            Compressing IPX Headers Over WAN Media (CIPX)

Status of this Memo

Abstract

   This document describes a method for compressing the headers of IPX
   datagrams (CIPX). With this method, it is possible to
   significantly improve performance over lower speed wide area
   network (WAN) media. For normal IPX packet traffic, CIPX can
   provide a compression ratio of approximately 2:1 including both IPX
   header and data. This method can be used on various type of WAN
   media, including those supporting PPP and X.25.

   This memo ia a product of the Point-to-Point Protocol Extensions
   (PPPEXT) Working Group of the IETF. Comments should be sent to
   the authors and the ietf-ppp@ucdavis.edu mailing list.

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 should 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.

Introduction

   Internetwork Packet Exchange (IPX) is a protocol defined by the
   Novell Corporation [1]. It is derived from the Internet Datagram
   Protocol (IDP) protocol of the Xerox Network Systems (XNS) family
   of protocols. IPX is a datagram, connectionless protocol that does
   not require an acknowledgment for each packet sent. The IPX
   protocol corresponds to the network layer of the ISO model.

   Usually, there is a transport layer protocol above IPX. The most
   common transport protocol is the Netware Core Protocol (NCP), which
   is used for file server access. The Sequenced Packet Exchange
   (SPX) is the reliable connection-based transport protocol commonly
   used by applications.

   The IPX packet consists of a 30 octet IPX header, usually followed
   by the transport layer protocol header. The NCP header is 6 octets
   in length. The SPX header is 12 octets in length.

   Two strategies are described below for compressing IPX headers.
   This specification requires that implementations of CIPX support
   both IPX header compression strategies. These header compression
   algorithms are based on those Van Jacobson described [2] for TCP/IP
   packets.
   The first strategy is to compress only the IPX header. This
   compression algorithm can be used to compress any IPX packet,
   without affecting the transport protocol. This algorithm
   compresses a 30 octet IPX header into a one to seven octet header.

   The second strategy is to compress the combined IPX and NCP
   headers. This algorithm compresses only NCP packets with NCP type
   of 0x2222 and 0x3333. This algorithm compresses a 36 octet NCP/IPX
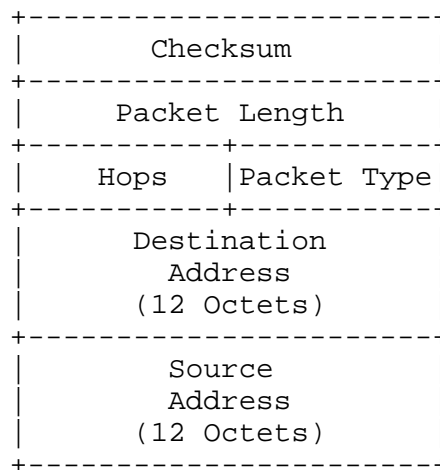
header into a one to eight octet header.

Lastly, it is possible and many times desirable, to use this type
of header compression in conjunction with some type of data
compression.

Data compression technology takes many forms. Link bit stream
compression is a common approach over very low speed asynchronous
links, normally performed by modems transparently. Transparent bit
stream compression is also offered in some DSUs, routers and
bridges. Data compression can be provided using protocols such as
CCITT V.42bis[3], MNP 5, Lempel-Ziv, or LAPB[4].

When using both header and data compression, the sequence of
compression is important. When sending packets, data compression
MUST be done after header compression. Conversely when receiving
packets, data decompression MUST be done before header
decompression.

IPX Compression Algorithm

    The normal IPX header consists of the following fields: checksum,
    packet length, transport control (hop count), packet type,
    destination and source address fields.

```
                    +-----------------------+
                    |       Checksum        |
                    +-----------------------+
                    |     Packet Length     |
                    +----------+------------+
                    |   Hops   |Packet Type|
                    +----------+------------+
                    |      Destination      |
                    |        Address        |
                    |      (12 Octets)      |
                    +-----------------------+
                    |        Source         |
                    |        Address        |
                    |      (12 Octets)      |
                    +-----------------------+
```

                      IPX PACKET HEADER

    The IPX header diagram above is shown without the field alignment
    details. Consider each field of the IPX header separately, and how
    it typically changes.

    Historically, Novell has not used the Checksum field in the IPX

header, and has required that this field be set to 0xFFFF. Since the
Checksum field remains constant, it is clear that the value can be
compressed.

Where Checksums are implemented (not 0xFFFF), the Checksum MUST be
included in the compressed packet. Recalculating the checksum would
destroy the end-to-end reliability of the connection. Note that
Checksums are now implemented in the Fault Tolerant Servers.

For most links, the Packet Length can be determined from the MAC
layer. There are cases in which the length cannot be determined from
the MAC layer. For example, some hardware devices pad packets to a
required minimum length. For links where it is not possible to
determine the IPX packet length from the MAC layer, packet length
needs to be included in the compressed packet.

The Transport Control (Hops) field usually does not change between
two end-points. For the purposes of compression, we will assume that
it never changes, and will not examine this field when determining a
connection.

The Packet Type field is constant for any connection.

The Destination and Source Address fields are each made up of 12
octets: Network (4 octets), Node (6 octets), and Socket (2 octets)
fields. An IPX connection is the logical association between two
endpoints known by a given source and destination address pair. For
any specific IPX connection, the Destination and Source Address
fields are constant.

Hence, the fields that may need to be included in the compressed IPX
header are the Checksum and the Packet Length.

While using this IPX header compression algorithm, packets can be
lost. The loss of an Initial packet presents a problem. In this
case, if the sender later tries to send a compressed packet, the
receiving end cannot decompress the packet correctly.

Sufficient information is not available in the IPX header to
determine when a re-transmission has occured. For this reason, it is
necessary that the sender of an Initial packet be guaranteed that the
packet has been received. Therefore, we provide a mechanism for
Confirmation of an Initial packet.

NCP/IPX Header Compression

Since most IPX packets are Netware Core Protocol packets (packet type
17), compressing the NCP header will give us added performance. A

minimal CIPX implementation MUST also implement NCP/IPX compression.

```
+------------+
|    NCP     |
|    Type    |
+------------+
|  Sequence  |
|   Number   |
+------------+
| Connection |
|(low octet) |
+------------+
|    Task    |
|   Number   |
+------------+
| Connection |
|(high octet)|
+------------+
```
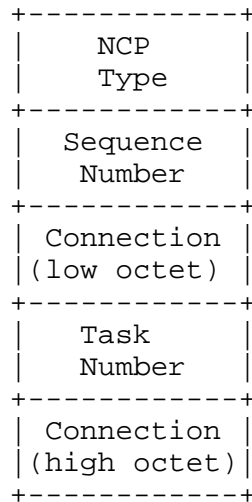
NCP HEADER

The NCP header is 6 octets in length consisting of the following
fields: NCP type, sequence number, connection number and task number.

The NCP type field values that are currently defined are:

```
        1111    Create Connection
        2222    NCP request from workstation
        3333    NCP reply from file server
        5555    Destroy Connection
        7777    Burst Mode Packet
        9999    Server Busy Packet
```

This NCP header compression algorithm only compresses packets that
have a type field value of 0x2222 or 0x3333. If the NCP type is
0x2222, this packet is a request from the client to the server.
Conversely if the NCP type is 0x3333, this is a response from the
server to the client. All other types of NCP packets are not
compressed at the NCP level, but are compressed at the IPX level.
The Create Connection (0x111), Destroy Connection (0x5555) and Server
Busy (0x9999) packets are not exchanged frequently enough to justify
special NCP compression. The Burst Mode (0x7777) packet is discussed
below.

The connection number is a constant for a given connection.

The sequence number is increased by one for each new request. Hence
the sequence number can be determined implicitly. The decompressor

increments the sequence number for each compressed packet it receives
for a connection.

The task number can change unpredictably, although it might remain
constant for several packets. If the NCP task number is different
from the last one for this connection, the NCP task number must be
included.

If the NCP packet is lost, it will be retransmitted through the
normal transport layer mechanisms. The Initial NCP packet does not
require confirmation, as a re-transmitted packet can be easily
identified. This is accomplished by comparing the sequence number of
the packet to the sequence number of the previous packet. If the
sequence number is not exactly one greater than the previous packet,
a new Initial packet must be sent, although the same connection slot
may be used.

In the event of compressed packet loss, the sequence number will be
too small. When the IPX Checksum is present, the loss can be
determined at the destination system by an incorrect checksum. When
there is no checksum present, the loss is more likely to be detected
upon receiving a later retransmission.

NCP Burst Mode Packets

   The burst mode protocol uses the NCP type value of 0x7777. This type
   of packet does not have the normal NCP header described above.
   Instead, it has a 36 octet burst header. The above NCP header
   compression algorithm should not be used to compress this packet.
   The IPX header in this packet is still compressible with the IPX
   header compression algorithm described.

SPX Packets

      SPX packets are typically used by applications which require
      reliable service such as print servers. It is possible to apply a
      similar NCP/IPX technique to SPX/IPX packets. At this time, we
      have not described such a mechanism. The IPX header in this
      packet is still compressible with the IPX header compression
      algorithm described.

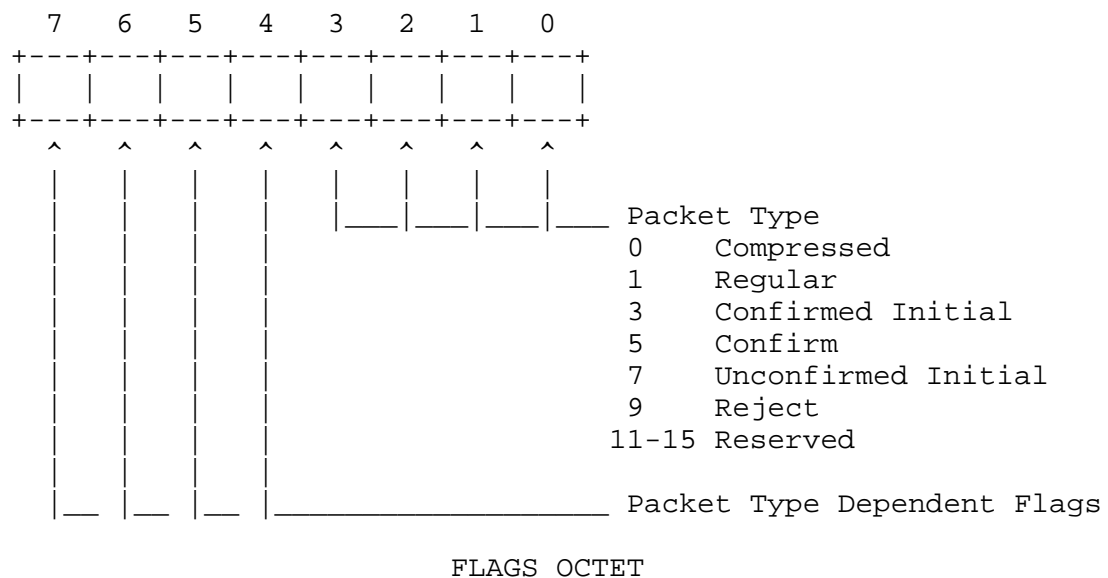Compression Header

      IPX compression should be negotiated by some means (eg. IPXCP or
      IPXWAN). Each end must negotiate the desired options, such as the
      maximum number of concurrent connections which will be maintained
      in each direction. Once IPX compression is negotiated, all IPX
      packets sent over that link have a CIPX header added to the

beginning of the packet. The CIPX header is variable in length.

The one octet CIPX header is added even when a regular IPX packet
is sent over the link. By including the CIPX header on every
packet, we support the ability to run CIPX over various WAN links
as if it were a normal IPX packet. It does not rely on any new
link specific packet demultiplexing.

Implementations of this compression protocol must maintain send
and receive tables indicating the state of each connection. The
original header for each connection is stored in a "slot".
Typically, each client-server connection will use a separate slot.
Both sides keep a copy of the full IPX header corresponding to
each slot. The sending side (compressor) uses this information to
determine the fields that have changed. The receiving side
(decompressor) uses this information to reconstruct the original
packet header.

The CIPX packet header specifies the type of the packet and any
options for that packet. The minimum CIPX header is one octet in
length.

```
   7   6   5   4   3   2   1   0
 +---+---+---+---+---+---+---+---+
 |   |   |   |   |   |   |   |   |
 +---+---+---+---+---+---+---+---+
   ^   ^   ^   ^   ^   ^   ^   ^
   |   |   |   |   |   |   |   |
   |   |   |   |   |___|___|___|___ Packet Type
   |   |   |   |              0    Compressed
   |   |   |   |              1    Regular
   |   |   |   |              3    Confirmed Initial
   |   |   |   |              5    Confirm
   |   |   |   |              7    Unconfirmed Initial
   |   |   |   |              9    Reject
   |   |   |   |           11-15 Reserved
   |   |   |   |
   |__ |__ |__ |_____ Packet Type Dependent Flags
```

                        FLAGS OCTET

As can be seen above, the low order bits specify the packet type.
All Compressed packets have a lowest bit of zero. The other
packet types are odd numbers.

Note that the Flags octet MUST NOT contain the value 0xFF. This
is necessary to distinguish the CIPX flags octet from a normal IPX
header with a 0xFFFF checksum field. It is important to be able

to recognize a normal IPX header regardless of the state of
compression. It is possible with some link layer protocols such
as X.25 Permanent Virtual Circuits that one end of the link may
fail and start sending regular IPX packets without the CIPX
header. CIPX implementations MUST recognize this situation and
renegotiate the use of CIPX.

Each packet type has associated flag bits, which are called Packet
Type Dependent Flags. Different packet types have different
Packet Type Dependent Flags. All bits that are reserved or are
not specified must be set to zero.

Since none of the packet types other than Compressed currently
uses any of the flag bits, the packet type field could easily be
expanded. Any future expansion must ensure that at least one of
the bits in the Flags octet remains zero so the value cannot be
0xFF.

Compressed Packet

This type of packet does not contain a packet header (either 30 byte
IPX, or 36 byte NCP). A slot number indicates to the receiver which
saved header to use to formulate the original packet header before
passing the packet up to IPX.

```
 _____    Slot Number
|                                        0    Assume same as last packet
|                                        1    Included in packet
|
|    _____   Checksum
|   |                                    0    Assume 0xFFFF
|   |                                    1    Included in packet
|   |
|   |    _____   Length
|   |   |                                0    Determine from MAC length
|   |   |                                1    Included in packet
|   |   |
|   |   |    _____   Task Number (NCP only)
|   |   |   |                            0    Assume same as last packet
|   |   |   |                            1    Included in packet
|   |   |   |
|   |   |   |    _____   Reserved (Must be zero)
|   |   |   |   |   |   |
|   |   |   |   |   |   |    ____   Packet Type
|   |   |   |   |   |   |   |        0    Compressed Packet
v   v   v   v   v   v   v   v
+---+---+---+---+---+---+---+---+
|   |   |   |   | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+
  7   6   5   4   3   2   1   0
```
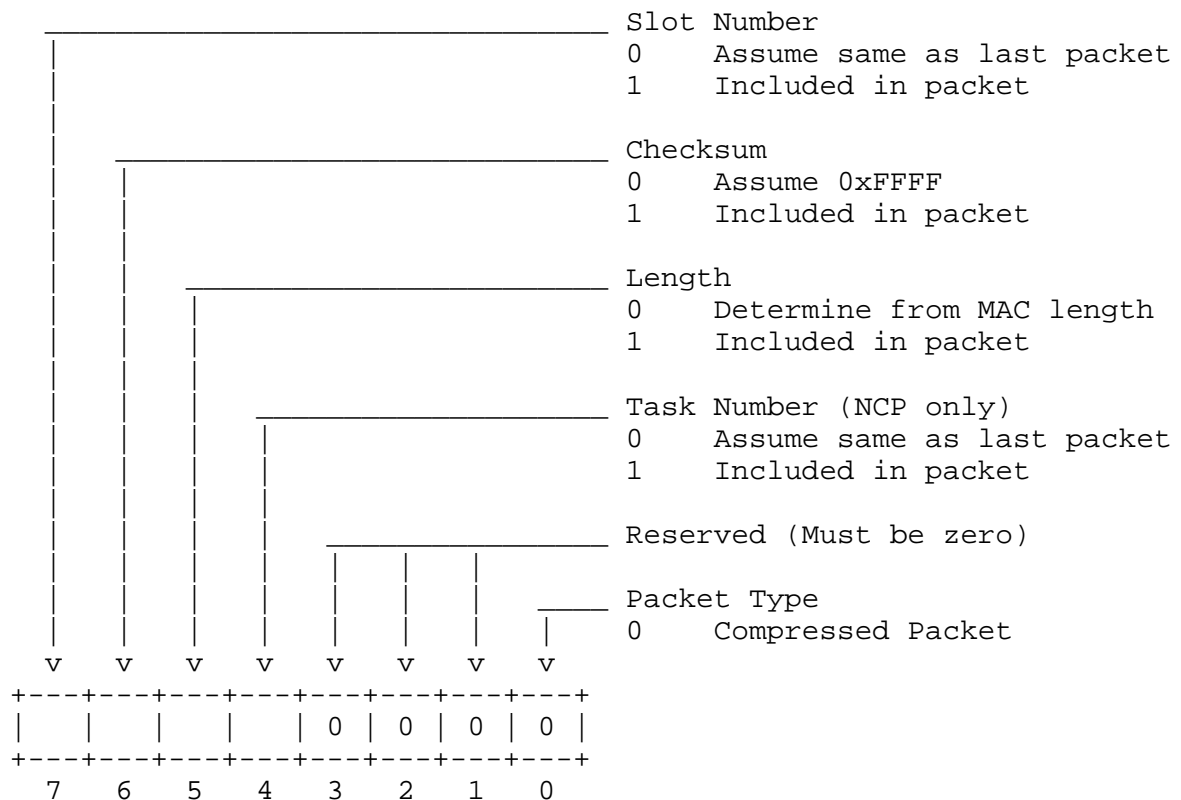
   Consider each flag in the flags octet, looking at the high order bits
   working toward the lower order bits. Each of the fields is optional,
   but if present will be found in the same order in the compressed
   packet header.

Slot Number

   The slot number flag indicates the slot number field is included in
   the compressed packet. The slot number field is one octet in length
   and specifies the number of the slot which corresponds to the Initial
   packet header. Slots are numbered starting at zero and continue to
   the maximum number of slots minus one.

   By default, slot compression is disabled. If negotiated, slot
   compression can be enabled for those slots which were created by the
   Unconfirmed Initial packet. When set to one (1), the slot number
   flag indicates the inclusion of the the slot number in the compressed
   packet. When set to zero (0), the slot number flag indicates the
   omission of the the slot number and specifies the use of the same
   slot number as for the last packet.

Implementation Note:

Slot compression MUST only be enabled in a receiver which can
account for all erroneous and discarded packets. When a packet
has been discarded, the slot number is indeterminate for future
packets. The decompressor MUST discard all further packets
until a slot number is received.

Checksum

When set to one (1), the checksum flag indicates the compressed
packet will include the 2 octet checksum. When set to zero (0),
this flag indicates the omission of the checksum and the decompressor
is to assume the checksum is 0xFFFF. Note that meaningful checksums
must be included in the packet with the checksum flag set to one (1).

Length

When set to one (1), the length flag indicates the inclusion of the
IPX data length field in the compressed packet. When set to zero
(0), the length flag indicates the omission of the IPX data length
field in the compressed packet.

This is the Length field from the original IPX packet header. It
specifies the length of IPX header and data in the packet prior to
compression. It does not include the CIPX compression field such as
flags, slot number, checksum, length field, or the NCP task number.
Note that it is preferable to determine the length field from the MAC
layer whenever possible, by subtracting the length of the compression
header fields and adding the length of the saved packet header.

Since every octet is significant over lower speed WAN links, an
optimization is used in the specification of the length. It can be
specified as a one, two or three octet field. If the length is in
the range 0 to 127, then it is specified as a one octet field. If
the length is in the range 128 to 16383, it is specified as a two
octet field in high to low order, with the first octet in the range
128 to 191. Otherwise, if the length is greater than 16383, the
first octet contains 192, and the second and third octets contain the
full length. (This scheme is extensible to 8 octets, but currently
is not required in the IPX protocol suite.)

```
+-+-+-+-+-+-+-+-+
|0|   length    |    length < 128
+-+-+-+-+-+-+-+-+
```

ONE OCTET LENGTH FIELD

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|           length          |    length < 16384
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TWO OCTET LENGTH FIELD

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 0 0 0 0 0 0|             length             |   length < 65535
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

THREE OCTET LENGTH FIELD

Task Number

   When set to one (1), the NCP task number flag indicates the NCP task
   number is included in the compressed packet (see NCP/IPX compression
   above). When set to zero (0), the NCP task number flag indicates the
   omission of the NCP task number in the compressed packet. When the
   NCP task number is not included in the compressed packet, we use the
   same NCP task number as that of last packet.

   Based upon the bits set in the flags octet, optional portions are
   included in the compressed IPX header. The minimum compressed IPX
   header contains only the Flags octet. All fields in the original IPX
   header have been compressed out of the header. The maximum
   compressed IPX header can include up to 7 octets, the Flags, Slot,
   Checksum (2 octets), and Length (3 octets) fields, or 8 octets if the
   NCP Task Number is included. The minimum and maximum compressed IPX
   packets are shown below. Header fields are one octet in length
   except where noted.

```
+--------+---------
| Flags  | DATA ...
|  0x00  |
+--------+---------
```

MINIMUM COMPRESSED IPX PACKET

```
+--------+--------+---------+---------+---------
| Flags  |  Slot  |Checksum | Length  | DATA ...
|  0xE0  | Number |2 octets |3 octets |
+--------+--------+---------+---------+---------
```

MAXIMUM COMPRESSED IPX PACKET

```
+--------+--------+---------+---------+--------+---------
| Flags  |  Slot  |Checksum | Length  |NCP Task| DATA ...
|  0xF0  | Number |2 octets |3 octets | Number |
+--------+--------+---------+---------+--------+---------
```

MAXIMUM COMPRESSED NCP/IPX PACKET

Regular Packet

   The Regular packet type designates an IPX packet for which no
   compression is desired. This type of packet is sent when a packet
   cannot be compressed, or a decision is made not to compress it.

```
        7   6   5   4   3   2   1   0
      +---+---+---+---+---+---+---+---+
      | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
      +---+---+---+---+---+---+---+---+
        ^   ^   ^   ^   ^   ^   ^   ^
        |   |   |   |   |   |   |   |
        |   |   |   |   |___|___|___|___ Packet Type
        |   |   |   |                    1    Regular
        |   |   |   |
        |__ |__ |__ |_____ Reserved (must be zero)
```

   The Regular packet is rarely sent. Usually, the Regular packet is
   sent when there is not enough memory for the overhead of a new
   compression slot. Also, this type is included for future unforeseen
   changes to the IPX protocol which defeat the effectiveness of
   compression.

      Implementation Note:

         The Regular Packet can be used for packets that are sporadic,
         which are not worth setting-up a compression slot. This may be
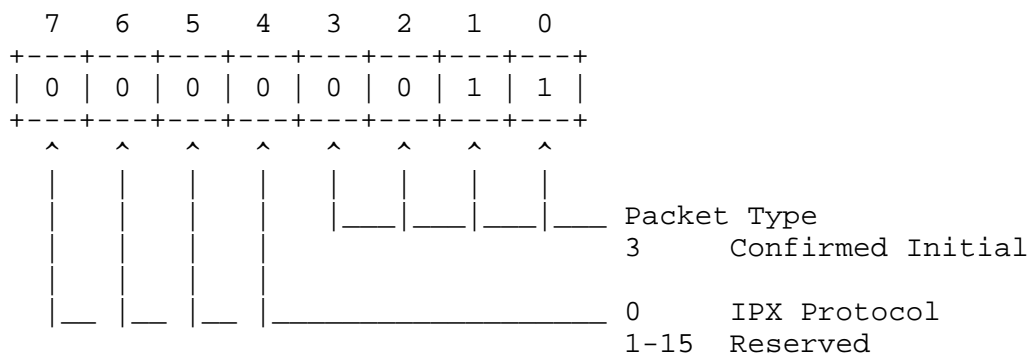
       hard to determine for specific protocols. Various methods such
       as hold-down and least-recently-used timers are currently being
       used.

       On receipt, the 1 octet header is simply removed and the packet
       passed up to IPX.

       The entire IPX packet follows the single Flags octet. Note for a
       Regular Packet (not compressed or uncompressed), the slot number
       field is not included.

Confirmed Initial Packet

    The Confirmed Initial packet type is used by the compressor to inform
    the decompressor of the original packet header which will be used for
    subsequent compression, and to request Confirmation. The high order
    4 bits are reserved for expansion to support additional protocols.

```
        7   6   5   4   3   2   1   0
      +---+---+---+---+---+---+---+---+
      | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
      +---+---+---+---+---+---+---+---+
        ^   ^   ^   ^   ^   ^   ^   ^
        |   |   |   |   |   |   |   |
        |   |   |   |   |___|___|___|___  Packet Type
        |   |   |   |                     3     Confirmed Initial
        |   |   |   |
        |__ |__ |__ |_____   0     IPX Protocol
                                          1-15  Reserved
```

    This type of packet is sent to inform the receiver to associate the
    IPX packet header with a slot number. This packet is sent each time
    a different header format is sent for a given slot, or when the
    sender has not received a Confirmation Packet from the receiver.

    The Flags octet lower 4 bits indicate the Confirmed Initial CIPX
    packet type. The high order 4 bits are reserved for expansion to
    support additional protocols. The Flags octet is always followed by
    the Slot Number and an ID field. The ID field is one octet in
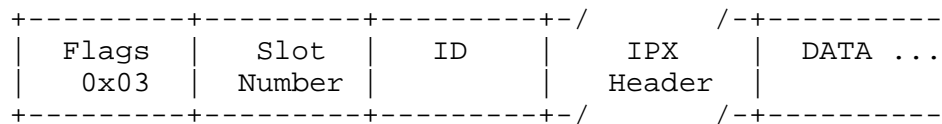   length.

    For each slot, the ID will increment with every new header sent.
    Different slots may have the same ID. The combination of slot and ID
    uniquely identify a header. In practice, the ID octet can be any
    number which is unique for a "reasonably long period" of time. A
    reasonably long period is a function of transmission speed, round
    trip delays, and network load. There must be very little chance of
    duplicate slot and ID combinations within this period. Otherwise,

there is ambiguity as to which header is being identified.

   Implementation Note:

      There is no requirement to hold or resend the Confirmed Initial
      packet until confirmation. When a new packet with the same IPX
      header is to be sent, another Confirmed Initial packet should
      be sent using the same slot, the same ID, and the new packet
      data.

      When a new packet with a different IPX header is to be sent, it
      may be sent using a slot which has not received confirmation.
      A Confirmed Initial packet is sent with the same slot, an
      incremented ID, and the new packet data. Assuming a least-
      recently-used policy for selecting a slot for a new IPX header,
      this provides the ability to reuse slots when a Confirmed
      Initial packet has been sent but not confirmed.

```
          +---------+---------+---------+-/       /-+----------
          |  Flags  |  Slot   |   ID    |   IPX     | DATA ...
          |  0x03   | Number  |         | Header    |
          +---------+---------+---------+-/       /-+----------
```
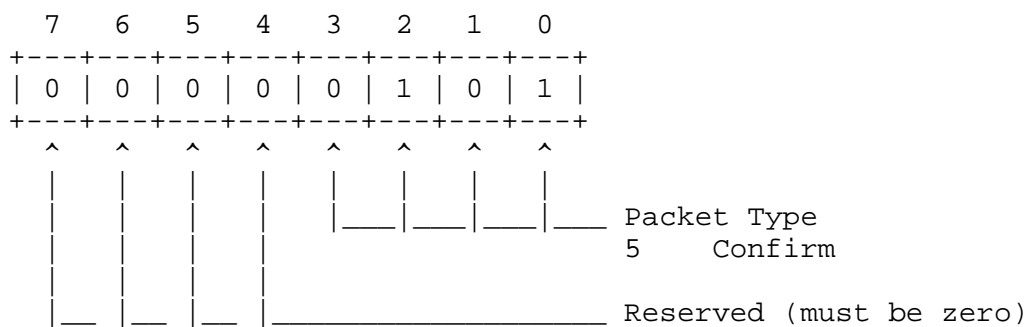
CONFIRMED INITIAL PACKET

   Note that a Confirmed Initial header is followed by a complete IPX
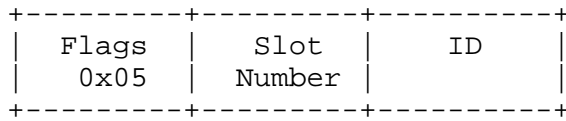   packet.

Confirm Packet

   The Confirm packet type is used by the decompressor to tell the
   compressor that it has received the Confirmed Initial packet.

   When the compressor receives this, it can start sending Compressed
   frames.

```
       7   6   5   4   3   2   1   0
     +---+---+---+---+---+---+---+---+
     | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
     +---+---+---+---+---+---+---+---+
       ^   ^   ^   ^   ^   ^   ^   ^
       |   |   |   |   |   |   |   |
       |   |   |   |   |___|___|___|___ Packet Type
       |   |   |   |               5     Confirm
       |   |   |   |
       |__ |__ |__ |_____ Reserved (must be zero)
```

   A Confirm Packet is exactly 3 octets in length. It consists of the

Flags, Slot Number and ID fields. The Slot Number field contains the
number of the slot which is being acknowledged. The ID field
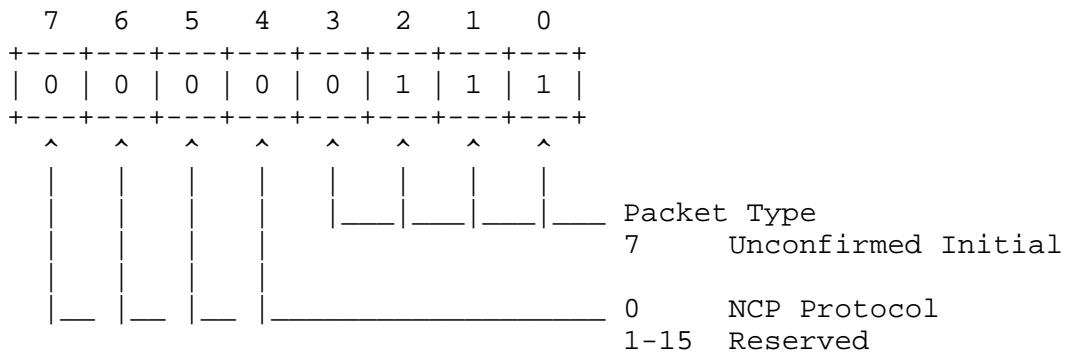contains the ID of the Confirmed Initial Packet which is being
acknowledged.

```
        +---------+---------+----------+
        |  Flags  |  Slot   |   ID     |
        |  0x05   | Number  |          |
        +---------+---------+----------+
```

CONFIRM PACKET

Unconfirmed Initial Packet

   The Unconfirmed Initial packet type is used by the compressor to
   inform the decompressor of the original packet header which will be
   used for subsequent compression while not requesting confirmation.

   After sending an Unconfirmed Initial packet, the compressor may
   immediately send Compressed packets without confirmation.

```
         7   6   5   4   3   2   1   0
        +---+---+---+---+---+---+---+---+
        | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
        +---+---+---+---+---+---+---+---+
          ^   ^   ^   ^   ^   ^   ^   ^
          |   |   |   |   |   |   |   |
          |   |   |   |   |___|___|___|___ Packet Type
          |   |   |   |               7    Unconfirmed Initial
          |   |   |   |
          |__ |__ |__ |_____ 0    NCP Protocol
                                        1-15 Reserved
```

   This type of packet is sent to inform the receiver to associate the
   IPX packet header with a slot number. This packet is sent each time
   a different header format is sent for a given slot.

   The Flags octet lower 4 bits indicate the Unconfirmed Initial CIPX
   packet type. The high order 4 bits are reserved for expansion to
   support additional protocols. The Flags octet is always followed by
   the Slot Number.

```
        +---------+---------+-/       /-+-/       /-+---------
        |  Flags  |  Slot   |  IPX    | NCP     | NCP
        |  0x07   | Number  | Header  | Header  | DATA ...
        +---------+---------+-/       /-+-/       /-+---------
```
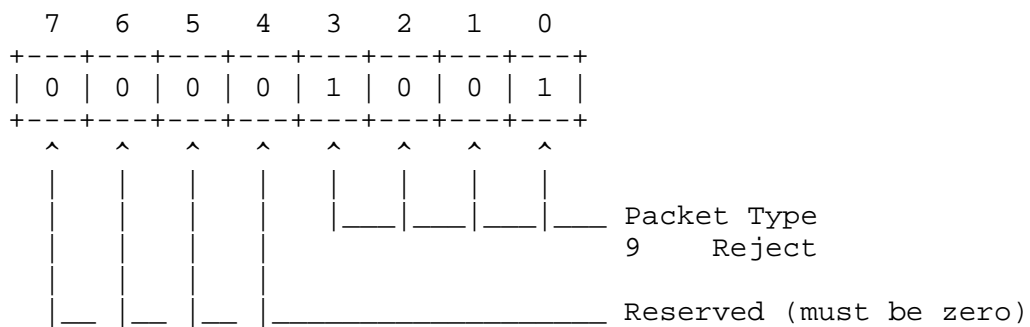
UNCONFIRMED INITIAL PACKET

    Note that an Unconfirmed Initial header is followed by a complete IPX
    packet.

Reject Packet

    The Reject packet type is used by the decompressor to tell the
    compressor that it has received a CIPX packet with a header which it
    does not support. This is provided to regulate future extensions to
    CIPX.

```
           7   6   5   4   3   2   1   0
         +---+---+---+---+---+---+---+---+
         | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
         +---+---+---+---+---+---+---+---+
           ^   ^   ^   ^   ^   ^   ^   ^
           |   |   |   |   |   |   |   |
           |   |   |   |   |___|___|___|___  Packet Type
           |   |   |   |                     9    Reject
           |   |   |   |
           |__ |__ |__ |_____ Reserved (must be zero)
```

    A Reject Packet is exactly 3 octets in length. It consists of the
    Flags, Slot Number and Rejected Flags fields.

    The Slot Number field contains the number of the slot of the packet
    which is being rejected. Since the actual packet type may be unknown
    or misunderstood, this field actually contains the second octet of
    the rejected packet. In the normal case of a known CIPX packet type,
    this will be the slot number of an initial packet.

    The Rejected Flags field contains the first octet of the packet being
    rejected. The packet type field is left untouched. Any flags which
    are correctly recognized should be cleared. The remaining flags
    indicate specific features that are being rejected. This information
    should be sufficient for implementations to adjust the use of certain
    packet types or dependent flags.

        Implementation Note:

            The Flags value of 0xFF is not a valid CIPX packet type.
            Hence, such a packet type should be recognized as a standard
            IPX header and forwarded without CIPX processing to the
            appropriate routines. Under no circumstances should a Flags
            value of 0xFF be rejected in a Reject Packet.

```
                +---------+---------+----------+
                |  Flags  |  Slot   | Rejected |
                |  0x09   | Number  |  Flags   |
                +---------+---------+----------+
```

           REJECT PACKET

Compression Negotiation over PPP Links

   For PPP links [5], the use of header compression can be negotiated by
   IPXCP [6]. By default, no compression is enabled.

   The IPX-Compression-Protocol Configuration Option is used to indicate
   the ability to receive compressed packets. Each end of the link must
   separately request this option if bi-directional compression is
   desired.

   The PPP Protocol field is set to the same value as the usual IPX
   packets, and all IPX packets sent over the link MUST conform to the
   compressed format.

   A summary of the IPX-Compression-Protocol Configuration Option format
   to negotiate Telebit IPX header compression (CIPX) 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    |   IPX-Compression-Protocol    |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |  Max-Slot-Id  |    Options    |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Type

       3

   Length

       6

   IPX-Compression-Protocol

       0002 (hex) for Telebit Compressed IPX headers (CIPX).

    Max-Slot-Id

       The Max-Slot-Id field is one octet and indicates the maximum

slot identifier. This is one less than the actual number of
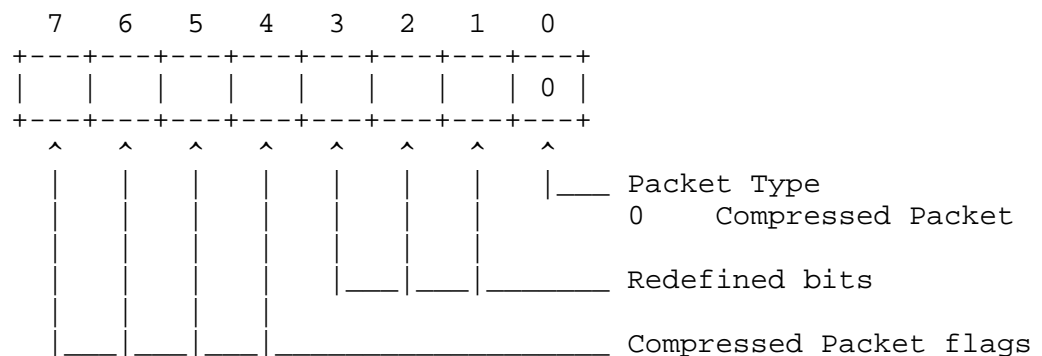slots; the slot identifier has values from zero to Max-Slot-
Id.

Options

   The Options field is one octet, and is comprised of the "logical or"
   of the following values:

   0 No options.

   1 The slot identifer may be compressed.

      The slot identifier must not be compressed if there is no
      ability for the PPP link level to indicate an error in
      reception to the decompression module. Synchronization after
      errors depends on receiving a packet with the slot identifier.

   2 Redefine Compressed Packet type bits 1-3.

      It was noted earlier that packet types have been chosen such
      that only the Compressed Packet type is an even number value
      with the lowest order bit of zero. All other packet types are
      odd values with a lowest order bit of one. The reason for this
      assignment was to make it possible to determine the Compressed
      Packet type by examining only one bit. This make it possible
      to use all the other 7 bits to indicate status in the
      Compressed Packet. The 7 bits are composed of the upper 4 bits
      which are permanently defined to indicate packet dependent
      flags, plus bits 1-3 which are otherwise part of the Packet
      Type. The upper 4 bits are defined above. The redefinition of
      bits 1-3 of the Compressed Packet type is left for future
      expansion.

```
            7   6   5   4   3   2   1   0
          +---+---+---+---+---+---+---+---+
          |   |   |   |   |   |   |   | 0 |
          +---+---+---+---+---+---+---+---+
            ^   ^   ^   ^   ^   ^   ^   ^
            |   |   |   |   |   |   |   |___ Packet Type
            |   |   |   |   |   |   |        0    Compressed Packet
            |   |   |   |   |   |   |
            |   |   |   |   |___|___|_____ Redefined bits
            |   |   |   |
            |___|___|___|_____ Compressed Packet flags
```

      By default, this feature in not enabled and this flag is
      set to zero. When this flag is set to one, it indicates

the desire to use this feature.

Compression Negotiation over IPXWAN Links

   "IPXWAN" is the protocol Novell uses to exchange necessary router
   to router information prior to exchanging standard IPX routing
   information and traffic over WAN datalinks [7]. To negotiate the
   Telebit compression option, we use Novell's allocated option number
   for CIPX (00) in the IPXWAN timer request/response packet.

   The Timer Request packet contains the following Telebit compression
   option:

     WOption Number   80   - Define compression type
     WAccept Option   01   - 0=No, 1=Yes, 3=N/A
     WOption Data Len   00 03   - Length of option
     WOption Data   00   - Telebit's compression (CIPX)
     WOption Data   XX   - Compression options
     WOption Data   NN   - Compression slots

   Where the WOption Data fields are:

     00   Telebit's compression option described in this
          document (CIPX).

     XX   Compression options as defined below:

          0x01   Compress slot ID when possible
          0x02   Redefine Compressed Packet type bits 1-3.

     NN   The requested # of compression slots.

   Accept Option (for compression type) must be set to YES if the
   option is supported and NO if the option is not supported. A Timer
   Response must respond with only one header compression type set to
   YES.

   The Timer Response packet that accepts the option will look like
   this:

     WOption Number   80   - Define compression type
     WAccept Option   01   - 0=No, 1=Yes, 3=N/A
     WOption Data Len   00 03   - Length of option
     WOption Data   00   - Telebit's compression (CIPX)
     WOption Data   XX   - Compression options
     WOption Data   NN   - Compression slots

Where the WOption Data fields are:

    00   Telebit's compression option described in this
           document (CIPX).

    XX   Compression options as defined below:

          0x01   Compress slot ID when possible
          0x02   Redefine Compressed Packet type bits 1-3.

    NN   The negotiated # of slots (The lower of each side's
           requested number of slots)

IPX packets (except of course IPXWAN packets) are not sent over the
link until the IPXWAN negotiations are completed. Once IPXWAN
negotiations are completed, regular IPX packets can be sent over the
link.

If both ends of the link agree on the compression options, then the
IPX packets are sent using the specified options. If either end of
the link does not accept a compression option, then this compression
option will not be used. Compression will be done using any
remaining options. Options, by definition, are not required.
Implementations MUST support CIPX without any options.

It is the responsibility of the router sending the IPXWAN Timer
Response to inform the other router of the options that will be used.
The Timer Response MUST contain a subset of the options received in a
Timer Request.

To be clear, IPXWAN is used to set up a symmetrical compression link.
Compression is configured identically in both directions. Each end
will use the same number of slots and same compression options. It
is illegal for link ends to use different number of slots or
different options.

IPX Compression Performance

The performance of this algorithm will depend on the number of active
connections and the number of slots negotiated. If the number of
slots is greater than the number of connections, the hit rate should
be very high giving a very high compression ratio. The performance
also depends on the average size of the IPX packets. If the average
size of packets is small, then compression will result in a more
noticeable performance improvement.

$$\text{Compression ratio} = \frac{avg\_data\_len + uncomp\_header\_len}{avg\_data\_len + avg\_comp\_header\_len}$$

   Where 'avg_data_len' is the average length of data in the IPX packet,
   and 'uncomp_head_len' is the uncompressed header length which is
   fixed at 30 octets. Where 'avg_comp_header_len' is the average
   length of the compressed IPX header. The length of the minimum
   compressed IPX header is 1 octet. The length of the maximum
   compressed NCP/IPX header is 8 octets (including the NCP task
   number), but since no implementation yet sends packets with a length
   greater than 16K, 7 octets is the commonly encountered maximum.
   Perhaps a reasonable 'avg_comp_header_len' is 2, assuming the
   inclusion of the flag and slot number octets.

   The maximum length of the data in an IPX packet is 546 octets (576
   octets - 30 octet IPX header), although newer implementations may
   send packets of up to 4096 octets. The minimum length of the data in
   an IPX packet is 1 octet. Within the normal distribution of small
   NCP packets, perhaps a reasonable 'avg_data_len' is 26 octets.

$$\text{Minimal Compression} = \frac{546 + 30}{546 + 6} = 1.04$$

$$\text{Maximal Compression} = \frac{1 + 30}{1 + 1} = 15.50$$

$$\text{Likely Compression} = \frac{26 + 30}{26 + 2} = 2.00$$

Security Considerations

   IPX provides some security features, which are fully applicable to
   CIPX. CIPX does not significantly alter the basic security of IPX.

References

   [1] Novell Inc., "IPX Router Specification", September 1992, Part
       Number: 107-000029-001

   [2] Jacobson, Van, "Compressing TCP/IP Headers for Low-Speed Serial
       Links", RFC 1144, February 1990

   [3] CCITT Recommendation V.42bis Error Correcting Procedures for DCEs
       using Error Correction Procedures

   [4] ISO 7776, Information Processing Systems - Data Communication -
       High Level Data Link Control Procedures - Description of the X.25
       LAPB-Compatible DTE Data Link Procedures

   [5] Simpson, W. A., "The Point-to-Point Protocol (PPP)", RFC 1548,
       December 1993

   [6] Simpson, W. A., "The PPP Internet Packet Exchange Control
       Protocol (IPXCP)", RFC 1552, December 1993

   [7] Allen, Michael, "Novell IPX Over Various WAN Media [IPXWAN]",
       RFC 1551, December 1993

Chair's Address
     Fred Baker
     Advanced Computer Communications
     315 Bollay Drive
     Santa Barbara, California 93117

     EMail: fbaker@acc.com

Authors' Addresses

        Saroop Mathur
        Telebit Corp.
        1315 Chesapeake Terrace
        Sunnyvale, CA 94089-1100

        EMail: mathur@telebit.com

        Mark S. Lewis
        Telebit Corp.
        1315 Chesapeake Terrace
        Sunnyvale, CA 94089-1100

        EMail: Mark.S.Lewis@telebit.com