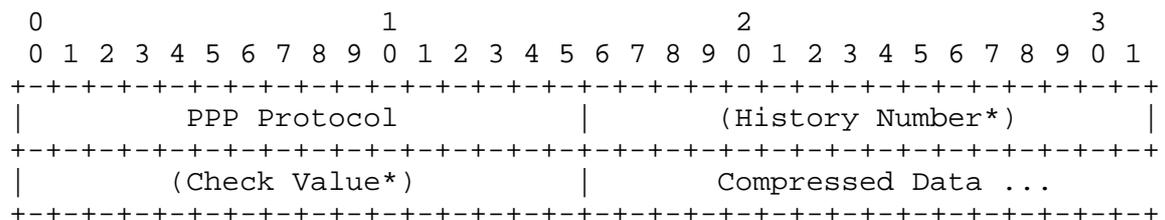
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          PPP Protocol         |         (History Number*)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          (Check Value*)       |       Compressed Data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
 * Note: these fields are variable length fields as described below.

2.5.1. PPP Protocol

   The PPP Protocol field is a 2 octet field described in the Point-
   to-Point Protocol Encapsulation [1].

   When the Stac LZS compression protocol is successfully negotiated
   by the PPP Compression Control Protocol [2], the value is 00FD hex
   or 00FB hex as described in section 2. This value MAY be
   compressed when Protocol-Field-Compression is negotiated.

2.5.2. History Number

   The history number field comprises 0, 1, or 2 octets.

   The number of the compression history which was used, ranging from
   2 to the negotiated History Count. By default a History Count of
   value 1 is supported and this field is not present.

   If the negotiated History Count is less than 2, this field is
   removed. There is no need for the field when no history is kept,
   or only a single history is kept.

   If the negotiated History Count is 2 or more, but less than
   256,this field is 1 octet. If 256 or more histories are
   negotiated, this field is 2 octets, most significant octet first.

2.5.3. Check Value

   The check value field comprises 0, 1, or 2 octets. By default,
   sequence number check is added to the packet (the field comprises
   1 octet).

   2.5.3.1. LCB

      A simple one octet Longitudinal Check Byte (LCB) MAY be used,
      after successful negotiation of the LCB option. The LCB is the
      Exclusive-OR of FF(hex) and each octet of the uncompressed
      datagram (prior to the compression operation). On receipt, the
      receiver computes the Exclusive-OR of FF(hex) and each octet of
      the decompressed packet. If this value does not match the
      received LCB, then a receive failure for that history has
      occurred. The receive failure is handled according to the
      history synchronization procedure in section 2.5.4.

   2.5.3.2. CRC

      A two octet Cyclic Redundancy Check (CRC) MAY be used, instead
      of the LCB, after successful negotiation of the CRC option.

The transmitter MUST initialize the CRC value to FFFF(hex) at
the beginning of each packet. The CRC computation is based on
the HDLC FCS-16 polynomial:

    $x**16 + x**12 + x**5 + 1$

The ones complement of the CRC is transmitted least significant
octet first, which contains the coefficient of the highest
term. On receipt, the receiver initializes the CRC to FFFF
(hex), and computes the CRC based on the formula above for each
octet of the decompressed packet. If the received CRC value
does not match the transmitted CRC value, then a receive
failure for that history has occurred. The receive failure is
handled according to the history synchronization procedure in
section 2.5.4.

2.5.3.3. Sequence Number

A one octet Sequence Number MAY be used, instead of a LCB or
CRC, after successful negotiation of the Sequence Number
option. After CCP has reached the open state, the transmitter
MUST set the value of the sequence number field (the sequence
number of the packet) to "1" and increment modulo 256 on
successive packets that contain data fields. The sequence
number is relative to the history number used.

After CCP has reached the open state, the receiver MUST set its
internal reference value of the next expected sequence number
(the sequence number of next packet to be received) to "1".

After a packet is received, the receiver MUST set the value of
its internal reference value of the next expected sequence
number for that history to the value of the sequence number
field of the received packet plus 1 modulo 256.

If the sequence number of the received packet is not equal to
the internal reference value of the expected sequence number
for the same history, a receive failure for that history has
occurred. The receiver MUST silently discard the out of order
packet, and handle the failure according to the history
synchronization procedure in section 2.5.4.

The sequence number MUST NOT be reset by the transmitter when a
packet containing a Reset-Req is received. The receiver MUST
always maintain its sequence number references for all
supported histories.

```
     2.5.3.3.1  History Synchronization with Sequence Numbers Example

     Compressing Sender                     Decompressing Receiver
     ....                                    ....
     send seq 101      ----------->          recv seq 101
                                             is 101 == 101?  Ok.
                                             forward packet for processing
                                             set internal reference to 102

     send seq 102      ----------->          recv seq 102
                                             is 102 == 102?  Ok.
                                             forward packet for processing
                                             set internal reference to 103

     send seq 103      ------X               (packet lost)

     send seq 104      ----------->          recv seq 104
                                             is 104 == 103?  Send reset req!
                                             silently discard packet
                                             set internal reference to 105

     (packet lost)         X-------          send reset request (ID=200)
                                             post-increment the identifier.

     send seq 105      ----------->          recv seq 105
                                             is 105 == 105?  Ok.
                                             was reset ack received?  No!
                                             silently discard packet
                                             set internal reference to 106

                       <-----------          send reset request again(ID=200)
                                             (e.g. reset-ack time out)

     send seq 106      ------X               (packet lost)

     recv reset req    <-----------
     (after line delay)
        (ID=200)

     reset compression
        history
     send reset ack    ----------->          recv reset ack (ID=200)
        (ID=200)

     send seq 107      ----------->          recv seq 107
                                             is 107 == 106?  Send reset req!
                                             silently discard packet
                                             set internal reference to 108
```

```
      recv reset req    <-----------    send reset request (ID=201)
         (ID=201)                       post-increment the identifier.

      send seq 108      ----------->    recv seq 108
                                        is 108 == 108?  Ok.
                                        was reset ack received?  No!
                                        silently discard packet
                                        set internal reference to 109

      send seq 109      ----------->    recv seq 109
                                        is 109 == 109?  Ok.
                                        was reset ack received?  No!
                                        silently discard packet
                                        set internal reference to 110

      reset compression
         history
      send reset ack    ----------->    recv reset ack (ID=201)
         (ID=201)

      send seq 110      ----------->    recv seq 110
                                        is 110 == 110?  Ok.
                                        forward packet for processing
                                        set internal reference to 111

      send seq 111      ----------->    recv seq 111
                                        is 111 == 111?  Ok.
                                        forward packet for processing
                                        set internal reference to 112
      ....                              ....
```

2.5.4. History Synchronization Procedure

   On receipt, if Sequence Number one (1) follows any other number
   than zero (0), or is otherwise out of sequence, or the LCB or CRC
   is invalid, a CCP Reset-Request MUST be sent, containing the two
   octet History Number (most significant octet first, and which is
   the value 1 when no History Number is present), with a CCP
   Identifier. Identifiers are incremented on each occurrence of an
   out of sequence packet.

   Upon receipt of the Reset-Request, the transmitter MUST reset the
   affected compression history, and transmit a CCP Reset-Ack packet
   with the Identifier field and data (history number) field set to
   the corresponding values of the Reset-Request. However, the
   Sequence Number (if implemented) is not reset.

For each packet that generates a receive failure, the receiver
MUST increment the Identifier and transmit a CCP Reset-Request.
For re-transmissions of existing receive failures, the Identifier
MUST NOT be incremented.

After transmitting the Reset-Request packet, the receiver MUST
continue silently discarding valid compressed packets for the
corresponding history, until the correct CCP Reset-Ack Identifier
(corresponding to the Reset-Request) for that History Number is
received. Note that if sequence numbers are used, the receiver
MUST process the sequence number of a received packet according to
the procedures in section 2.5.4.

2.5.5. Compressed Data

The data field MUST contain only one datagram in compressed form.
The length of this field is always an integer number of octets.
There MUST BE only one end marker per block of compressed data.

The form of the data field is one block of compressed data as
defined in 3.2 of X3.241-1994, and is repeated here for
informational purposes ONLY.

```
<Compressed Stream> := [<Compressed String>] <End Marker>
<Compressed String> := 0 <Raw Byte> | 1 <Compressed Bytes>
<Raw Byte> := <b><b><b><b><b><b><b><b>           (8-bit byte)
<Compressed Bytes> := <Offset> <Length>

<Offset> := 1 <b><b><b><b><b><b><b> |             (7-bit offset)
            0 <b><b><b><b><b><b><b><b><b><b><b> (11-bit offset)
<End Marker> := 110000000
```

```
<b> := 1 | 0
```

```
<Length> :=
00        = 2    1111 0110      = 14
01        = 3    1111 0111      = 15
10        = 4    1111 1000      = 16
1100      = 5    1111 1001      = 17
1101      = 6    1111 1010      = 18
1110      = 7    1111 1011      = 19
1111 0000 = 8    1111 1100      = 20
1111 0001 = 9    1111 1101      = 21
1111 0010 = 10   1111 1110      = 22
1111 0011 = 11   1111 1111 0000 = 23
1111 0100 = 12   1111 1111 0001 = 24
1111 0101 = 13       ...
```

3. Sending Compressed Datagrams

   The reliable and efficient transport of datagrams on the data link
   depends on the following processes.

3.1. Transmitter Process

   When a network datagram is received, it is assigned to a particular
   history buffer and processed according to ANSI X3.241-1994 to form
   compressed data. Prior to the compression operation, if a Reset-
   Request is outstanding for the history buffer to be used or if the
   negotiated history count for this data link is 0, the history buffer
   is cleared.

   Uncompressed data MUST be sent (in the original PPP packet form with
   the "native" PPP Protocol ID number) if compression causes enough
   expansion to cause the data compression datagram size to exceed the
   Information field's MRU. In this case, since the compressor has
   modified the history buffer before sending an uncompressed datagram,
   the history buffer MUST be cleared before the next datagram is
   processed.

   The output of the compression operation is placed in the information
   field of the datagram. If the sequence number field is present
   according the value of the check mode field, the sequence number
   counter for the applicable history number MUST be incremented and its
   value placed in the sequence number field. If the LCB field is
   present according the value of the check mode field, the LCB value
   MUST be computed as specified in section 2.5.3.1. and the resultant
   value placed in the LCB field. If the CRC field is present according
   the value of the check mode field, the CRC value MUST be computed as
   specified in section 2.5.3.2. and the resultant value placed in the
   LCB field. Upon reception of a CCP Reset-Request packet, the
   transmitting compressor MUST be cleared to an initial state, which
   includes clearing the history buffer. In addition to the reset of
   the compressor, a CCP Reset-Ack packet MUST be transmitted. The data
   field of this packet MUST be filled with the corresponding two octet
   history number, most significant octet first.

3.2. Receiver Process

   If a CCP Reset-Request packet is received, the local compression
   engine MUST be signaled that a Reset-Request has been received for
   the history number specified in the data field. If a CCP Reset-Ack
   packet is received, any outstanding receive failure for the specified
   history MUST be cleared. If no receive failure is outstanding, and
   the sequence number field is present, its value is checked. If a
   receive failure has occurred, it MUST be handled according to the

history resynchronization mechanism described below, and the
remainder of the datagram is discarded.

If no receive failure is detected, the data is assigned to the
indicated decompression history buffer and the compressed data block
MUST be decompressed according to ANSI X3.241-1994. If the LCB or
CRC fields are present on the received datagram, an LCB or CRC for
the uncompressed data MUST be computed and checked against the
received LCB or CRC according to sections 2.5.3.1. or 2.5.3.2.,
respectively. If a receive failure has occurred, it MUST be handled
according to the History Resynchronization Mechanism described in
section 3.4.

If a CCP Reset-Ack packet is received, the receiving decompressor's
corresponding history MAY be reset to an initial state. (However,
due to the characteristics of the Stac LZS algorithm, a decompressor
history reset is not required). After reset, any compressed or
uncompressed data contained in the packet is processed.

On the occurrence of a receive failure, an implementation MUST
transmit a CCP Reset-Request packet with the data field containing
the two octet history number (most significant octet first) matching
the history that had the failure. Once a receive failure has
occurred, the data in any subsequent packets received for that
history MUST be discarded until a CCP Reset-Ack packet containing a
valid Identifier matching the Identifier that was sent with the last
CCP Reset-Request packet is received. It is the responsibility of
the receiver to ensure the reliability of the Reset-Request/Ack
mechanism. This may require the transmission of additional CCP
Reset-Request packets before a CCP Reset-Ack packet is received.

3.3. History Maintenance

The History Count field determines the number of history buffers to
be maintained for the compression protocol. For example, each
history buffer could represent a separate logical connection between
the data compression peers. When maintaining a history, the peers
MUST use link error detection and signaling to ensure that both the
compressor and decompressor copies of each history buffer are always
identical.

Setting the History Count field to the value "0" indicates that the
compression is to be on a connectionless basis. In this case, a
single history buffer is used and MUST be cleared at the beginning of
every datagram.

When the History Count field is set to the value "1", a single
history buffer is maintained by each of the data compression peers.

(A single logical connection.)

When the History Count field is set to a value greater than "1",
separate history buffers, error detection states, and signaling
states are maintained by the decompressing entity for each history.
The compressing peer may transmit data on any number of separate
histories, up to the value of the History Count field.

3.4. History Resynchronization Mechanism

The Stac LZS protocol utilizes CCP Reset-Request/Reset-Ack mechanism
in order to provide a mechanism for indicating a receiver failure in
one direction of a compressed link without affecting traffic in the
other direction. A receive failure is determined using the LCB, CRC,
or sequence number mechanisms, according to the value of the check
mode field.

Reset-Requests and Reset-Acks are specific to the history number of
the packet containing them.

Reset-Request/Reset-Ack history synchronization signaling is provided
to recover from a loss of synchronization between peers, especially
in unreliable transport layers. As with all compression algorithms,
the decompressor can not recover from dropped, erroneous, or mis-
ordered datagrams, and will propagate errors catastrophically until
both peers are reset to an initial state.

The Stac LZS protocol provides a means to detect these error
conditions: LCB or CRC for erroneous datagrams, and sequence number
for dropped or mis-ordered datagrams. There is a means for
correcting a loss of synchronization: clear both the failing
compression and decompression histories, and follow the transmitter
and receiver processes in sections 3.1. and 3.2.

4. Configuration Option Format

Description

     The CCP Stac LZS Configuration Option negotiates the use of
     Stac LZS on the link. By ultimate disagreement, no compression is
     used.

     All implementations must support the default values.

A summary of the Stac LZS Configuration Option format is shown
below. The fields are transmitted from left to right.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |     Length    |          History Count        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Check Mode  |
+-+-+-+-+-+-+-+-+
```
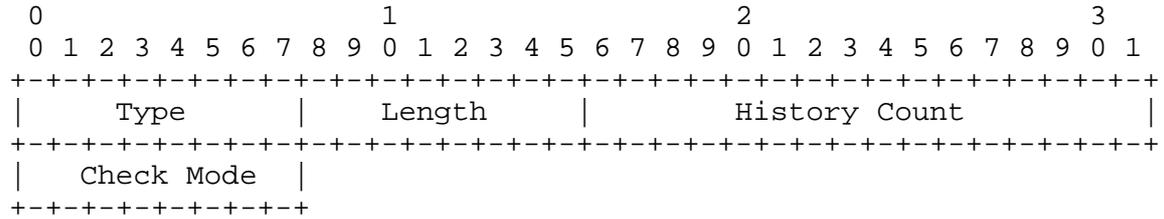
Type

     17

Length

     5

History Count

     The History Count field is two octets, most significant octet
     first, and specifies the maximum number of Compression Histories.

     The value 0 indicates that the implementation expects the peer to
     clear the Compression History at the beginning of every packet.

     The value 1 is the default value, and is used to indicate that
     only one history is maintained.

     Other valid values range from 2 to 65535. The peer is not
     required to send as many histories as the implementation indicates
     that it can accept. However, it should be noted that resources
     are allocated in each peer to support the number of negotiated
     histories in this field.

   Check Mode

     The Check Mode field indicates support of LCB, CRC or Sequence
     checking, and other future extensions to this standard. This
     field comprises 2 sub-fields, and is considered to be bit-mapped.
     The 3 least significant bits comprise 5 mutually exclusive values.
     The upper 5 bits are all "Reserved" bit locations must be set to
     "0" to allow for future backward-compatible extensions to this
     standard.

     For compatibility, Sequence Numbers MUST be implemented; the other
     four check modes MAY be implemented.

Defined values:

     0   None   (MAY be implemented; however, MUST
                               implement history count of zero)
     1   LCB   (MAY be implemented)
     2   CRC   (MAY be implemented)
     3   Sequence Number   (MUST be implemented)
     4   Extended Mode   (MAY be implemented)

        0   1   2   3   4   5   6   7
     +-------+-------+----------+-----+-----+-----+-----+-----+
     | LCB/CRC/Seq#/Ext'd | Res | Res | Res | Res | Res |
     +-------+-------+----------+-----+-----+-----+-----+-----+
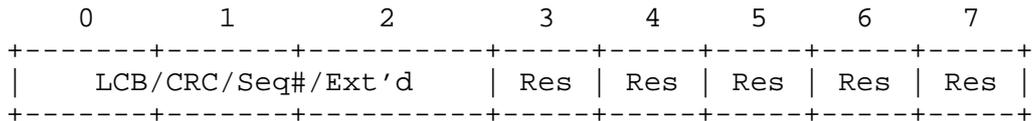
5. Definition of Extended Mode

   When Check Mode 4 (Extended Mode) is successfully negotiated, the
   packet format is different from the format described above. The
   Extended Mode format is described below. Extended Mode only supports
   a history count of 1.

5.1. Extended Mode Packet Format

    0   1   2   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+
   | PPP Protocol |A|B|C|D| Coherency Count |
   +-+
   | Compressed Data...
   +-+
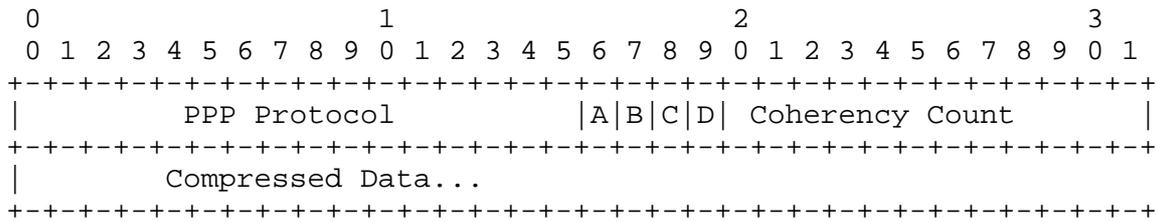
   PPP Protocol

      The PPP Protocol field is described in the Point-to-Point Protocol
      Encapsulation [1].

      When a compression protocol is successfully negotiated by
      the PPP Compression Control Protocol [2], the value is hex 00FD.
      Protocol-Field-Compression MUST NOT be used on this value when
      extended mode is negotiated on the link, even if Protocol-Field-
      Compression was successfully negotiated before data compression.

   Bit A - PACKET_FLUSHED

      This bit indicates that the history buffer has just been reset
      before this packet was generated. Thus, this packet can ALWAYS
      be decompressed because it is not based on any previous history.
      This bit is typically sent to inform the peer that it has reset
      its history buffer and that the peer can accept this packet
      and re-synchronize.

   Bit B

      This bit is not used with Stac LZS compression.

   Bit C - PACKET_COMPRESSED

      This bit is used to indicate that the packet is compressed. A
      value of 0 indicates uncompressed data, and a value of 1 indicates
      compressed data.

   Bit D

      This bit is not used with Stac LZS compression.

   Coherency Count

      The coherency count is used to assure that the packets are sent in
      proper order and that no packet has been dropped. This count is
      initialized to the value 0x000, and is always increased by 1 after
      each PPP packet is sent. When all bits are 1, the count returns
      to 0.

      The coherency count is 12 bits so the decompressor must handle the
      rollover case.

   Compressed Data

      The compressed data begins with the protocol field. For example,
      an IP packet may contain 0021 followed by an IP header. The
      compressor will first try to compress the 0021 protocol field and
      then move on to the IP header.

      Protocol-Field-Compression MUST NOT be used on this value when
      extended mode is negotiated on the link, even if Protocol-Field-
      Compression was successfully negotiated before data compression.

      Zero deletion/insertion described in section 2.2 MUST NOT be
      performed when extended mode is negotiated.

5.2. Extended Mode Transmitter Process

   When a network datagram is received, it is processed according to
   ANSI X3.241-1994 to form compressed data. If a CCP Reset-Request has
   been received from the decompressor, the compressor must clear its
   history buffer before sending the next packet.

   Uncompressed data MUST be sent if the compression operation causes
   the compressed datagram to expand. In this case, since the
   compressor has modified the history buffer before sending an
   uncompressed datagram, the history buffer MUST be cleared before the
   next datagram is processed. The uncompressed data is placed in the
   information field of the datagram, and Bit-A MUST be set (indicating
   the history was cleared) and Bit-C MUST be clear (indicating
   uncompressed data) in the current packet's header. The value of the
   coherency counter is placed in the coherency count field and then the
   coherency counter is incremented.

   If the compression operation does not cause the compressed datagram
   to expand and if a received Reset-Request is outstanding, then the
   output of the compression operation is placed in the information
   field of the datagram, and Bit-A MUST be set (indicating the history
   was cleared) and Bit-C MUST be set (indicating compressed data) in
   the current packet's header. The value of the coherency counter is
   placed in the coherency count field and then the coherency counter is
   incremented.

   If the compression operation does not cause the compressed datagram
   to expand and there is not a Reset-Request outstanding, then the
   output of the compression operation is placed in the information
   field of the datagram, and Bit-A MUST be clear (indicating the
   history was not cleared) and Bit-C MUST be set (indicating compressed
   data) in the current packet's header. The value of the coherency
   counter is placed in the coherency count field and then the coherency
   counter is incremented.

   Upon reception of a CCP Reset-Request packet, the transmitting
   compressor MUST be cleared to an initial state, which includes
   clearing the history buffer. In addition to the reset of the
   compressor, the PACKET_FLUSHED bit MUST be set in the header of the
   next transmitted data packet.

5.3. Extended Mode Receiver Process

   When a data compression datagram is received from the peer, Bit-A and
   Bit-C MUST be checked. Prior to the decompression operation, if
   Bit-A is set, then the coherency count MUST be resynchronized to the
   received value in the coherency count field of the received packet,

and the receiving decompressor's corresponding history MAY be reset
to an initial state. (However, due to the characteristics of the
Stac LZS algorithm, a decompressor history reset is not required).
After reset, any compressed or uncompressed data contained in the
packet is processed, depending on the state of Bit-C.

Prior to the decompression operation, if Bit-C is clear (indicating
uncompressed data), then the decompression history buffer must not be
modified and the decompressor is not involved with deencapsulation.
If Bit-C is set (indicating compressed data) then the received packet
is decompressed according to ANSI X3.241-1994.

If the received packet is corrupt, then a Reset-Request is sent and
this packet is discarded. If the received packet contains an
incorrect coherency count, a Reset-Request is sent and this packet is
discarded.

5.4. Extended Mode Synchronization

Packets may be lost during transfer. If the decompressor maintained
coherency count does not match the coherency count received in the
compressed packet or if the decompressor detects that a received
packet is corrupted, the decompressor drops the packet and sends a
CCP Reset-Request packet. The compressor on receiving this packet
resets the history buffer and sets the PACKET_FLUSHED bit in the next
frame it sends. The decompressor on receiving a packet with its
PACKET_FLUSHED bit set, resets its history buffer and sets its
coherency count to the one shipped by the compressor in that packet.

Thus synchronization is achieved without a Reset-Ack packet.

Security Considerations

Security issues are not discussed in this memo.

References

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

    [2]   Rand, D., "The PPP Compression Control Protocol (CCP)", RFC
           1962, July 1996.

    [3]   Lempel, A. and Ziv, J., "A Universal Algorithm for Sequential
           Data Compression", IEEE Transactions On Information Theory,
           Vol. IT-23, No. 3, May 1977.

    [4]   Rand, D., "PPP Reliable Transmission", RFC 1663, Novell, July
           1994.

Chair's Address

   The working group can be contacted via the current chair:

      Karl F. Fox
      Ascend Communications
      3518 Riverside Dr., Suite 101
      Columbus, Ohio   43221

      (614) 451-1883

      EMail: karl@ascend.Com

Authors' Addresses

   Questions about this memo can also be directed to:

      Robert Friend
      Stac Technology
      12636 High Bluff Drive
      San Diego, CA   92130
      (619) 794-4542
      EMail: rfriend@stac.com

      William Allen Simpson
      Daydreamer
      Computer Systems Consulting Services
      1384 Fontaine
      Madison Heights, Michigan   48071
      Bill.Simpson@um.cc.umich.edu
          bsimpson@MorningStar.com (preferred)