                    PPP Link Quality Monitoring

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) [1] provides a standard method for
   transporting multi-protocol datagrams over point-to-point links.

   PPP also defines an extensible Link Control Protocol, which allows
   negotiation of a Quality Protocol for continuous monitoring of the
   viability of the link.

   This document defines a protocol for generating Link-Quality-Reports.

Table of Contents

1. Introduction

   In order to establish communications over a point-to-point link, each
   end of the PPP link must first send LCP packets to configure the data
   link during Link Establishment phase. During the Authentication and
   Network-Layer Protocol phases, the link may be tested to determine if
   quality is sufficient for operation. This testing is completely
   optional.

   If an implementation desires that the peer use some specific link
   quality monitoring protocol, then it MUST negotiate the use of that
   protocol using the Quality-Protocol Configuration Option during Link
   Establishment phase.

   The negotiation mechanism is independent in each direction. However,
   if the peer agrees to send Quality-Protocol packets, it MUST
   correctly process such packets on reception, even if it does not
   request such packets or implement a monitoring policy.

2. Link Quality Monitoring

   Data communications links are rarely perfect. Packets can be dropped
   or corrupted for various reasons (line noise, equipment failure,
   buffer overruns, etc.). Sometimes, it is desirable to determine
   when, and how often, the link is dropping data. For example, routers
   may want to temporarily allow another route to take precedence. An
   implementation may also have the option of disconnecting and
   switching to an alternate link. The process of determining data loss
   is called "Link Quality Monitoring".

2.1. Design Motivation

   There are many different ways to measure link quality, and even more
   ways to react to it. Rather than specifying a single scheme, Link
   Quality Monitoring is divided into a "mechanism" and a "policy". PPP
   fully specifies the "mechanism" for Link Quality Monitoring by
   defining the Link-Quality-Report (LQR) packet and specifying a
   procedure for its use.

   PPP does NOT specify a Link Quality Monitoring "policy" -- how to
   judge link quality or what to do when it is inadequate. That is left
   as an implementation decision, and can be different at each end of
   the link. Implementations are allowed, and even encouraged, to
   experiment with various link quality policies. The Link Quality
   Monitoring mechanism specification ensures that two implementations
   with different policies may communicate and interoperate.

To allow flexible policies to be implemented, the PPP Link Quality
Monitoring mechanism measures data loss in units of packets, octets,
and Link-Quality-Reports. Each measurement is made separately for
each half of the link, both inbound and outbound. All measurements
are communicated to both ends of the link, so that each end of the
link can implement its own link quality policy for both its outbound
and inbound links.

Finally, the Link Quality Monitoring protocol is designed to be
implementable on many different kinds of systems. Although it may be
common to implement PPP (and especially Link Quality Monitoring) as a
single software process, multi-process implementations with hardware
support are also envisioned. The PPP Link Quality Monitoring
mechanism provides for careful definition of the Link-Quality-Report
packet format, and specifies reference points for all data
transmission and reception measurements.

2.2. Counters

Each Link Quality Monitoring implementation maintains counts of the
number of packets and octets transmitted and successfully received,
and periodically transmits this information to its peer in a Link-
Quality-Report packet.

These counters are similar to sequence numbers; they are constantly
increasing to give a "relative" indication of the number of packets
and octets communicated across the outbound link. By comparing the
values in successive Link-Quality-Reports, an LQR receiver can
compute the "delta" number of packets and octets successfully
communicated across the link. Comparing these absolute numbers then
gives an indication of a link's quality. Relative numbers, rather
than absolute, are transmitted because they greatly simplify link
synchronization.

The Link-Quality-Report uses the Interface counters defined by SNMP
MIB-II [2]. These counters are not initialized to any particular
value when the LCP enters the Establishment phase.

In addition, the Link-Quality-Report requires the implementation of
the following three unsigned, monotonically increasing counters which
conform to the type and size requirements for SNMP MIB Counters [3].

OutLQRs

   OutLQRs is a 32-bit counter which increases by one for each
   tranmitted Link-Quality-Report packet. This counter MUST be set
   to zero when the LCP enters the Establishment phase, and MUST NOT
   be reset until the LCP leaves the Termination phase. This counter
   is incremented before it is inserted into the LQR packet.

InLQRs

   InLQRs is a 32-bit counter which increases by one for each
   received Link-Quality-Report packet. This counter MUST be set to
   zero when the LCP enters the Establishment phase, and MUST NOT be
   reset until the LCP leaves the Termination phase. This counter is
   incremented before it is inserted (in an implementation dependent
   fashion) into the LQR packet.

InGoodOctets

   InGoodOctets is a 32-bit counter which increases by the number of
   octets in each successfully received Data Link Layer packet.
   Unlike the MIB ifInOctets, octets for frames which are counted in
   ifInDiscards and ifInErrors MUST NOT be counted. This counter MAY
   be set to any initial value when the LCP enters the Establishment
   phase, but MUST NOT be reset until the LCP leaves the Termination
   phase.

2.3. Counting Packets and Octets

   The intent of the counters is to provide an indication of the amount
   of information passing over the link, rather than an actual
   measurement of the total bandwidth used. This specification is
   designed to yield the same count in various circumstances, such as
   when a separate device provides the framing and escaping mechanisms
   invisibly to the implementation, or a synchronous-to-asynchronous
   converter in the link changes between mechanisms.

   All octets which are included in the FCS calculation MUST be counted,
   including the packet header, the information field, and any padding.
   The FCS octets MUST also be counted, and one flag octet per frame
   MUST be counted. All other octets (such as additional flag
   sequences, and escape bits or octets) MUST NOT be counted.

   When inserting the packet and octet counts in the LQR, the counts
   MUST include the expected values for the LQR itself.

2.4. Processes

   The PPP Link Quality Monitoring mechanism is described using a
   "logical process" model. As shown below, there are five logical
   processes duplicated at each end of the duplex link.

```
+---------+   +-------+   +----+ Outbound
|         |-->|  Mux  |-->| Tx |=========>
| Link-   |   +-------+   +----+
| Manager |
|         |   +-------+   +----+ Inbound
|         |<--| Demux |<--| Rx |<=========
+---------+   +-------+   +----+
```
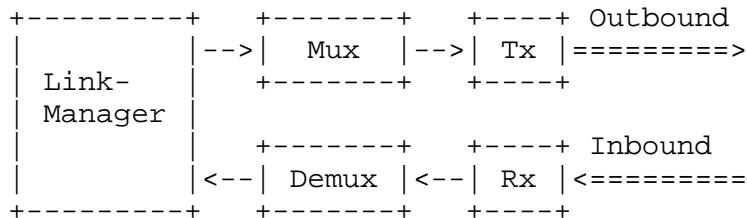
   Link-Manager

      The Link-Manager process transmits and receives Link-Quality-
      Reports, and implements the desired link quality policy. LQR
      packets are transmitted at a constant rate, which is negotiated by
      the LCP Quality-Protocol Configuration Option.

   Mux

      The Mux process multiplexes packets from the various protocols
      (e.g., LCP, IP, XNS, etc.) into a single, sequential, and
      prioritized stream of packets. Link-Quality-Report packets MUST
      be given the highest possible priority to insure that link quality
      information is communicated in a timely manner.

   Tx

      The Tx process maintains the MIB counters ifOutUniPackets and
      ifOutOctets, and the internal counter OutLQRs, which are used to
      measure the amount of data which is transmitted on the outbound
      link. When Tx processes a Link-Quality-Report packet, it inserts
      the values of these counters into the corresponding PeerOut...
      fields of the packet. The Tx process MUST follow the Mux process
      so that packets are counted in the order transmitted to the link.

   Rx

      The Rx process maintains the MIB counters ifInUniPackets,
      ifInDiscards, ifInErrors and IfInOctets, and the internal counters
      InLQRs and InGoodOctets, which are used to measure the amount of
      data which is received by the inbound link. When Rx processes a
      Link-Quality-Report packet, it inserts the values of these
      counters into the corresponding SaveIn... fields of the packet (in

an implementation dependent manner).

Demux

The Demux process demultiplexes packets for the various protocols.
The Demux process MUST follow the Rx process so that packets are
counted in the order received from the link.

2.5. Configuration Option Format

Description

Implementations MUST be prepared to receive the Quality-Protocol
Configuration Option for the Link-Quality-Report. However,
negotiation is not required. Negotiation is only necessary when
the implementation wishes to ensure that the peer transmits Link-
Quality-Reports as opposed to some other Quality-Protocol, or else
to prevent the peer from maintaining its own timer, or else to
establish a maximum time between transmissions of Link-Quality-
Reports.

A summary of the Quality-Protocol Configuration Option format to
negotiate the Link-Quality-Report 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       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Reporting-Period                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
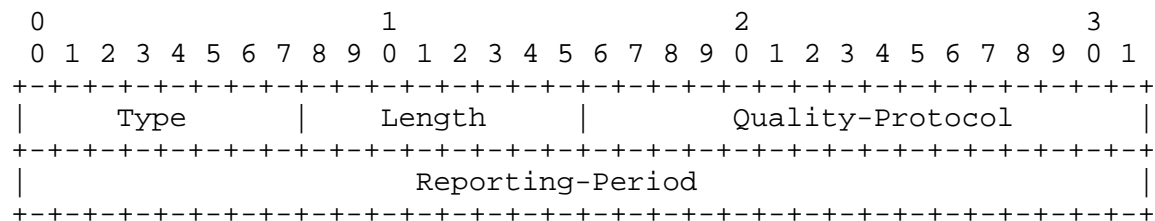

Type

    4

Length

    8

Quality-Protocol

    c025 (hex) for Link-Quality-Report

Reporting-Period

   The Reporting-Period field is four octets and indicates the
   maximum time in hundredths of seconds between transmission of
   packets. The peer MAY transmit packets at a faster rate than that
   which was negotiated.

   A value of zero indicates that the peer does not need to maintain
   a timer. Instead, the peer generates a LQR immediately upon
   receiving a LQR. A value of zero MUST be Nak'd by the peer with
   an appropriate non-zero value when that peer has sent or will send
   a Configure-Request packet containing the Quality-Protocol
   Configuration Option for the Link-Quality-Report with a zero
   Reporting-Period.

2.6. Packet Format

   Exactly one Link-Quality-Report packet is encapsulated in the
   Information field of PPP Data Link Layer frames where the protocol
   field indicates type hex c025 (Link-Quality-Report). A summary of
   the LQR packet format is shown below. The names of the fields are
   relative to the packet receiver, since it is the receiver who
   requested the packet in the Configuration Option. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Magic-Number                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          LastOutLQRs                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         LastOutPackets                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         LastOutOctets                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           PeerInLQRs                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         PeerInPackets                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         PeerInDiscards                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          PeerInErrors                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          PeerInOctets                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          PeerOutLQRs                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         PeerOutPackets                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          PeerOutOctets                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   The following fields are not actually transmitted over the inbound
   link. Rather, they are logically appended (in an implementation
   dependent manner) to the packet by the implementation's Rx process.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          SaveInLQRs                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SaveInPackets                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SaveInDiscards                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SaveInErrors                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         SaveInOctets                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Magic-Number

   The Magic-Number field is four octets and aids in detecting links
   which are in the looped-back condition. Unless modified by a
   Configuration Option, the Magic-Number MUST be transmitted as zero
   and MUST be ignored on reception. If Magic-Numbers have been
   negotiated, incoming LQR packets SHOULD be checked to ensure that
   the local end is not seeing its own Magic-Number and thus a
   looped-back link. See the Magic-Number Configuration Option for
   further explanation.

LastOutLQRs

   The LastOutLQRs field is four octets, and is copied from the most
   recently received PeerOutLQRs on transmission.

LastOutPackets

   The LastOutPackets field is four octets, and is copied from the
   most recently received PeerOutPackets on transmission.

LastOutOctets

   The LastOutOctets field is four octets, and is copied from the
   most recently received PeerOutOctets on transmission.

PeerInLQRs

   The PeerInLQRs field is four octets, and is copied from the most
   recently received SaveInLQRs on transmission.

   Whenever the PeerInLQRs field is discovered to be zero, the
   LastOut... fields are indeterminate, and the PeerIn... fields
   contain the initial values for the peer.

PeerInPackets

   The PeerInPackets field is four octets, and is copied from the
   most recently received SaveInPackets on transmission.

PeerInDiscards

   The PeerInDiscards field is four octets, and is copied from the
   most recently received SaveInDiscards on transmission.

PeerInErrors

   The PeerInErrors field is four octets, and is copied from the most
   recently received SaveInErrors on transmission.

PeerInOctets

   The PeerInOctets field is four octets, and is copied from the most
   recently received SaveInOctets on transmission.

PeerOutLQRs

   The PeerOutLQRs field is four octets, and is copied from OutLQRs
   on transmission. This number MUST include this LQR.

PeerOutPackets

   The PeerOutPackets field is four octets, and is copied from the
   current MIB ifOutUniPackets and ifOutNUniPackets on transmission.
   This number MUST include this LQR.

PeerOutOctets

   The PeerOutOctets field is four octets, and is copied from the
   current MIB ifOutOctets on transmission. This number MUST include
   this LQR.

SaveInLQRs

   The SaveInLQRs field is four octets, and is copied from InLQRs on
   reception. This number MUST include this LQR.

SaveInPackets

   The SaveInPackets field is four octets, and is copied from the
   current MIB ifInUniPackets and ifInNUniPackets on reception. This
   number MUST include this LQR.

SaveInDiscards

   The SaveInDiscards field is four octets, and is copied from the
   current MIB ifInDiscards on reception. This number MUST include
   this LQR.

SaveInErrors

   The SaveInErrors field is four octets, and is copied from the
   current MIB ifInErrors on reception. This number MUST include
   this LQR.

SaveInOctets

   The SaveInOctets field is four octets, and is copied from the
   current InGoodOctets on reception. This number MUST include this
   LQR.

   Note that InGoodOctets is not the same as the MIB ifInOctets
   counter, as InGoodOctets does not include octets for packets which
   are discards or errors.

2.7. Transmission of Reports

   When the PPP Link Control Protocol has reached the Opened state, the
   Link Quality Monitoring process MAY commence sending Link-Quality-
   Reports. If a Protocol-Reject is received specifying a LQR packet,
   the LQM process MUST cease sending LQRs.

   Usually, the LQR is transmitted when the LQR timer for the link
   expires. If no LQR timer is used, a LQR is generated upon receipt of
   an incoming LQR. The negotiation process ensures that at least one
   side of the link is using a LQR timer.

   In addition, a LQR is generated whenever two successive LQRs are
   received which have the same PeerInLQRs value. This may indicate
   that a LQR has been missed, or that the implementation is sending at
   a significantly slower rate than the peer, or that the peer has
   accelerated LQR generation to better quantify errors on the link.

   Whenever a LQR is sent, the LQR timer MUST be restarted.

2.8. Calculations

   Each time a Link-Quality-Report packet is received from the inbound
   link, the Link-Manager can compare the associated fields. The fields
   of the previous LQR can be subtracted from the current LQR values to
   obtain an absolute "delta", which allows comparision of the changes
   seen by each end of the link.

   If the received PeerInLQRs field is zero, the LastOut... fields are
   indeterminate, and the PeerIn... fields contain the initial values
   for the peer. No calculations using these fields can be performed at
   this time.

   Implementation Note:

      The following counters wrap to zero when their maximum value is
      reached. Care must be taken to ensure that correct "delta"
      calculations are performed at that time.

   The LastOutLQRs field may be directly compared with the PeerInLQRs
   field to determine how many outbound LQRs have been lost.

   The LastOutLQRs field may be directly compared with the OutLQRs
   counter to determine how many outbound LQRs are still in the
   pipeline.

The change in PeerInPackets may be compared with the change in
LastOutPackets to determine the number of lost packets over the
outgoing link.

The change in PeerInOctets may be compared with the change in
LastOutOctets to determine the number of lost octets over the
outgoing link.

The change in SaveInPackets may be compared with the change in
PeerOutPackets to determine the number of lost packets over the
incoming link.

The change in SaveInOctets may be compared with the change in
PeerOutOctets to determine the number of lost octets over the
incoming link.

The change in the PeerInDiscards and PeerInErrors fields may be used
to determine whether packet loss is due to congestion in the peer
rather than physical link failure.

2.9. Failure Detection

   When the link is operating well in both directions of the link, the
   LQR is superfluous. The maximum time interval for transmitting LQRs
   SHOULD be chosen to minimally interfere with active traffic.

   When there is a measurable loss of data in either direction, if the
   overall throughput is adequate, conditions are not severe enough to
   warrant dropping the link. Sending LQRs faster will gain nothing,
   except to measure peaks in the loss rate. The time interval MUST be
   chosen to be long enough to have a good smoothing effect on the data,
   while short enough to ensure fast enough response to complete
   failure.

   When the link is good incoming, but very bad outgoing, incoming LQRs
   indicate a high loss on the outgoing side of the link. Sending LQRs
   faster won't help, because they are probably lost on the way to the
   peer.

   When the link is good outgoing, but very bad incoming, incoming LRQs
   will be frequently lost. In this case, LQRs SHOULD be sent at a
   faster rate. This primarily relies on the peer to make an informed
   policy decision. The peer will also send LQRs in response (due to
   the duplicate PeerInLQRs field), and some of those LQRs may
   successfully arrive.

When a LQR does not arrive within the time expected, or the LQR
received indicates that the links are truly bad, at least one
additional LQR SHOULD be sent. An algorithmic decision requires at
least 2 round trip intervals. The loss rate could be transient, due
to a heavily loaded link, or a lost outgoing LQR.

2.10. Policy Suggestions

Link-Quality-Report packets provide a mechanism to determine the link
quality, but it is up to each implementation to decide when the link
is usable. It is recommended that this policy implement some amount
of hysteresis so that the link does not bounce up and down. One
policy is to use a K out of N algorithm. In such an algorithm, there
must be K successes out of the last N periods for the link to be
considered of good quality.

Procedures for recovery from poor quality links are unspecified and
may vary from implementation to implementation. A suggested approach
is to immediately close all other Network-Layer protocols (i.e.,
cause IPCP to transmit a Terminate-Request), but to continue
transmitting Link-Quality-Reports. Once the link quality again
reaches an acceptable level, Network-Layer protocols can be
reconfigured.

Security Considerations

    Security issues are not discussed in this memo.

Acknowledgements

    Some of the text in this document is taken from RFC 1172, by Drew
    Perkins of Carnegie Mellon University, and by Russ Hobby of the
    University of California at Davis.

    Special thanks to Craig Fox (Network Systems), and Karl Fox (Morning
    Star Technologies), for design suggestions based on implementation
    experience.

References

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

    [2]   McCloghrie, K., and M. Rose, "Management Information Base for
           Network Management of TCP/IP-based internets: MIB-II", STD
           17, RFC 1213, March 1991.

    [3]   Rose, M., and K. McCloghrie, "Structure and Identification of
           Management Information for TCP/IP-based Internets", STD 16,
           RFC 1155, May 1990.

Chair's Address

   The working group can be contacted via the current chair:

   Karl Fox
   Ascend Communications
   3518 Riverside Drive, Suite 101
   Columbus, Ohio 43221

   EMail: karl@ascend.com

Author'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 (prefered)