                   The Point-to-Point Protocol (PPP)

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Abstract

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

      1. A method for encapsulating multi-protocol datagrams.

      2. A Link Control Protocol (LCP) for establishing, configuring,
         and testing the data-link connection.

      3. A family of Network Control Protocols (NCPs) for establishing
         and configuring different network-layer protocols.

   This document defines the PPP organization and methodology, and the
   PPP encapsulation, together with an extensible option negotiation
   mechanism which is able to negotiate a rich assortment of
   configuration parameters and provides additional management
   functions. The PPP Link Control Protocol (LCP) is described in terms
   of this mechanism.

Table of Contents

1. Introduction

   The Point-to-Point Protocol is designed for simple links which
   transport packets between two peers. These links provide full-duplex
   simultaneous bi-directional operation, and are assumed to deliver
   packets in order. It is intended that PPP provide a common solution
   for easy connection of a wide variety of hosts, bridges and routers
   [1].

   Encapsulation

      The PPP encapsulation provides for multiplexing of different
      network-layer protocols simultaneously over the same link. The
      PPP encapsulation has been carefully designed to retain
      compatibility with most commonly used supporting hardware.

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

      To support high speed implementations, the default encapsulation
      uses only simple fields, only one of which needs to be examined
      for demultiplexing. The default header and information fields
      fall on 32-bit boundaries, and the trailer may be padded to an
      arbitrary boundary.

   Link Control Protocol

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

   Network Control Protocols

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

respective network-layer protocols. These NCPs are defined in
companion documents.

Configuration

It is intended that PPP links be easy to configure. By design,
the standard defaults handle all common configurations. The
implementor can specify improvements to the default configuration,
which are automatically communicated to the peer without operator
intervention. Finally, the operator may explicitly configure
options for the link which enable the link to operate in
environments where it would otherwise be impossible.

This self-configuration is implemented through an extensible
option negotiation mechanism, wherein each end of the link
describes to the other its capabilities and requirements.
Although the option negotiation mechanism described in this
document is specified in terms of the Link Control Protocol (LCP),
the same facilities are designed to be used by other control
protocols, especially the family of NCPs.

1.1. 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.

1.2. Terminology

   This document frequently uses the following terms:

   datagram   The unit of transmission in the network layer (such as IP).
              A datagram may be encapsulated in one or more packets
              passed to the data link layer.

   frame   The unit of transmission at the data link layer. A frame
              may include a header and/or a trailer, along with some
              number of units of data.

   packet   The basic unit of encapsulation, which is passed across the
              interface between the network layer and the data link
              layer. A packet is usually mapped to a frame; the
              exceptions are when data link layer fragmentation is being
              performed, or when multiple packets are incorporated into a
              single frame.

   peer   The other end of the point-to-point link.

   silently discard
              The implementation discards the packet without further
              processing. The implementation SHOULD provide the
              capability of logging the error, including the contents of
              the silently discarded packet, and SHOULD record the event
              in a statistics counter.

2. PPP Encapsulation

   The PPP encapsulation is used to disambiguate multiprotocol
   datagrams. This encapsulation requires framing to indicate the
   beginning and end of the encapsulation. Methods of providing framing
   are specified in companion documents.

   A summary of the PPP encapsulation is shown below. The fields are
   transmitted from left to right.

```
        +----------+-------------+---------+
        | Protocol | Information | Padding |
        | 8/16 bits|      *      |    *    |
        +----------+-------------+---------+
```

   Protocol Field

      The Protocol field is one or two octets, and its value identifies
      the datagram encapsulated in the Information field of the packet.
      The field is transmitted and received most significant octet
      first.

      The structure of this field is consistent with the ISO 3309
      extension mechanism for address fields. All Protocols MUST be
      odd; the least significant bit of the least significant octet MUST
      equal "1". Also, all Protocols MUST be assigned such that the
      least significant bit of the most significant octet equals "0".
      Frames received which don't comply with these rules MUST be
      treated as having an unrecognized Protocol.

      Protocol field values in the "0***" to "3***" range identify the
      network-layer protocol of specific packets, and values in the
      "8***" to "b***" range identify packets belonging to the
      associated Network Control Protocols (NCPs), if any.

      Protocol field values in the "4***" to "7***" range are used for
      protocols with low volume traffic which have no associated NCP.
      Protocol field values in the "c***" to "f***" range identify
      packets as link-layer Control Protocols (such as LCP).

Up-to-date values of the Protocol field are specified in the most
recent "Assigned Numbers" RFC [2]. This specification reserves
the following values:

Value (in hex)   Protocol Name

0001   Padding Protocol
0003 to 001f   reserved (transparency inefficient)
007d   reserved (Control Escape)
00cf   reserved (PPP NLPID)
00ff   reserved (compression inefficient)

8001 to 801f   unused
807d   unused
80cf   unused
80ff   unused

c021   Link Control Protocol
c023   Password Authentication Protocol
c025   Link Quality Report
c223   Challenge Handshake Authentication Protocol

Developers of new protocols MUST obtain a number from the Internet
Assigned Numbers Authority (IANA), at IANA@isi.edu.

Information Field

   The Information field is zero or more octets. The Information
   field contains the datagram for the protocol specified in the
   Protocol field.

   The maximum length for the Information field, including Padding,
   but not including the Protocol field, is termed the Maximum
   Receive Unit (MRU), which defaults to 1500 octets. By
   negotiation, consenting PPP implementations may use other values
   for the MRU.

Padding

   On transmission, the Information field MAY be padded with an
   arbitrary number of octets up to the MRU. It is the
   responsibility of each protocol to distinguish padding octets from
   real information.
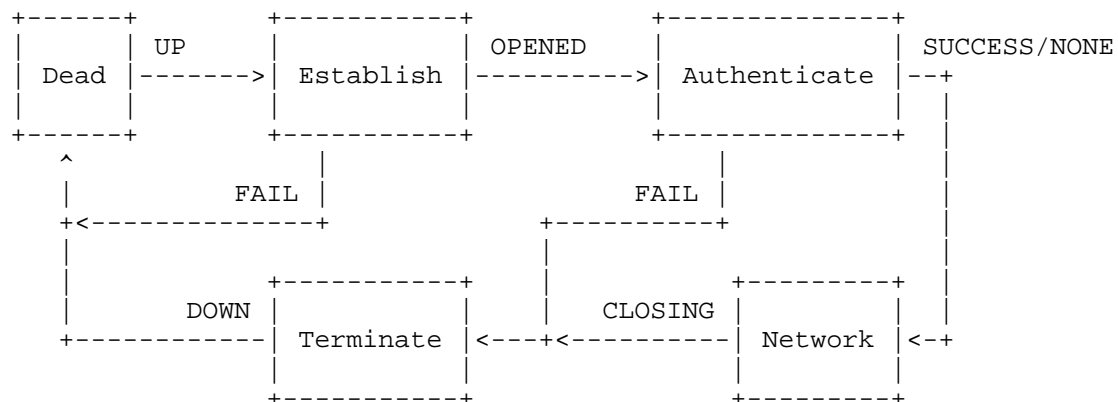
3. PPP Link Operation

3.1. Overview

   In order to establish communications over a point-to-point link, each
   end of the PPP link MUST first send LCP packets to configure and test
   the data link. After the link has been established, the peer MAY be
   authenticated.

   Then, PPP MUST send NCP packets to choose and configure one or more
   network-layer protocols. Once each of the chosen network-layer
   protocols has been configured, datagrams from each network-layer
   protocol can be sent over the link.

   The link will remain configured for communications until explicit LCP
   or NCP packets close the link down, or until some external event
   occurs (an inactivity timer expires or network administrator
   intervention).

3.2. Phase Diagram

   In the process of configuring, maintaining and terminating the
   point-to-point link, the PPP link goes through several distinct
   phases which are specified in the following simplified state diagram:

```
+------+               +-----------+              +--------------+
|      | UP            |           | OPENED       |              | SUCCESS/NONE
| Dead |------->| Establish |---------->| Authenticate |--+
|      |               |           |              |              |  |
+------+               +-----------+              +--------------+  |
  ^                          |                          |          |
  |             FAIL |                     FAIL |          |
+<-------------+              +----------+   |
  |                          |                          |          |
  |                          |              +----------+   |          |
  |             DOWN |              |              |  CLOSING |   |          |
+------------| Terminate |<---+<----------| Network  |<-+
  |              |              |              |          |
+----------+              +----------+
```

   Not all transitions are specified in this diagram. The following
   semantics MUST be followed.

3.3. Link Dead (physical-layer not ready)

   The link necessarily begins and ends with this phase. When an
   external event (such as carrier detection or network administrator
   configuration) indicates that the physical-layer is ready to be used,
   PPP will proceed to the Link Establishment phase.

   During this phase, the LCP automaton (described later) will be in the
   Initial or Starting states. The transition to the Link Establishment
   phase will signal an Up event to the LCP automaton.

   Implementation Note:

      Typically, a link will return to this phase automatically after
      the disconnection of a modem. In the case of a hard-wired link,
      this phase may be extremely short -- merely long enough to detect
      the presence of the device.

3.4. Link Establishment Phase

   The Link Control Protocol (LCP) is used to establish the connection
   through an exchange of Configure packets. This exchange is complete,
   and the LCP Opened state entered, once a Configure-Ack packet
   (described later) has been both sent and received.

   All Configuration Options are assumed to be at default values unless
   altered by the configuration exchange. See the chapter on LCP
   Configuration Options for further discussion.

   It is important to note that only Configuration Options which are
   independent of particular network-layer protocols are configured by
   LCP. Configuration of individual network-layer protocols is handled
   by separate Network Control Protocols (NCPs) during the Network-Layer
   Protocol phase.

   Any non-LCP packets received during this phase MUST be silently
   discarded.

   The receipt of the LCP Configure-Request causes a return to the Link
   Establishment phase from the Network-Layer Protocol phase or
   Authentication phase.

3.5. Authentication Phase

   On some links it may be desirable to require a peer to authenticate
   itself before allowing network-layer protocol packets to be
   exchanged.

   By default, authentication is not mandatory. If an implementation
   desires that the peer authenticate with some specific authentication
   protocol, then it MUST request the use of that authentication
   protocol during Link Establishment phase.

   Authentication SHOULD take place as soon as possible after link
   establishment. However, link quality determination MAY occur
   concurrently. An implementation MUST NOT allow the exchange of link
   quality determination packets to delay authentication indefinitely.

   Advancement from the Authentication phase to the Network-Layer
   Protocol phase MUST NOT occur until authentication has completed. If
   authentication fails, the authenticator SHOULD proceed instead to the
   Link Termination phase.

   Only Link Control Protocol, authentication protocol, and link quality
   monitoring packets are allowed during this phase. All other packets
   received during this phase MUST be silently discarded.

   Implementation Notes:

      An implementation SHOULD NOT fail authentication simply due to
      timeout or lack of response. The authentication SHOULD allow some
      method of retransmission, and proceed to the Link Termination
      phase only after a number of authentication attempts has been
      exceeded.

      The implementation responsible for commencing Link Termination
      phase is the implementation which has refused authentication to
      its peer.

3.6. Network-Layer Protocol Phase

   Once PPP has finished the previous phases, each network-layer
   protocol (such as IP, IPX, or AppleTalk) MUST be separately
   configured by the appropriate Network Control Protocol (NCP).

   Each NCP MAY be Opened and Closed at any time.

Implementation Note:

   Because an implementation may initially use a significant amount
   of time for link quality determination, implementations SHOULD
   avoid fixed timeouts when waiting for their peers to configure a
   NCP.

After a NCP has reached the Opened state, PPP will carry the
corresponding network-layer protocol packets. Any supported
network-layer protocol packets received when the corresponding NCP is
not in the Opened state MUST be silently discarded.

Implementation Note:

   While LCP is in the Opened state, any protocol packet which is
   unsupported by the implementation MUST be returned in a Protocol-
   Reject (described later). Only protocols which are supported are
   silently discarded.

During this phase, link traffic consists of any possible combination
of LCP, NCP, and network-layer protocol packets.

3.7. Link Termination Phase

PPP can terminate the link at any time. This might happen because of
the loss of carrier, authentication failure, link quality failure,
the expiration of an idle-period timer, or the administrative closing
of the link.

LCP is used to close the link through an exchange of Terminate
packets. When the link is closing, PPP informs the network-layer
protocols so that they may take appropriate action.

After the exchange of Terminate packets, the implementation SHOULD
signal the physical-layer to disconnect in order to enforce the
termination of the link, particularly in the case of an
authentication failure. The sender of the Terminate-Request SHOULD
disconnect after receiving a Terminate-Ack, or after the Restart
counter expires. The receiver of a Terminate-Request SHOULD wait for
the peer to disconnect, and MUST NOT disconnect until at least one
Restart time has passed after sending a Terminate-Ack. PPP SHOULD
proceed to the Link Dead phase.

Any non-LCP packets received during this phase MUST be silently
discarded.

   Implementation Note:

      The closing of the link by LCP is sufficient. There is no need
      for each NCP to send a flurry of Terminate packets. Conversely,
      the fact that one NCP has Closed is not sufficient reason to cause
      the termination of the PPP link, even if that NCP was the only NCP
      currently in the Opened state.

4. The Option Negotiation Automaton

   The finite-state automaton is defined by events, actions and state
   transitions. Events include reception of external commands such as
   Open and Close, expiration of the Restart timer, and reception of
   packets from a peer. Actions include the starting of the Restart
   timer and transmission of packets to the peer.

   Some types of packets -- Configure-Naks and Configure-Rejects, or
   Code-Rejects and Protocol-Rejects, or Echo-Requests, Echo-Replies and
   Discard-Requests -- are not differentiated in the automaton
   descriptions. As will be described later, these packets do indeed
   serve different functions. However, they always cause the same
   transitions.

   Events   Actions

   Up   = lower layer is Up   tlu = This-Layer-Up
   Down = lower layer is Down   tld = This-Layer-Down
   Open = administrative Open   tls = This-Layer-Started
   Close= administrative Close   tlf = This-Layer-Finished

   TO+ = Timeout with counter > 0   irc = Initialize-Restart-Count
   TO- = Timeout with counter expired   zrc = Zero-Restart-Count

   RCR+ = Receive-Configure-Request (Good) scr = Send-Configure-Request
   RCR- = Receive-Configure-Request (Bad)
   RCA = Receive-Configure-Ack   sca = Send-Configure-Ack
   RCN = Receive-Configure-Nak/Rej   scn = Send-Configure-Nak/Rej

   RTR = Receive-Terminate-Request   str = Send-Terminate-Request
   RTA = Receive-Terminate-Ack   sta = Send-Terminate-Ack

   RUC = Receive-Unknown-Code   scj = Send-Code-Reject
   RXJ+ = Receive-Code-Reject (permitted)
       or Receive-Protocol-Reject
   RXJ- = Receive-Code-Reject (catastrophic)
       or Receive-Protocol-Reject
   RXR = Receive-Echo-Request   ser = Send-Echo-Reply
       or Receive-Echo-Reply
       or Receive-Discard-Request

4.1. State Transition Table

   The complete state transition table follows. States are indicated
   horizontally, and events are read vertically. State transitions and
   actions are represented in the form action/new-state. Multiple
   actions are separated by commas, and may continue on succeeding lines
   as space requires; multiple actions may be implemented in any
   convenient order. The state may be followed by a letter, which
   indicates an explanatory footnote. The dash ('-') indicates an
   illegal transition.

```
       | State
       |   0         1        2          3         4          5
Events | Initial  Starting  Closed    Stopped   Closing    Stopping
-------+---------------------------------------------------------------
 Up    |   2      irc,scr/6    -          -         -          -
 Down  |   -         -         0        tls/1       0          1
 Open  | tls/1       1      irc,scr/6     3r        5r         5r
 Close |   0       tlf/0       2          2         4          4
       |
  TO+  |   -         -         -          -       str/4      str/5
  TO-  |   -         -         -          -       tlf/2      tlf/3
       |
 RCR+  |   -         -       sta/2  irc,scr,sca/8   4          5
 RCR-  |   -         -       sta/2  irc,scr,scn/6   4          5
 RCA   |   -         -       sta/2      sta/3       4          5
 RCN   |   -         -       sta/2      sta/3       4          5
       |
 RTR   |   -         -       sta/2      sta/3     sta/4      sta/5
 RTA   |   -         -         2          3       tlf/2      tlf/3
       |
 RUC   |   -         -       scj/2      scj/3     scj/4      scj/5
 RXJ+  |   -         -         2          3         4          5
 RXJ-  |   -         -       tlf/2      tlf/3     tlf/2      tlf/3
       |
 RXR   |   -         -         2          3         4          5
```

```
        | State
        |    6          7          8           9
Events| Req-Sent   Ack-Rcvd   Ack-Sent     Opened
------+------------------------------------------
 Up   |    -          -          -            -
 Down |    1          1          1         tld/1
 Open |    6          7          8           9r
 Close|irc,str/4 irc,str/4 irc,str/4 tld,irc,str/4
      |
  TO+ |   scr/6      scr/6      scr/8         -
  TO- |   tlf/3p     tlf/3p     tlf/3p        -
      |
 RCR+ |   sca/8    sca,tlu/9    sca/8    tld,scr,sca/8
 RCR- |   scn/6      scn/7      scn/6    tld,scr,scn/6
 RCA  |   irc/7      scr/6x   irc,tlu/9   tld,scr/6x
 RCN  |irc,scr/6    scr/6x   irc,scr/8   tld,scr/6x
      |
 RTR  |   sta/6      sta/6      sta/6    tld,zrc,sta/5
 RTA  |    6          6          8        tld,scr/6
      |
 RUC  |   scj/6      scj/7      scj/8       scj/9
 RXJ+ |    6          6          8           9
 RXJ- |   tlf/3      tlf/3      tlf/3    tld,irc,str/5
      |
 RXR  |    6          7          8          ser/9
```

   The states in which the Restart timer is running are identifiable by
   the presence of TO events. Only the Send-Configure-Request, Send-
   Terminate-Request and Zero-Restart-Count actions start or re-start
   the Restart timer. The Restart timer is stopped when transitioning
   from any state where the timer is running to a state where the timer
   is not running.

   The events and actions are defined according to a message passing
   architecture, rather than a signalling architecture. If an action is
   desired to control specific signals (such as DTR), additional actions
   are likely to be required.

   [p]   Passive option; see Stopped state discussion.

   [r]   Restart option; see Open event discussion.

   [x]   Crossed connection; see RCA event discussion.

4.2. States

   Following is a more detailed description of each automaton state.

   Initial

      In the Initial state, the lower layer is unavailable (Down), and
      no Open has occurred. The Restart timer is not running in the
      Initial state.

   Starting

      The Starting state is the Open counterpart to the Initial state.
      An administrative Open has been initiated, but the lower layer is
      still unavailable (Down). The Restart timer is not running in the
      Starting state.

      When the lower layer becomes available (Up), a Configure-Request
      is sent.

   Closed

      In the Closed state, the link is available (Up), but no Open has
      occurred. The Restart timer is not running in the Closed state.

      Upon reception of Configure-Request packets, a Terminate-Ack is
      sent. Terminate-Acks are silently discarded to avoid creating a
      loop.

   Stopped

      The Stopped state is the Open counterpart to the Closed state. It
      is entered when the automaton is waiting for a Down event after
      the This-Layer-Finished action, or after sending a Terminate-Ack.
      The Restart timer is not running in the Stopped state.

      Upon reception of Configure-Request packets, an appropriate
      response is sent. Upon reception of other packets, a Terminate-
      Ack is sent. Terminate-Acks are silently discarded to avoid
      creating a loop.

      Rationale:

         The Stopped state is a junction state for link termination,
         link configuration failure, and other automaton failure modes.
         These potentially separate states have been combined.

         There is a race condition between the Down event response (from

the This-Layer-Finished action) and the Receive-Configure-
Request event. When a Configure-Request arrives before the
Down event, the Down event will supercede by returning the
automaton to the Starting state. This prevents attack by
repetition.

Implementation Option:

After the peer fails to respond to Configure-Requests, an
implementation MAY wait passively for the peer to send
Configure-Requests. In this case, the This-Layer-Finished
action is not used for the TO- event in states Req-Sent, Ack-
Rcvd and Ack-Sent.

This option is useful for dedicated circuits, or circuits which
have no status signals available, but SHOULD NOT be used for
switched circuits.

Closing

In the Closing state, an attempt is made to terminate the
connection. A Terminate-Request has been sent and the Restart
timer is running, but a Terminate-Ack has not yet been received.

Upon reception of a Terminate-Ack, the Closed state is entered.
Upon the expiration of the Restart timer, a new Terminate-Request
is transmitted, and the Restart timer is restarted. After the
Restart timer has expired Max-Terminate times, the Closed state is
entered.

Stopping

The Stopping state is the Open counterpart to the Closing state.
A Terminate-Request has been sent and the Restart timer is
running, but a Terminate-Ack has not yet been received.

Rationale:

The Stopping state provides a well defined opportunity to
terminate a link before allowing new traffic. After the link
has terminated, a new configuration may occur via the Stopped
or Starting states.

Request-Sent

In the Request-Sent state an attempt is made to configure the
connection. A Configure-Request has been sent and the Restart
timer is running, but a Configure-Ack has not yet been received

      nor has one been sent.

   Ack-Received

      In the Ack-Received state, a Configure-Request has been sent and a
      Configure-Ack has been received. The Restart timer is still
      running, since a Configure-Ack has not yet been sent.

   Ack-Sent

      In the Ack-Sent state, a Configure-Request and a Configure-Ack
      have both been sent, but a Configure-Ack has not yet been
      received. The Restart timer is running, since a Configure-Ack has
      not yet been received.

   Opened

      In the Opened state, a Configure-Ack has been both sent and
      received. The Restart timer is not running.

      When entering the Opened state, the implementation SHOULD signal
      the upper layers that it is now Up. Conversely, when leaving the
      Opened state, the implementation SHOULD signal the upper layers
      that it is now Down.

4.3. Events

   Transitions and actions in the automaton are caused by events.

   Up

      This event occurs when a lower layer indicates that it is ready to
      carry packets.

      Typically, this event is used by a modem handling or calling
      process, or by some other coupling of the PPP link to the physical
      media, to signal LCP that the link is entering Link Establishment
      phase.

      It also can be used by LCP to signal each NCP that the link is
      entering Network-Layer Protocol phase. That is, the This-Layer-Up
      action from LCP triggers the Up event in the NCP.

   Down

      This event occurs when a lower layer indicates that it is no

longer ready to carry packets.

Typically, this event is used by a modem handling or calling
process, or by some other coupling of the PPP link to the physical
media, to signal LCP that the link is entering Link Dead phase.

It also can be used by LCP to signal each NCP that the link is
leaving Network-Layer Protocol phase. That is, the This-Layer-
Down action from LCP triggers the Down event in the NCP.

   Open

      This event indicates that the link is administratively available
      for traffic; that is, the network administrator (human or program)
      has indicated that the link is allowed to be Opened. When this
      event occurs, and the link is not in the Opened state, the
      automaton attempts to send configuration packets to the peer.

      If the automaton is not able to begin configuration (the lower
      layer is Down, or a previous Close event has not completed), the
      establishment of the link is automatically delayed.

      When a Terminate-Request is received, or other events occur which
      cause the link to become unavailable, the automaton will progress
      to a state where the link is ready to re-open. No additional
      administrative intervention is necessary.

      Implementation Option:

         Experience has shown that users will execute an additional Open
         command when they want to renegotiate the link. This might
         indicate that new values are to be negotiated.

         Since this is not the meaning of the Open event, it is
         suggested that when an Open user command is executed in the
         Opened, Closing, Stopping, or Stopped states, the
         implementation issue a Down event, immediately followed by an
         Up event. Care must be taken that an intervening Down event
         cannot occur from another source.

         The Down followed by an Up will cause an orderly renegotiation
         of the link, by progressing through the Starting to the
         Request-Sent state. This will cause the renegotiation of the
         link, without any harmful side effects.

   Close

      This event indicates that the link is not available for traffic;

      that is, the network administrator (human or program) has
      indicated that the link is not allowed to be Opened. When this
      event occurs, and the link is not in the Closed state, the
      automaton attempts to terminate the connection. Futher attempts
      to re-configure the link are denied until a new Open event occurs.

      Implementation Note:

         When authentication fails, the link SHOULD be terminated, to
         prevent attack by repetition and denial of service to other
         users. Since the link is administratively available (by
         definition), this can be accomplished by simulating a Close
         event to the LCP, immediately followed by an Open event. Care
         must be taken that an intervening Close event cannot occur from
         another source.

         The Close followed by an Open will cause an orderly termination
         of the link, by progressing through the Closing to the Stopping
         state, and the This-Layer-Finished action can disconnect the
         link. The automaton waits in the Stopped or Starting states
         for the next connection attempt.

   Timeout (TO+,TO-)

      This event indicates the expiration of the Restart timer. The
      Restart timer is used to time responses to Configure-Request and
      Terminate-Request packets.

      The TO+ event indicates that the Restart counter continues to be
      greater than zero, which triggers the corresponding Configure-
      Request or Terminate-Request packet to be retransmitted.

      The TO- event indicates that the Restart counter is not greater
      than zero, and no more packets need to be retransmitted.

   Receive-Configure-Request (RCR+,RCR-)

      This event occurs when a Configure-Request packet is received from
      the peer. The Configure-Request packet indicates the desire to
      open a connection and may specify Configuration Options. The
      Configure-Request packet is more fully described in a later
      section.

      The RCR+ event indicates that the Configure-Request was
      acceptable, and triggers the transmission of a corresponding
      Configure-Ack.

      The RCR- event indicates that the Configure-Request was

unacceptable, and triggers the transmission of a corresponding
Configure-Nak or Configure-Reject.

Implementation Note:

   These events may occur on a connection which is already in the
   Opened state. The implementation MUST be prepared to
   immediately renegotiate the Configuration Options.

Receive-Configure-Ack (RCA)

   This event occurs when a valid Configure-Ack packet is received
   from the peer. The Configure-Ack packet is a positive response to
   a Configure-Request packet. An out of sequence or otherwise
   invalid packet is silently discarded.

   Implementation Note:

      Since the correct packet has already been received before
      reaching the Ack-Rcvd or Opened states, it is extremely
      unlikely that another such packet will arrive. As specified,
      all invalid Ack/Nak/Rej packets are silently discarded, and do
      not affect the transitions of the automaton.

      However, it is not impossible that a correctly formed packet
      will arrive through a coincidentally-timed cross-connection.
      It is more likely to be the result of an implementation error.
      At the very least, this occurance SHOULD be logged.

Receive-Configure-Nak/Rej (RCN)

   This event occurs when a valid Configure-Nak or Configure-Reject
   packet is received from the peer. The Configure-Nak and
   Configure-Reject packets are negative responses to a Configure-
   Request packet. An out of sequence or otherwise invalid packet is
   silently discarded.

   Implementation Note:

      Although the Configure-Nak and Configure-Reject cause the same
      state transition in the automaton, these packets have
      significantly different effects on the Configuration Options
      sent in the resulting Configure-Request packet.

Receive-Terminate-Request (RTR)

   This event occurs when a Terminate-Request packet is received.
   The Terminate-Request packet indicates the desire of the peer to

      close the connection.

      Implementation Note:

         This event is not identical to the Close event (see above), and
         does not override the Open commands of the local network
         administrator. The implementation MUST be prepared to receive
         a new Configure-Request without network administrator
         intervention.

   Receive-Terminate-Ack (RTA)

      This event occurs when a Terminate-Ack packet is received from the
      peer. The Terminate-Ack packet is usually a response to a
      Terminate-Request packet. The Terminate-Ack packet may also
      indicate that the peer is in Closed or Stopped states, and serves
      to re-synchronize the link configuration.

   Receive-Unknown-Code (RUC)

      This event occurs when an un-interpretable packet is received from
      the peer. A Code-Reject packet is sent in response.

   Receive-Code-Reject, Receive-Protocol-Reject (RXJ+,RXJ-)

      This event occurs when a Code-Reject or a Protocol-Reject packet
      is received from the peer.

      The RXJ+ event arises when the rejected value is acceptable, such
      as a Code-Reject of an extended code, or a Protocol-Reject of a
      NCP. These are within the scope of normal operation. The
      implementation MUST stop sending the offending packet type.

      The RXJ- event arises when the rejected value is catastrophic,
      such as a Code-Reject of Configure-Request, or a Protocol-Reject
      of LCP! This event communicates an unrecoverable error that
      terminates the connection.

   Receive-Echo-Request, Receive-Echo-Reply, Receive-Discard-Request
   (RXR)

      This event occurs when an Echo-Request, Echo-Reply or Discard-
      Request packet is received from the peer. The Echo-Reply packet
      is a response to an Echo-Request packet. There is no reply to an
      Echo-Reply or Discard-Request packet.

4.4. Actions

   Actions in the automaton are caused by events and typically indicate
   the transmission of packets and/or the starting or stopping of the
   Restart timer.

   Illegal-Event (-)

      This indicates an event that cannot occur in a properly
      implemented automaton. The implementation has an internal error,
      which should be reported and logged. No transition is taken, and
      the implementation SHOULD NOT reset or freeze.

   This-Layer-Up (tlu)

      This action indicates to the upper layers that the automaton is
      entering the Opened state.

      Typically, this action is used by the LCP to signal the Up event
      to a NCP, Authentication Protocol, or Link Quality Protocol, or
      MAY be used by a NCP to indicate that the link is available for
      its network layer traffic.

   This-Layer-Down (tld)

      This action indicates to the upper layers that the automaton is
      leaving the Opened state.

      Typically, this action is used by the LCP to signal the Down event
      to a NCP, Authentication Protocol, or Link Quality Protocol, or
      MAY be used by a NCP to indicate that the link is no longer
      available for its network layer traffic.

   This-Layer-Started (tls)

      This action indicates to the lower layers that the automaton is
      entering the Starting state, and the lower layer is needed for the
      link. The lower layer SHOULD respond with an Up event when the
      lower layer is available.

      This results of this action are highly implementation dependent.

   This-Layer-Finished (tlf)

      This action indicates to the lower layers that the automaton is
      entering the Initial, Closed or Stopped states, and the lower
      layer is no longer needed for the link. The lower layer SHOULD
      respond with a Down event when the lower layer has terminated.

Typically, this action MAY be used by the LCP to advance to the
Link Dead phase, or MAY be used by a NCP to indicate to the LCP
that the link may terminate when there are no other NCPs open.

This results of this action are highly implementation dependent.

Initialize-Restart-Count (irc)

This action sets the Restart counter to the appropriate value
(Max-Terminate or Max-Configure). The counter is decremented for
each transmission, including the first.

Implementation Note:

In addition to setting the Restart counter, the implementation
MUST set the timeout period to the initial value when Restart
timer backoff is used.

Zero-Restart-Count (zrc)

This action sets the Restart counter to zero.

Implementation Note:

This action enables the FSA to pause before proceeding to the
desired final state, allowing traffic to be processed by the
peer. In addition to zeroing the Restart counter, the
implementation MUST set the timeout period to an appropriate
value.

Send-Configure-Request (scr)

A Configure-Request packet is transmitted. This indicates the
desire to open a connection with a specified set of Configuration
Options. The Restart timer is started when the Configure-Request
packet is transmitted, to guard against packet loss. The Restart
counter is decremented each time a Configure-Request is sent.

Send-Configure-Ack (sca)

A Configure-Ack packet is transmitted. This acknowledges the
reception of a Configure-Request packet with an acceptable set of
Configuration Options.

Send-Configure-Nak (scn)

A Configure-Nak or Configure-Reject packet is transmitted, as
appropriate. This negative response reports the reception of a

Configure-Request packet with an unacceptable set of Configuration
Options.

Configure-Nak packets are used to refuse a Configuration Option
value, and to suggest a new, acceptable value. Configure-Reject
packets are used to refuse all negotiation about a Configuration
Option, typically because it is not recognized or implemented.
The use of Configure-Nak versus Configure-Reject is more fully
described in the chapter on LCP Packet Formats.

Send-Terminate-Request (str)

A Terminate-Request packet is transmitted. This indicates the
desire to close a connection. The Restart timer is started when
the Terminate-Request packet is transmitted, to guard against
packet loss. The Restart counter is decremented each time a
Terminate-Request is sent.

Send-Terminate-Ack (sta)

A Terminate-Ack packet is transmitted. This acknowledges the
reception of a Terminate-Request packet or otherwise serves to
synchronize the automatons.

Send-Code-Reject (scj)

A Code-Reject packet is transmitted. This indicates the reception
of an unknown type of packet.

Send-Echo-Reply (ser)

An Echo-Reply packet is transmitted. This acknowledges the
reception of an Echo-Request packet.

4.5. Loop Avoidance

The protocol makes a reasonable attempt at avoiding Configuration
Option negotiation loops. However, the protocol does NOT guarantee
that loops will not happen. As with any negotiation, it is possible
to configure two PPP implementations with conflicting policies that
will never converge. It is also possible to configure policies which
do converge, but which take significant time to do so. Implementors
should keep this in mind and SHOULD implement loop detection
mechanisms or higher level timeouts.

4.6. Counters and Timers

   Restart Timer

      There is one special timer used by the automaton. The Restart
      timer is used to time transmissions of Configure-Request and
      Terminate-Request packets. Expiration of the Restart timer causes
      a Timeout event, and retransmission of the corresponding
      Configure-Request or Terminate-Request packet. The Restart timer
      MUST be configurable, but SHOULD default to three (3) seconds.

      Implementation Note:

         The Restart timer SHOULD be based on the speed of the link.
         The default value is designed for low speed (2,400 to 9,600
         bps), high switching latency links (typical telephone lines).
         Higher speed links, or links with low switching latency, SHOULD
         have correspondingly faster retransmission times.

         Instead of a constant value, the Restart timer MAY begin at an
         initial small value and increase to the configured final value.
         Each successive value less than the final value SHOULD be at
         least twice the previous value. The initial value SHOULD be
         large enough to account for the size of the packets, twice the
         round trip time for transmission at the link speed, and at
         least an additional 100 milliseconds to allow the peer to
         process the packets before responding. Some circuits add
         another 200 milliseconds of satellite delay. Round trip times
         for modems operating at 14,400 bps have been measured in the
         range of 160 to more than 600 milliseconds.

   Max-Terminate

      There is one required restart counter for Terminate-Requests.
      Max-Terminate indicates the number of Terminate-Request packets
      sent without receiving a Terminate-Ack before assuming that the
      peer is unable to respond. Max-Terminate MUST be configurable,
      but SHOULD default to two (2) transmissions.

   Max-Configure

      A similar counter is recommended for Configure-Requests. Max-
      Configure indicates the number of Configure-Request packets sent
      without receiving a valid Configure-Ack, Configure-Nak or
      Configure-Reject before assuming that the peer is unable to
      respond. Max-Configure MUST be configurable, but SHOULD default
      to ten (10) transmissions.

   Max-Failure

      A related counter is recommended for Configure-Nak. Max-Failure
      indicates the number of Configure-Nak packets sent without sending
      a Configure-Ack before assuming that configuration is not
      converging. Any further Configure-Nak packets for peer requested
      options are converted to Configure-Reject packets, and locally
      desired options are no longer appended. Max-Failure MUST be
      configurable, but SHOULD default to five (5) transmissions.

5. LCP Packet Formats

    There are three classes of LCP packets:

        1. Link Configuration packets used to establish and configure a
           link (Configure-Request, Configure-Ack, Configure-Nak and
           Configure-Reject).

        2. Link Termination packets used to terminate a link (Terminate-
           Request and Terminate-Ack).

        3. Link Maintenance packets used to manage and debug a link
           (Code-Reject, Protocol-Reject, Echo-Request, Echo-Reply, and
           Discard-Request).

    In the interest of simplicity, there is no version field in the LCP
    packet. A correctly functioning LCP implementation will always
    respond to unknown Protocols and Codes with an easily recognizable
    LCP packet, thus providing a deterministic fallback mechanism for
    implementations of other versions.

    Regardless of which Configuration Options are enabled, all LCP Link
    Configuration, Link Termination, and Code-Reject packets (codes 1
    through 7) are always sent as if no Configuration Options were
    negotiated. In particular, each Configuration Option specifies a
    default value. This ensures that such LCP packets are always
    recognizable, even when one end of the link mistakenly believes the
    link to be open.

    Exactly one LCP packet is encapsulated in the PPP Information field,
    where the PPP Protocol field indicates type hex c021 (Link Control
    Protocol).

    A summary of the Link Control Protocol 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...
+-+-+-+-+
```

Code

    The Code field is one octet, and identifies the kind of LCP

packet. When a packet is received with an unknown Code field, a
Code-Reject packet is transmitted.

Up-to-date values of the LCP Code field are specified in the most
recent "Assigned Numbers" RFC [2]. This document concerns the
following values:

       1   Configure-Request
       2   Configure-Ack
       3   Configure-Nak
       4   Configure-Reject
       5   Terminate-Request
       6   Terminate-Ack
       7   Code-Reject
       8   Protocol-Reject
       9   Echo-Request
       10   Echo-Reply
       11   Discard-Request

Identifier

   The Identifier field is one octet, and aids in matching requests
   and replies. When a packet is received with an invalid Identifier
   field, the packet is silently discarded without affecting the
   automaton.

Length

   The Length field is two octets, and indicates the length of the
   LCP packet, including the Code, Identifier, Length and Data
   fields. The Length MUST NOT exceed the MRU of the link.

   Octets outside the range of the Length field are treated as
   padding and are ignored on reception. When a packet is received
   with an invalid Length field, the packet is silently discarded
   without affecting the automaton.

Data

   The Data field is zero or more octets, as indicated by the Length
   field. The format of the Data field is determined by the Code
   field.

5.1. Configure-Request

   Description

      An implementation wishing to open a connection MUST transmit a
      Configure-Request. The Options field is filled with any desired
      changes to the link defaults. Configuration Options SHOULD NOT be
      included with default values.

      Upon reception of a Configure-Request, an appropriate reply MUST
      be transmitted.

   A summary of the Configure-Request 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Options ...
+-+-+-+-+
```

   Code

      1 for Configure-Request.

   Identifier

      The Identifier field MUST be changed whenever the contents of the
      Options field changes, and whenever a valid reply has been
      received for a previous request. For retransmissions, the
      Identifier MAY remain unchanged.

   Options

      The options field is variable in length, and contains the list of
      zero or more Configuration Options that the sender desires to
      negotiate. All Configuration Options are always negotiated
      simultaneously. The format of Configuration Options is further
      described in a later chapter.

5.2. Configure-Ack

   Description

      If every Configuration Option received in a Configure-Request is
      recognizable and all values are acceptable, then the
      implementation MUST transmit a Configure-Ack. The acknowledged
      Configuration Options MUST NOT be reordered or modified in any
      way.

      On reception of a Configure-Ack, the Identifier field MUST match
      that of the last transmitted Configure-Request. Additionally, the
      Configuration Options in a Configure-Ack MUST exactly match those
      of the last transmitted Configure-Request. Invalid packets are
      silently discarded.

   A summary of the Configure-Ack 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Options ...
+-+-+-+-+
```

   Code

      2 for Configure-Ack.

   Identifier

      The Identifier field is a copy of the Identifier field of the
      Configure-Request which caused this Configure-Ack.

   Options

      The Options field is variable in length, and contains the list of
      zero or more Configuration Options that the sender is
      acknowledging. All Configuration Options are always acknowledged
      simultaneously.

5.3. Configure-Nak

   Description

      If every instance of the received Configuration Options is
      recognizable, but some values are not acceptable, then the
      implementation MUST transmit a Configure-Nak. The Options field
      is filled with only the unacceptable Configuration Options from
      the Configure-Request. All acceptable Configuration Options are
      filtered out of the Configure-Nak, but otherwise the Configuration
      Options from the Configure-Request MUST NOT be reordered.

      Options which have no value fields (boolean options) MUST use the
      Configure-Reject reply instead.

      Each Configuration Option which is allowed only a single instance
      MUST be modified to a value acceptable to the Configure-Nak
      sender. The default value MAY be used, when this differs from the
      requested value.

      When a particular type of Configuration Option can be listed more
      than once with different values, the Configure-Nak MUST include a
      list of all values for that option which are acceptable to the
      Configure-Nak sender. This includes acceptable values that were
      present in the Configure-Request.

      Finally, an implementation may be configured to request the
      negotiation of a specific Configuration Option. If that option is
      not listed, then that option MAY be appended to the list of Nak'd
      Configuration Options, in order to prompt the peer to include that
      option in its next Configure-Request packet. Any value fields for
      the option MUST indicate values acceptable to the Configure-Nak
      sender.

      On reception of a Configure-Nak, the Identifier field MUST match
      that of the last transmitted Configure-Request. Invalid packets
      are silently discarded.

      Reception of a valid Configure-Nak indicates that when a new
      Configure-Request is sent, the Configuration Options MAY be
      modified as specified in the Configure-Nak. When multiple
      instances of a Configuration Option are present, the peer SHOULD
      select a single value to include in its next Configure-Request
      packet.

      Some Configuration Options have a variable length. Since the
      Nak'd Option has been modified by the peer, the implementation
      MUST be able to handle an Option length which is different from

      the original Configure-Request.

   A summary of the Configure-Nak 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Options ...
+-+-+-+-+
```


   Code

      3 for Configure-Nak.

   Identifier

      The Identifier field is a copy of the Identifier field of the
      Configure-Request which caused this Configure-Nak.

   Options

      The Options field is variable in length, and contains the list of
      zero or more Configuration Options that the sender is Nak'ing.
      All Configuration Options are always Nak'd simultaneously.

5.4. Configure-Reject

   Description

      If some Configuration Options received in a Configure-Request are
      not recognizable or are not acceptable for negotiation (as
      configured by a network administrator), then the implementation
      MUST transmit a Configure-Reject. The Options field is filled
      with only the unacceptable Configuration Options from the
      Configure-Request. All recognizable and negotiable Configuration
      Options are filtered out of the Configure-Reject, but otherwise
      the Configuration Options MUST NOT be reordered or modified in any
      way.

      On reception of a Configure-Reject, the Identifier field MUST
      match that of the last transmitted Configure-Request.
      Additionally, the Configuration Options in a Configure-Reject MUST

be a proper subset of those in the last transmitted Configure-
Request. Invalid packets are silently discarded.

Reception of a valid Configure-Reject indicates that when a new
Configure-Request is sent, it MUST NOT include any of the
Configuration Options listed in the Configure-Reject.

A summary of the Configure-Reject 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Options ...
+-+-+-+-+
```

Code

   4 for Configure-Reject.

Identifier

   The Identifier field is a copy of the Identifier field of the
   Configure-Request which caused this Configure-Reject.

Options

   The Options field is variable in length, and contains the list of
   zero or more Configuration Options that the sender is rejecting.
   All Configuration Options are always rejected simultaneously.

5.5.   Terminate-Request and Terminate-Ack

   Description

      LCP includes Terminate-Request and Terminate-Ack Codes in order to
      provide a mechanism for closing a connection.

      An implementation wishing to close a connection SHOULD transmit a
      Terminate-Request. Terminate-Request packets SHOULD continue to
      be sent until Terminate-Ack is received, the lower layer indicates
      that it has gone down, or a sufficiently large number have been
      transmitted such that the peer is down with reasonable certainty.

      Upon reception of a Terminate-Request, a Terminate-Ack MUST be
      transmitted.

      Reception of an unelicited Terminate-Ack indicates that the peer
      is in the Closed or Stopped states, or is otherwise in need of
      re-negotiation.

   A summary of the Terminate-Request and Terminate-Ack packet formats
   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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...
+-+-+-+-+
```

   Code

      5 for Terminate-Request;

      6 for Terminate-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 Terminate-Request is
      copied into the Identifier field of the Terminate-Ack packet.

   Data

      The Data field is zero or more octets, and contains uninterpreted
      data for use by the sender. The data may consist of any binary
      value. The end of the field is indicated by the Length.

5.6. Code-Reject

   Description

      Reception of a LCP packet with an unknown Code indicates that the
      peer is operating with a different version. This MUST be reported
      back to the sender of the unknown Code by transmitting a Code-
      Reject.

      Upon reception of the Code-Reject of a code which is fundamental
      to this version of the protocol, the implementation SHOULD report
      the problem and drop the connection, since it is unlikely that the
      situation can be rectified automatically.

   A summary of the Code-Reject 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rejected-Packet ...
+-+-+-+-+-+-+-+-+
```


   Code

      7 for Code-Reject.

   Identifier

      The Identifier field MUST be changed for each Code-Reject sent.

   Rejected-Packet

      The Rejected-Packet field contains a copy of the LCP packet which
      is being rejected. It begins with the Information field, and does
      not include any Data Link Layer headers nor an FCS. The
      Rejected-Packet MUST be truncated to comply with the peer's

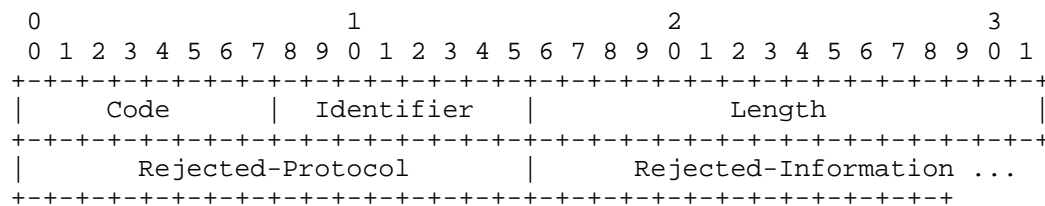established MRU.

5.7. Protocol-Reject

   Description

      Reception of a PPP packet with an unknown Protocol field indicates
      that the peer is attempting to use a protocol which is
      unsupported. This usually occurs when the peer attempts to
      configure a new protocol. If the LCP automaton is in the Opened
      state, then this MUST be reported back to the peer by transmitting
      a Protocol-Reject.

      Upon reception of a Protocol-Reject, the implementation MUST stop
      sending packets of the indicated protocol at the earliest
      opportunity.

      Protocol-Reject packets can only be sent in the LCP Opened state.
      Protocol-Reject packets received in any state other than the LCP
      Opened state SHOULD be silently discarded.

   A summary of the Protocol-Reject 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Rejected-Protocol       |    Rejected-Information ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


   Code

      8 for Protocol-Reject.

   Identifier

      The Identifier field MUST be changed for each Protocol-Reject
      sent.

   Rejected-Protocol

      The Rejected-Protocol field is two octets, and contains the PPP
      Protocol field of the packet which is being rejected.

   Rejected-Information

      The Rejected-Information field contains a copy of the packet which
      is being rejected. It begins with the Information field, and does
      not include any Data Link Layer headers nor an FCS. The
      Rejected-Information MUST be truncated to comply with the peer's
      established MRU.

5.8. Echo-Request and Echo-Reply

   Description

      LCP includes Echo-Request and Echo-Reply Codes in order to provide
      a Data Link Layer loopback mechanism for use in exercising both
      directions of the link. This is useful as an aid in debugging,
      link quality determination, performance testing, and for numerous
      other functions.

      Upon reception of an Echo-Request in the LCP Opened state, an
      Echo-Reply MUST be transmitted.

      Echo-Request and Echo-Reply packets MUST only be sent in the LCP
      Opened state. Echo-Request and Echo-Reply packets received in any
      state other than the LCP Opened state SHOULD be silently
      discarded.

   A summary of the Echo-Request and Echo-Reply packet formats 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Magic-Number                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...
+-+-+-+-+
```

   Code

      9 for Echo-Request;

      10 for Echo-Reply.

   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 Echo-Request is copied
      into the Identifier field of the Echo-Reply packet.

   Magic-Number

      The Magic-Number field is four octets, and aids in detecting links
      which are in the looped-back condition. Until the Magic-Number
      Configuration Option has been successfully negotiated, the Magic-
      Number MUST be transmitted as zero. See the Magic-Number
      Configuration Option for further explanation.

   Data

      The Data field is zero or more octets, and contains uninterpreted
      data for use by the sender. The data may consist of any binary
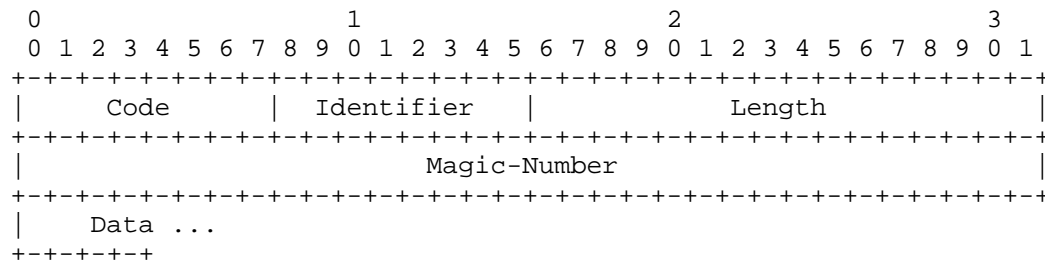   value. The end of the field is indicated by the Length.

5.9. Discard-Request

   Description

      LCP includes a Discard-Request Code in order to provide a Data
      Link Layer sink mechanism for use in exercising the local to
      remote direction of the link. This is useful as an aid in
      debugging, performance testing, and for numerous other functions.

      Discard-Request packets MUST only be sent in the LCP Opened state.
      On reception, the receiver MUST silently discard any Discard-
      Request that it receives.

A summary of the Discard-Request 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Magic-Number                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...
+-+-+-+-+
```

Code

   11 for Discard-Request.

Identifier

   The Identifier field MUST be changed for each Discard-Request
   sent.

Magic-Number

   The Magic-Number field is four octets, and aids in detecting links
   which are in the looped-back condition. Until the Magic-Number
   Configuration Option has been successfully negotiated, the Magic-
   Number MUST be transmitted as zero. See the Magic-Number
   Configuration Option for further explanation.

Data

   The Data field is zero or more octets, and contains uninterpreted
   data for use by the sender. The data may consist of any binary
   value. The end of the field is indicated by the Length.

6. LCP Configuration Options

   LCP Configuration Options allow negotiation of modifications to the
   default characteristics of a point-to-point link. If a Configuration
   Option is not included in a Configure-Request packet, the default
   value for that Configuration Option is assumed.

   Some Configuration Options MAY be listed more than once. The effect
   of this is Configuration Option specific, and is specified by each
   such Configuration Option description. (None of the Configuration
   Options in this specification can be listed more than once.)

   The end of the list of Configuration Options is indicated by the
   Length field of the LCP packet.

   Unless otherwise specified, all Configuration Options apply in a
   half-duplex fashion; typically, in the receive direction of the link
   from the point of view of the Configure-Request sender.

   Design Philosophy

      The options indicate additional capabilities or requirements of
      the implementation that is requesting the option. An
      implementation which does not understand any option SHOULD
      interoperate with one which implements every option.

      A default is specified for each option which allows the link to
      correctly function without negotiation of the option, although
      perhaps with less than optimal performance.

      Except where explicitly specified, acknowledgement of an option
      does not require the peer to take any additional action other than
      the default.

      It is not necessary to send the default values for the options in
      a Configure-Request.

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

    0   1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
   +-+
   | Type | Length | Data ...
   +-+

   Type

      The Type field is one octet, and indicates the type of
      Configuration Option. Up-to-date values of the LCP Option Type
      field are specified in the most recent "Assigned Numbers" RFC [2].
      This document concerns the following values:

         0   RESERVED
         1   Maximum-Receive-Unit
         3   Authentication-Protocol
         4   Quality-Protocol
         5   Magic-Number
         7   Protocol-Field-Compression
         8   Address-and-Control-Field-Compression

   Length

      The Length field is one octet, and indicates the length of this
      Configuration Option including the Type, Length and Data fields.

      If a negotiable Configuration Option is received in a Configure-
      Request, but with an invalid or unrecognized Length, a Configure-
      Nak SHOULD be transmitted which includes the desired Configuration
      Option with an appropriate Length and Data.

   Data

      The Data field is zero or more octets, and contains information
      specific to the Configuration Option. The format and length of
      the Data field is determined by the Type and Length fields.

      When the Data field is indicated by the Length to extend beyond
      the end of the Information field, the entire packet is silently
      discarded without affecting the automaton.

6.1. Maximum-Receive-Unit (MRU)

   Description

      This Configuration Option may be sent to inform the peer that the
      implementation can receive larger packets, or to request that the
      peer send smaller packets.

      The default value is 1500 octets. If smaller packets are
      requested, an implementation MUST still be able to receive the
      full 1500 octet information field in case link synchronization is
      lost.

      Implementation Note:

         This option is used to indicate an implementation capability.
         The peer is not required to maximize the use of the capacity.
         For example, when a MRU is indicated which is 2048 octets, the
         peer is not required to send any packet with 2048 octets. The
         peer need not Configure-Nak to indicate that it will only send
         smaller packets, since the implementation will always require
         support for at least 1500 octets.

   A summary of the Maximum-Receive-Unit 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     |      Maximum-Receive-Unit     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Type

      1

   Length

      4

   Maximum-Receive-Unit

      The Maximum-Receive-Unit field is two octets, and specifies the
      maximum number of octets in the Information and Padding fields.
      It does not include the framing, Protocol field, FCS, nor any
      transparency bits or bytes.

6.2. Authentication-Protocol

   Description

      On some links it may be desirable to require a peer to
      authenticate itself before allowing network-layer protocol packets
      to be exchanged.

      This Configuration Option provides a method to negotiate the use
      of a specific protocol for authentication. By default,
      authentication is not required.

      An implementation MUST NOT include multiple Authentication-
      Protocol Configuration Options in its Configure-Request packets.
      Instead, it SHOULD attempt to configure the most desirable
      protocol first. If that protocol is Configure-Nak'd, then the
      implementation SHOULD attempt the next most desirable protocol in
      the next Configure-Request.

      The implementation sending the Configure-Request is indicating
      that it expects authentication from its peer. If an
      implementation sends a Configure-Ack, then it is agreeing to
      authenticate with the specified protocol. An implementation
      receiving a Configure-Ack SHOULD expect the peer to authenticate
      with the acknowledged protocol.

      There is no requirement that authentication be full-duplex or that
      the same protocol be used in both directions. It is perfectly
      acceptable for different protocols to be used in each direction.
      This will, of course, depend on the specific protocols negotiated.

   A summary of the Authentication-Protocol 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     |     Authentication-Protocol   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...
+-+-+-+-+
```

   Type

      3

   Length

      >= 4

   Authentication-Protocol

      The Authentication-Protocol field is two octets, and indicates the
      authentication protocol desired. Values for this field are always
      the same as the PPP Protocol field values for that same
      authentication protocol.

      Up-to-date values of the Authentication-Protocol field are
      specified in the most recent "Assigned Numbers" RFC [2]. Current
      values are assigned as follows:

      Value (in hex) Protocol

      c023   Password Authentication Protocol
      c223   Challenge Handshake Authentication Protocol

    Data

      The Data field is zero or more octets, and contains additional
      data as determined by the particular protocol.

6.3. Quality-Protocol

   Description

      On some links it may be desirable to determine when, and how
      often, the link is dropping data. This process is called link
      quality monitoring.

      This Configuration Option provides a method to negotiate the use
      of a specific protocol for link quality monitoring. By default,
      link quality monitoring is disabled.

      The implementation sending the Configure-Request is indicating
      that it expects to receive monitoring information from its peer.
      If an implementation sends a Configure-Ack, then it is agreeing to
      send the specified protocol. An implementation receiving a
      Configure-Ack SHOULD expect the peer to send the acknowledged
      protocol.

      There is no requirement that quality monitoring be full-duplex or

      that the same protocol be used in both directions. It is
      perfectly acceptable for different protocols to be used in each
      direction. This will, of course, depend on the specific protocols
      negotiated.

   A summary of the Quality-Protocol 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     |        Quality-Protocol       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    Data ...
   +-+-+-+-+
```

   Type

      4

   Length

      >= 4

   Quality-Protocol

      The Quality-Protocol field is two octets, and indicates the link
      quality monitoring protocol desired. Values for this field are
      always the same as the PPP Protocol field values for that same
      monitoring protocol.

      Up-to-date values of the Quality-Protocol field are specified in
      the most recent "Assigned Numbers" RFC [2]. Current values are
      assigned as follows:

      Value (in hex) Protocol

      c025   Link Quality Report

   Data

      The Data field is zero or more octets, and contains additional
      data as determined by the particular protocol.

6.4. Magic-Number

   Description

      This Configuration Option provides a method to detect looped-back
      links and other Data Link Layer anomalies. This Configuration
      Option MAY be required by some other Configuration Options such as
      the Quality-Protocol Configuration Option. By default, the
      Magic-Number is not negotiated, and zero is inserted where a
      Magic-Number might otherwise be used.

      Before this Configuration Option is requested, an implementation
      MUST choose its Magic-Number. It is recommended that the Magic-
      Number be chosen in the most random manner possible in order to
      guarantee with very high probability that an implementation will
      arrive at a unique number. A good way to choose a unique random
      number is to start with a unique seed. Suggested sources of
      uniqueness include machine serial numbers, other network hardware
      addresses, time-of-day clocks, etc. Particularly good random
      number seeds are precise measurements of the inter-arrival time of
      physical events such as packet reception on other connected
      networks, server response time, or the typing rate of a human
      user. It is also suggested that as many sources as possible be
      used simultaneously.

      When a Configure-Request is received with a Magic-Number
      Configuration Option, the received Magic-Number is compared with
      the Magic-Number of the last Configure-Request sent to the peer.
      If the two Magic-Numbers are different, then the link is not
      looped-back, and the Magic-Number SHOULD be acknowledged. If the
      two Magic-Numbers are equal, then it is possible, but not certain,
      that the link is looped-back and that this Configure-Request is
      actually the one last sent. To determine this, a Configure-Nak
      MUST be sent specifying a different Magic-Number value. A new
      Configure-Request SHOULD NOT be sent to the peer until normal
      processing would cause it to be sent (that is, until a Configure-
      Nak is received or the Restart timer runs out).

      Reception of a Configure-Nak with a Magic-Number different from
      that of the last Configure-Nak sent to the peer proves that a link
      is not looped-back, and indicates a unique Magic-Number. If the
      Magic-Number is equal to the one sent in the last Configure-Nak,
      the possibility of a looped-back link is increased, and a new
      Magic-Number MUST be chosen. In either case, a new Configure-
      Request SHOULD be sent with the new Magic-Number.

      If the link is indeed looped-back, this sequence (transmit
      Configure-Request, receive Configure-Request, transmit Configure-

Nak, receive Configure-Nak) will repeat over and over again. If
the link is not looped-back, this sequence might occur a few
times, but it is extremely unlikely to occur repeatedly. More
likely, the Magic-Numbers chosen at either end will quickly
diverge, terminating the sequence. The following table shows the
probability of collisions assuming that both ends of the link
select Magic-Numbers with a perfectly uniform distribution:

Number of Collisions	Probability
1	$1/2^{32}$ = 2.3 E-10
2	$1/2^{32^2}$ = 5.4 E-20
3	$1/2^{32^3}$ = 1.3 E-29

Good sources of uniqueness or randomness are required for this
divergence to occur. If a good source of uniqueness cannot be
found, it is recommended that this Configuration Option not be
enabled; Configure-Requests with the option SHOULD NOT be
transmitted and any Magic-Number Configuration Options which the
peer sends SHOULD be either acknowledged or rejected. In this
case, looped-back links cannot be reliably detected by the
implementation, although they may still be detectable by the peer.

If an implementation does transmit a Configure-Request with a
Magic-Number Configuration Option, then it MUST NOT respond with a
Configure-Reject when it receives a Configure-Request with a
Magic-Number Configuration Option. That is, if an implementation
desires to use Magic Numbers, then it MUST also allow its peer to
do so. If an implementation does receive a Configure-Reject in
response to a Configure-Request, it can only mean that the link is
not looped-back, and that its peer will not be using Magic-
Numbers. In this case, an implementation SHOULD act as if the
negotiation had been successful (as if it had instead received a
Configure-Ack).

The Magic-Number also may be used to detect looped-back links
during normal operation, as well as during Configuration Option
negotiation. All LCP Echo-Request, Echo-Reply, and Discard-
Request packets have a Magic-Number field. If Magic-Number has
been successfully negotiated, an implementation MUST transmit
these packets with the Magic-Number field set to its negotiated
Magic-Number.

The Magic-Number field of these packets SHOULD be inspected on
reception. All received Magic-Number fields MUST be equal to
either zero or the peer's unique Magic-Number, depending on
whether or not the peer negotiated a Magic-Number.

Reception of a Magic-Number field equal to the negotiated local
Magic-Number indicates a looped-back link. Reception of a Magic-
Number other than the negotiated local Magic-Number, the peer's
negotiated Magic-Number, or zero if the peer didn't negotiate one,
indicates a link which has been (mis)configured for communications
with a different peer.

Procedures for recovery from either case are unspecified, and may
vary from implementation to implementation. A somewhat
pessimistic procedure is to assume a LCP Down event. A further
Open event will begin the process of re-establishing the link,
which can't complete until the looped-back condition is
terminated, and Magic-Numbers are successfully negotiated. A more
optimistic procedure (in the case of a looped-back link) is to
begin transmitting LCP Echo-Request packets until an appropriate
Echo-Reply is received, indicating a termination of the looped-
back condition.

A summary of the Magic-Number 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     |          Magic-Number
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     Magic-Number (cont)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

    5

Length

    6

Magic-Number

    The Magic-Number field is four octets, and indicates a number
    which is very likely to be unique to one end of the link. A
    Magic-Number of zero is illegal and MUST always be Nak'd, if it is
    not Rejected outright.

6.5. Protocol-Field-Compression (PFC)

   Description

      This Configuration Option provides a method to negotiate the
      compression of the PPP Protocol field. By default, all
      implementations MUST transmit packets with two octet PPP Protocol
      fields.

      PPP Protocol field numbers are chosen such that some values may be
      compressed into a single octet form which is clearly
      distinguishable from the two octet form. This Configuration
      Option is sent to inform the peer that the implementation can
      receive such single octet Protocol fields.

      As previously mentioned, the Protocol field uses an extension
      mechanism consistent with the ISO 3309 extension mechanism for the
      Address field; the Least Significant Bit (LSB) of each octet is
      used to indicate extension of the Protocol field. A binary "0" as
      the LSB indicates that the Protocol field continues with the
      following octet. The presence of a binary "1" as the LSB marks
      the last octet of the Protocol field. Notice that any number of
      "0" octets may be prepended to the field, and will still indicate
      the same value (consider the two binary representations for 3,
      00000011 and 00000000 00000011).

      When using low speed links, it is desirable to conserve bandwidth
      by sending as little redundant data as possible. The Protocol-
      Field-Compression Configuration Option allows a trade-off between
      implementation simplicity and bandwidth efficiency. If
      successfully negotiated, the ISO 3309 extension mechanism may be
      used to compress the Protocol field to one octet instead of two.
      The large majority of packets are compressible since data
      protocols are typically assigned with Protocol field values less
      than 256.

      Compressed Protocol fields MUST NOT be transmitted unless this
      Configuration Option has been negotiated. When negotiated, PPP
      implementations MUST accept PPP packets with either double-octet
      or single-octet Protocol fields, and MUST NOT distinguish between
      them.

      The Protocol field is never compressed when sending any LCP
      packet. This rule guarantees unambiguous recognition of LCP
      packets.

      When a Protocol field is compressed, the Data Link Layer FCS field
      is calculated on the compressed frame, not the original

        uncompressed frame.

   A summary of the Protocol-Field-Compression Configuration Option
   format is shown below. The fields are transmitted from left to
   right.

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

   Type

      7

   Length

      2

6.6. Address-and-Control-Field-Compression (ACFC)

   Description

      This Configuration Option provides a method to negotiate the
      compression of the Data Link Layer Address and Control fields. By
      default, all implementations MUST transmit frames with Address and
      Control fields appropriate to the link framing.

      Since these fields usually have constant values for point-to-point
      links, they are easily compressed. This Configuration Option is
      sent to inform the peer that the implementation can receive
      compressed Address and Control fields.

      If a compressed frame is received when Address-and-Control-Field-
      Compression has not been negotiated, the implementation MAY
      silently discard the frame.

      The Address and Control fields MUST NOT be compressed when sending
      any LCP packet. This rule guarantees unambiguous recognition of
      LCP packets.

      When the Address and Control fields are compressed, the Data Link
      Layer FCS field is calculated on the compressed frame, not the
      original uncompressed frame.

   A summary of the Address-and-Control-Field-Compression configuration
   option format is shown below. The fields are transmitted from left
   to right.

    0   1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Type | Length |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   Type

      8

   Length

      2

Security Considerations

   Security issues are briefly discussed in sections concerning the
   Authentication Phase, the Close event, and the Authentication-
   Protocol Configuration Option.

References

   [1]   Perkins, D., "Requirements for an Internet Standard Point-to-
          Point Protocol", RFC 1547, Carnegie Mellon University,
          December 1993.

   [2]   Reynolds, J., and Postel, J., "Assigned Numbers", STD 2, RFC
          1340, USC/Information Sciences Institute, July 1992.

Acknowledgements

Chair's Address

    The working group can be contacted via the current chair:

        Fred Baker
        Advanced Computer Communications
        315 Bollay Drive
        Santa Barbara, California 93117

        fbaker@acc.com

Editor's Address

    Questions about this memo can also be directed to:

        William Allen Simpson
        Daydreamer
        Computer Systems Consulting Services
        1384 Fontaine
        Madison Heights, Michigan 48071

        Bill.Simpson@um.cc.umich.edu
            bsimpson@MorningStar.com