INTERNET-DRAFT
Expires: September 1997

Carsten Bormann
Universitaet Bremen
March 1997

The Multi-Class Extension to Multi-Link PPP
draft-ietf-issll-isslow-mcml-01.txt

Status of this memo

Abstract

A companion document describes an architecture for providing integrated services over low-bitrate links, such as modem lines, ISDN B-channels, and sub-T1 links [1]. The main components of the architecture are: a real-time encapsulation format for asynchronous and synchronous low-bitrate links, a header compression architecture optimized for real-time flows, elements of negotiation protocols used between routers (or between hosts and routers), and announcement protocols used by applications to allow this negotiation to take place.

This document proposes the fragment-oriented solution for the real-time encapsulation format part of the architecture. The general approach is to start from the PPP Multilink fragmentation protocol [2] and provide a small number of extensions to add functionality and reduce the overhead.

This document is a product of the IETF ISSLL working group. Comments are solicited and should be addressed to the two working groups' mailing lists at issll@mercury.lcs.mit.edu and ietf-

ppp@merit.edu and/or the author.

1. Introduction

As an extension to the ``best-effort'' services the Internet is well-known for, additional types of services (``integrated services'') that support the transport of real-time multimedia information are being developed for and deployed in the Internet.

A companion document describes an architecture for providing integrated services over low-bitrate links, such as modem lines, ISDN B-channels, and sub-T1 links [1]. The main components of the architecture are: a real-time encapsulation format for asynchronous and synchronous low-bitrate links, a header compression architecture optimized for real-time flows, elements of negotiation protocols used between routers (or between hosts and routers), and announcement protocols used by applications to allow this negotiation to take place.

The present document defines the fragment-oriented solution for the real-time encapsulation format part of the architecture, i.e. for the queues-of-fragments type sender [1]. As described in more detail in the architecture document, a real-time encapsulation format is required as, e.g., a 1500 byte packet on a 28.8 kbit/s modem link makes this link unavailable for the transmission of real-time information for about 400 ms. This adds a worst-case delay that causes real-time applications to operate with round-trip delays on the order of at least a second -- unacceptable for real-time conversation. The PPP extensions defined in this document allow a sender to fragment the packets of various priorities into multiple classes of fragments, allowing high-priority packets to be sent between fragments of lower priorities.

A companion document based on these extensions [5] defines a suspend/resume-oriented solution for those cases where the best possible delay is required and the senders are of type 1 [1].

2. Requirements

The main design goal for the components of an architecture that addresses real-time multimedia flows over low-bitrate links is that of minimizing the end-to-end delay. More specifically, the worst case delay (after removing possible outliers, which are equivalent to packet losses from an application point of view) is what determines the playout points selected by the applications and thus the delay actually perceived by the user.

In addition, every attempt should obviously be undertaken to maximize the bandwidth actually available to media data; overheads must be minimized.

The solution should not place unnecessary burdens on the non-real-time flows. In particular, the usual MTU should be available to these flows.

The most general approach would provide the ability to suspend any packet (real-time or not) for a more urgent real-time packet, up to an infinite number of levels of nesting. On the other hand, it is likely that there would rarely be a requirement for a real-time packet to suspend another real-time packet that is not at least about twice as long. Typically, the largest packet size to be expected on a PPP link is the default MTU of 1500 bytes. The smallest high-priority packets are likely to have on the order of 21 bytes (compressed RTP/G.723.1 packets). In the 1:72 range of packet sizes to be expected, this translates to a maximum requirement of about eight levels of suspension (including one level where long real-time packets suspend long non-real-time packets). On 28.8kbit/s modems, there seems to be a practical requirement for at least two levels of suspension (i.e., audio suspends any longer packet including video, video suspends other very long packets).

On an architectural level, there are several additional requirements for the fragmentation scheme:

a) The scheme must be predictable enough that admission control can make decisions based on its characteristics. As is argued in [1], this will often only be the case when additional hints about the characteristics of the flow itself are available (application hints).

b) The scheme must be robust against errors, at least with the same level of error detection as PPP.

c) The scheme must in general cooperate nicely with PPP. In particular, it should be as compatible to existing PPP standards as possible. On a link that (based on PPP negotiation) makes use of the scheme, it should always be possible to fall back to standard LCP without ambiguity.

d) The scheme must work well with existing chips and router systems. (See [1] for a more extensive discussion of implementation models.) For synchronous links this means using HDLC framing; with much existing hardware, it is also hard to switch off the HDLC per-frame CRC. For asynchronous links, there is much more freedom in design; on the other hand, a design that treats them much different from synchronous links would lose a number of desirable properties of PPP.

e) The scheme must be future proof. In particular, the emergence of V.80 based modems may significantly change the way PPP is used with modems.

The current draft does not address additional requirements that may be relevant in conjunction with Frame Relay; however, there seems to be little problem in applying the principles of this draft to ``PPP in Frame Relay'' [3].

3. Using existing mechanisms

Transmitting only part of a packet to allow higher-priority traffic to intervene and resuming its transmission later on is a kind of fragmentation. The purpose of this section is to examine existing mechanisms that are available for packet fragmentation and, by relating them to the requirements listed above, to outline their areas and limits of applicability.

Fragmentation is existing functionality of the IP layer. As fragmentation and reassembly also are useful in a logical link composed of multiple physical links, PPP Multilink (a standards track protocol) already defines a fragmentation mechanism [2]. Unfortunately, neither approach, as is, fulfills all the requirements listed above.

3.1. Using IP fragmentation

An IPv4 header already contains fields that allow a large IP datagram to be fragmented into small parts. A queues-of-fragments type sender [1] might simply indicate a small MTU to its IP stack and thus cause all larger datagrams to be fragmented down to a size that allows the access delay goals to be met[1]. (Also, a PPP implementation can negotiate down the MTU of its peer, causing the peer to fragment to a small size, which might be considered a crude form of negotiating an access delay goal with the peer system -- if that system supports priority queueing at the fragment level.)

Unfortunately, a full, 20 byte IP header is needed for each fragment (larger when IP options are used). This limits the minimum size of fragment that can be used without too much overhead. (Also, the size of non-final fragments must be a multiple of 8 bytes, further limiting the choice.) With path MTU discovery, IP level fragmentation causes TCP implementations to use small MSSs -- this further increases the per-packet overhead to 40 bytes per fragment.

In any case, fragmentation at the IP level persists on the path further down to the datagram receiver, increasing the transmission overheads and router load throughout the network. With its high overhead and the adverse effect on the Internet, IP level fragmentation can only be a stop-gap mechanism when no other fragmentation protocol is available in the peer implementation.

----

[1] This assumes that the IP stack is able to priority-tag fragments, or that the PPP implementation is able to correlate the fragments to the initial one that carries the information relevant for prioritizing, or that only initial fragments can be high-priority.

3.2. Using PPP Multilink as-is

The PPP Multilink Protocol (MP) provides for sequence numbering and begin/end bits, allowing packets to be split into fragments.

Figure 1: Multilink Short Sequence Number Fragment Format [2]

```
                +---------------+---------------+
PPP Header:     | Address 0xff  | Control 0x03  |
                +---------------+---------------+
                | PID(H)  0x00  | PID(L)  0x3d  |
                +-+-+-+-+-------+---------------+
   MP Header:    |B|E|0|0|    sequence number    |
                +-+-+-+-+-------+---------------+
                |    fragment data              |
                |               .               |
                |               .               |
                |               .               |
                +---------------+---------------+
   PPP FCS:      |              FCS              |
                +---------------+---------------+
```

(Note that the address, control, and most significant PID bytes are often negotiated to be compressed away.)

MP's monotonically increasing sequence numbering (contiguous numbers are needed for all fragments of a packet) does not allow to suspend sending a sequence of fragments of one packet for sending another packet. It is, however, possible to send intervening packets that are not encapsulated in multilink headers; thus, MP supports two levels of priority.

The multilink-as-is approach can be built using existing standards; multilink capability is now widely deployed and only the sending side needs to be aware that they are using this for giving priority to real-time packets.

3.3. Limitations of multilink as-is

Multilink-as-is is not the complete solution for a number of reasons. First, because of the single monotonically increasing serial number, there is only one level of suspension: ``Big'' packets that are sent via multilink can be suspended by ``small'' packets sent outside of multilink; the latter are not fragmentable.

A problem not solved by this specification is that the multi-link header is relatively large; as delay bounds become small (for queues-of-fragments type implementations) the overhead may become significant.

The general approach of this document is to start from PPP Multilink and provide a number of extensions to add functionality and reduce the overhead of using PPP Multilink for real-time transmission.

4. Extending PPP Multilink to multiple classes

The obvious approach to providing more than one level of suspension with PPP Multilink is to run Multilink multiple times over one link. Multilink as it is defined provides no way for more than one instance to be active. Fortunately, a number of bits are unused in the Multilink header: two bits in the short sequence number format (as can be seen in Figure 1), six in the long sequence number format.

This document defines (some of the) previously unused bits as a class number:

```
    Figure 2: Short Sequence Number Fragment Format With Classes
             +---------------+---------------+
PPP Header:  | Address 0xff  | Control 0x03  |
             +---------------+---------------+
             | PID(H)  0x00  | PID(L)  0x3d  |
             +-+-+-+-+-------+---------------+
MP Header:   |B|E|cls|    sequence number    |
             +-+-+-+-+-------+---------------+
             |    fragment data              |
             |               .               |
             |               .               |
             |               .               |
             +---------------+---------------+
PPP FCS:     |              FCS              |
             +---------------+---------------+
```

Each class runs a separate copy of the mechanism defined in [2], i.e. uses a separate sequence number space and reassembly buffer.

Similarly, for the long sequence number format:

Figure 3: Long Sequence Number Fragment Format With Classes

```
                +---------------+---------------+
   PPP Header:  | Address 0xff  | Control 0x03  |
                +---------------+---------------+
                | PID(H)  0x00  | PID(L)  0x3d  |
                +-+-+-+-+-+-+-+-+---------------+
   MP Header:   |B|E| class |0|0|sequence number|
                +-+-+-+-+-+-+-+-+---------------+
                |      sequence number (L)      |
                +---------------+---------------+
                |        fragment data          |
                |               .               |
                |               .               |
                |               .               |
                +---------------+---------------+
   PPP FCS:     |              FCS              |
                +---------------+---------------+
```

Together with the ability to send packets without a multilink header, this provides four levels of suspension with 12-bit headers (probably sufficient for many practical applications) and sixteen levels with 24-bit headers (only four of the six free bits are used in this case -- based on the rationale given above, sixteen levels should generally be more than sufficient).

5. Prefix elision: Compressing common header bytes

For some applications, all packets of a certain class will have a common protocol identifier (or even more than one common prefix byte). In this case, the following optimization is possible: the class number can be associated with a prefix of bytes that are removed from each packet before transmission and that are implicitly prepended to the reassembled packet after reception.

Note that if only some of the packets to be transmitted at a certain level of priority have the common prefix, it may still be possible to utilize this method by allocating two class numbers and only associating one of them with the prefix. (This is the reason why four of the unused bits in the long sequence number format have been allocated to the class number instead of the three that generally should suffice.)

Prefix elision is not a replacement for header compression or data compression: it allows to compress away prefixes that often are not reachable by these other methods.

6. Negotiable options

The following options are already defined by MP:

o Multilink Maximum Received Reconstructed Unit

o Multilink Short Sequence Number Header Format

o Endpoint Discriminator

This document defines two new options:

o Multilink Header Format

o Prefix Elision

6.1. Multilink header format option

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

Figure 4:

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

This option advises the peer that the implementation wishes to receive fragments with a format given by the code number. By default, long sequence number multilink headers without classes are used. When this option is received, an implementation MUST either transmit all subsequent multilink packets on all links of the bundle with the multilink header format given or configure-NAK or configure-Reject the option.

The values defined for the use of this option are:

- Neither this option nor the Short Sequence Number Header Format Option (type 18) [2] is present: long sequence number fragment format

- This option present with code = 2: long sequence number fragment format with classes

- Short Sequence Number Header Format Option (type 18) present: short sequence number fragment format

- This option present with code = 6: short sequence number fragment format with classes

An implementation MUST NOT request a combination of both the Short Sequence Number Header Format Option and this option.

6.2. Prefix elision option

This option advises the peer that the implementation wishes to send only packets with a certain prefix in each of the given classes; the prefix is not sent as part of the information in the fragment(s) of this class. By default, this common prefix is empty for all classes. When this option is received, an implementation MUST either add the prefix given for the class to all subsequently received multilink packets of each of the given classes on all links of the bundle or configure-NAK or configure-Reject the option.

If none of the formats with classes has been negotiated, class 0 is used to indicate a common prefix for all packets sent within multilink fragments.

Figure 5:

```
 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 = TBD  | Option Length |     Class     | Prefix Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Prefix...
+-+-+-+-+-+-+-+-+
```

NOTA BENE: the sense of this option is an indication from the sender to the receiver, UNLIKE most PPP options that indicate capabilities of the receiver to the sender.

7. Acknowledgements

David Oran suggested using PPP Multilink for real-time framing and reminded the author of his earlier attempts of making Multilink more useful for this purpose. The participants in a lunch BOF at the Montreal IETF gave useful input on the design tradeoffs in various environments. The members of the ISSLL subgroup on low bitrate links (ISSLOW) have helped reducing the large set of options that initial versions of this draft had.

8. References

[1] C. Bormann, Providing integrated services over low-bitrate links, work in progress (draft-ietf-issll-isslow-01.txt), February 1997.

[2] K. Sklower, B. Lloyd, G. McGregor, D. Carr, T. Coradetti, ``The PPP Multilink Protocol (MP)'', RFC 1990, August 1996 (obsoletes RFC1717).

[3] W. Simpson, ``PPP in Frame Relay'', RFC 1973, June 1996.

[4] R. Andrades, F. Burg, ``QOSPPP Framing Extensions to PPP'', September 20, 1996, Work in Progress (draft-andrades-framing-ext-00.txt).

[5] C. Bormann, ``PPP in a real-time oriented HDLC-like framing, internet Draft draft-ietf-issll-isslow-rtf-00.txt, Work in Progress, March 1997.

9. Addresses

9.1. Working Group

The ISSLL working group can be contacted via the co-chairs, Eric Crawley <esc@baynetworks.com> and John Wroclawski <jtw@lcs.mit.edu>, or via its WG mailing list <issll@mercury.lcs.mit.edu>.

9.2. Author's address

Carsten Bormann
Universitaet Bremen FB3 TZI
Postfach 330440
D-28334 Bremen, GERMANY
cabo@tzi.uni-bremen.de
phone +49.421.218-7024
fax +49.421.218-7000