The MIDI Wire Protocol Packetization (MWPP)

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Abstract

This memo describes the MIDI Wire Protocol Packetization (MWPP). MWPP is a resilient RTP packetization for the MIDI wire protocol. MWPP defines a multicast-compatible recovery journal format, to support the graceful recovery from lost packets during a MIDI session. MWPP is compatible with the MPEG-4 generic RTP payload format, to support MPEG 4 Audio codecs that accept MIDI control input.
0. Changes from <draft-lazzaro-avt-mwpp-midi-nmp-00.txt>

The document has been extensively changed, in response to WG comments at Salt Lake and MIDI developer comments on the Linux Audio Developers mailing list. Major changes are listed below.

- Normative algorithm specifications for the sending and receiving of MWPP packets have been deleted from the memo. Normative text is confined to the packetization format (in Sections 2-4 and Appendices A.1-6), a new policy for resilient sending (Section 5), and SDP issues (Section 6 for standard RTP, Section 7 for MPEG 4 generic RTP).

- The memo now casts MWPP as a general-purpose transport for the MIDI wire protocol; text in the former document about network musical performance specialization has been deleted.

- MWPP no longer uses the MPEG 4 Structured Audio standard as a normative reference. The only MPEG issue left in the document concerns MWPP’s dual role as both a standalone RTP packetization and an MPEG-4 generic RTP packetization.

- The MIDI command payload of a packet now specifies the event time of each MIDI command in the payload.

- The marker bit in the RTP header is now always set to 1. This modification lets us define a single MWPP payload format that is compatible with both standalone RTP and MPEG-4 generic RTP transport.

- In the recovery journal header, we replace the redundant K flag bit with a new "G" (guaranteed) flag bit. The G flag bit codes that the sender is following the sending policy defined in Section 5; this sending policy provides the "graceful recovery upon receipt of the first packet following a loss" guarantee which motivates the recovery journal concept.

- The "mpeg-generic" SDP typo was also fixed, and is now "mpeg4-generic."

- Sender and receiver proxy discussions have been deleted.

- New name reflects MWPP’s AVT WG item status.
Several work items remain, and are listed below. These items are a consequence of recasting MWPP as a general-purpose MIDI packetization.

- Currently, MWPP does not support MIDI Systems commands. This decision was originally made because Structured Audio does not support MIDI Systems commands, and needs to be revisited.

- The recovery journal chapter for the MIDI Control Change command, detailed in Appendix A.6, reflects the limited usage of MIDI Control Change commands by MPEG 4 Structured Audio. Modifications of this chapter may be necessary to properly support the full semantics of the MIDI Control Change command.
1. Introduction

The MIDI standard [1] defines a real-time networking standard for the interconnection of electronic musical devices and general-purpose computers. The standard defines the MIDI command set, a wire protocol for the command set, and a physical layer to carry the wire protocol (short coaxial "MIDI cables"). This memo concerns the transport of the MIDI wire protocol on alternative network layers, using the Real-Time Protocol (RTP).

This memo describes the MIDI Wire Protocol Packetization (MWPP), a resilient RTP [2] payload format for the MIDI wire protocol. MWPP is defined as a stand-alone RTP payload. However, MWPP is also suitable for use in conjunction with the MPEG-4 generic RTP payload format [3] [4], to support MPEG codecs that accept MIDI control input [5]. MWPP normatively specifies a payload format, but does not specify algorithms for sending and receiving MWPP packets.

MWPP is designed for use over unreliable datagram transport such as unicast and multicast UDP: the design goal is graceful recovery from lost packets, without using packet retransmission. MWPP also supports reliable transport such as TCP. MWPP is self-framing, to simplify TCP transport.

Sending the MIDI wire protocol over unreliable transport is not trivial. The MIDI standard defines a set of commands that reflect the gestures musicians make in playing their instruments ("NoteOn" command to start a new note, "NoteOff" command to end the note, etc). Gestural commands make MIDI data streams very compact, but also very fragile: a single lost "NoteOff" command could result in a sound that sustains indefinitely long.

MWPP does not use packet retransmission to provide resiliency. Instead, each MWPP packet includes a special section (the "recovery journal") that codes the recent history of the stream. The recovery journal protects against the loss of RTP packets sent since an earlier "checkpoint" RTP packet.

The remainder of this memo defines the MWPP payload format, and specifies Session Description Protocol configuration for both RTP and MPEG-4 generic transport.

This memo describes a format, not an algorithm or an application. Readers unfamiliar with the application domain should first read [6], a paper that describes an experimental system [7] that uses an RTP packetization similar to MWPP. In addition, [8] describes another experimental system for MIDI transport, whose algorithms are compatible with MWPP.
2. MWPP Packet Format.

Figure 1 shows the format of an MWPP packet, suitable for both RTP transport and MPEG 4 generic RTP transport. An MWPP packet consists of three sections: the RTP header, the MIDI command section, and the recovery journal. In Figure 1, vertical space delineates the RTP header and the payload.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |        Sequence number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             SSRC                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             CSRCs                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     MIDI command section ...                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Recovery journal ...                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1 -- MWPP packet format

An MWPP packet begins with an RTP header. The marker bit is always set to 1, for compatibility with the MPEG 4 generic payload format. The RTP sequence number increments by one (modulo 65536) for each packet sent. MWPP does not use header extensions.

The RTP timestamp sets a base timestamp value for the packet. The event times coded in the MIDI command section are specified relative to this base timestamp value. If the MIDI command section carries no events, the timestamp indicates the instant the RTP packet was sent.

The RTP timestamp has the units of the SDP rtpmap attribute srate (see Section 6). For example, if srate has a value of 44100 (Hz), two MWPP packets whose base timestamp values differ by 2 seconds have RTP timestamps that differ by 88200.
RTP timestamps do not increment at a fixed rate, but instead reflect the execution timing of the encoded MIDI data. The timestamps for two sequential RTP packets may be identical, or the second packet may have a timestamp arbitrarily larger than the first packet (modulo 32). As is standard in RTP, the timestamp field is initialized to a randomly chosen value.

MWPP does not provide tools to multiplex several 16-channel MIDI cable streams onto a single MWPP payload. Instead, implementors should use the multiplexing tools provided by RTP: each MIDI cable stream should map to a separate RTP stream, identified by a distinct SSRC value.

The MWPP payload always begins with the variable-length MIDI command section, described in detail in Section 3. If a stream is configured for resilient coding, the MIDI command section of every packet is followed by the variable-length recovery journal, described in detail in Section 4. If a stream is not configured for resiliency, the recovery journal never appears in the MWPP payload.

The SDP rtpmap attribute rj (see Section 6) configures an MWPP stream for resilient coding.

3. MIDI Command Section

Figure 2 shows the format of the variable-length MIDI command section.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      LEN      |          MIDI list ...                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Figure 2 -- MIDI command section**

The MIDI command section begins with a one-octet header. The 8-bit LEN field codes the length (in units of octets) of the MIDI list that follows the header. A LEN value of 0 is legal, and codes an empty MIDI list. If the MIDI list is empty, the RTP timestamp indicates the instant the RTP packet was sent.
If LEN is nonzero, the MIDI list has the structure shown in Figure 3.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    MIDI Command 0 ...     |     Delta time 1 ...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    MIDI Command 1 ...     |     ...                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ...                    |     Delta time N ...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    MIDI Command N ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3 -- MIDI list structure

The MIDI list always begins with one complete MIDI channel command (MIDI Command 0 in Figure 3). MIDI Command 0 must include a status byte; running status coding [1] is not permitted. The RTP timestamp encodes the execution time of the MIDI Command 0.

Following MIDI Command 0, the MIDI list structure may optionally encode a list of N additional complete MIDI channel commands. Each command is preceded by a delta time; the execution time for MIDI command K is the modulo-2 summation of the RTP timestamp and delta times 1 through K. These additional MIDI commands may use running status coding.

MWPP borrows its delta time encoding format from the MIDI File Standard [1]. Delta times encode 7, 14, 21, or 28 bit delta timestamps, using 1, 2, 3, or 4 octets. The MSB of each octet is reserved to code the number of octets in the timestamp, using a coding technique compatible with the MIDI command syntax.

One-Octet Delta Time:

   Encoded form: 0ddddddd
   Decoded form: 00000000 00000000 00000000 0ddddddd

Two-Octet Delta Time:

   Encoded form: 1ccccccc 0ddddddd
   Decoded form: 00000000 00000000 00cccccc cddddddd
Three-Octet Delta Time:

Encoded form: 1bbbbbbb 1ccccccc 0ddddd
Decoded form: 00000000 000bbbbb bbcccccc cddddddd

Four-Octet Delta Time:

Encoded form: 1aaaaaaa 1bbbbbbb 1ccccccc 0ddddd
Decoded form: 0000aaaa aaabbbbb bbcccccc cddddddd

Figure 4 -- Decoding delta time formats

Figure 4 shows how transform delta time formats into 32-bit unsigned integers suitable for modulo-2 summation with the RTP timestamp.

4. The Recovery Journal

This section introduces the structure of the recovery journal, and defines the bitfields of recovery journal headers. Appendices to this memo complete the bitfield definition of the recovery journal.

A recovery journal codes information about the MIDI command section of all previous packets in an MWPP stream, back to and including an earlier packet called the checkpoint packet. We identify the checkpoint packet by its sequence number. Note that the recovery journal for a packet does not contain information about the MIDI command section of its own packet.

The recovery journal has a three-level structure:

- Top-level header. Encodes recovery journal structure.

- Channel journal header. Encodes recovery information for a single MIDI channel.

- Chapters. Describes recovery information for a single MIDI command type.

Figure 5 shows the top-level structure of the recovery journal. A recovery journals consists of a 3-octet header, followed by a list of channel journals. Channel journals encode recovery information for a single MIDI channel.
If the A bit is set in the recovery journal header, the recovery journal is "empty", and contains no channel journals. If the A bit is clear, the channel journal list contains (TOTCHAN + 1) channel journals.

The recovery journal header includes an S bit. S bits appear on structures throughout the recovery journal format, with uniform semantics: if the S bit is set to 1, the structure does not encode information about the MIDI command section of the previous packet in the stream.

S bits support efficient recovery journal parsing in the common case of a single packet loss. A set S bit on the recovery journal header indicates the previous packet contained an empty MIDI command section.

The 16-bit Checkpoint Packet Seqnum field codes the sequence number of the checkpoint packet for this journal. The G ("guaranteed") bit specifies the method used to update the checkpoint packet; we describe the G bit in detail in Section 5.

Figure 6 shows the structure of a channel journal: a 3-octet header, followed by a list of leaf elements called chapters. A channel journal encodes information about MIDI commands on the MIDI channel coded by the 4-bit CHAN header field. The 10-bit LENGTH field codes the number of octets in the channel journal, including the header.
The third octet of the channel journal header is the Table of Contents (TOC) of the channel journal. The TOC is a set of bits to encode the presence of a chapter in the journal. Each chapter contains information about a certain class of MIDI command:

- Chapter P: MIDI Program Change (0xC)
- Chapter W: MIDI Pitch Wheel (0xE)
- Chapter N: MIDI NoteOff (0x8), NoteOn (0x9)
- Chapter A: MIDI Poly Aftertouch (0xA)
- Chapter T: MIDI Channel Aftertouch (0xD)
- Chapter C: MIDI Control Change (0xB)

Chapters appear in a list following the header, in order of their appearance in the TOC. The Appendices of this memo describe the bitfield format for each chapter.

5. Checkpoint Packet Policy

In this section, we describe a normative policy that MWPP sender implementations may use to update the checkpoint packet identity during an MWPP session. If this policy is in effect, a receiver is able to "gracefully" recover from the loss of an arbitrary number of packets, upon the receipt of the first packet following the loss; see Section 7 of reference [6] for details.

Senders that implement this policy SHOULD set the G bit on the top-level recovery journal header (Figure 5) to 1; senders that do not implement this policy MUST set the G bit to 0. If a sender starts a session with the policy in effect, and then later abandons the policy, it MUST set the G bit on all recovery journals sent after abandonment to 0, for the remainder of the session. Receivers SHOULD monitor the G bit and adjust its recovery procedure based on its state.

In this description, we specify the identity of the checkpoint packet by the extended sequence number of the packet as maintained by the sender. We assume that senders can compensate for sequence number rollover in the implementation of the policy.

In order to implement the policy, senders must not advance the checkpoint packet to extended sequence number N, until it has direct knowledge that all known receivers have received an MWPP RTP packet with extended sequence number M >= (N - 1). Senders may deduce this knowledge by examining the "last extended sequence number received" fields of the standard RTCP packets from each receiver, or may use other direct feedback mechanisms.
Senders may find that a receiver is not providing feedback for an extended period of time, and that the recovery journal size has grown unacceptably large as a result. To maintain the policy, the only acceptable action in this case is to drop the offending receiver from the session; a time-out mechanism may not be used in lieu of direct feedback to advance the checkpoint packet.

Note that the policy is in effect for "known receivers." If MWPP is sent over true multicast, the receiver may be processing MWPP packets before the sender is aware of its existence. Receiver implementors SHOULD be aware of this start-up phenomena, and adjust its recovery procedures accordingly.

6. Session Description Protocol for RTP Transport

This section describes Session Description Protocol (SDP) [9] definitions for MWPP transport directly over RTP. Section 8 describes the SDP definitions for MWPP transport over the MPEG-4 generic RTP payload format.

The MIME name for this packetization is mwpp. The SDP rtpmap attribute is declared as

\[ a=rtpmap: \text{<payload>} \text{mwpp/<srate>/<rj}> \]

The integer parameter \text{srate} codes the sampling rate used for the RTP timestamp field, and has the units of Hz.

The binary parameter \text{rj} codes the presence (1) or absence (0) of the recovery journal section in MWPP packets.

For example, the following lines bind the packetization to dynamic payload number 96, and specifies an \text{srate} of 44100 Hz and the presence of a recovery journal in each RTP packet:

\[ m=audio 5004 RTP/AVP 96 \]
\[ c=IN IP4 171.64.92.160 \]
\[ a=rtpmap: 96 mwpp/44100/1 \]
Note that the packetization does not directly support multiple 16-channel MIDI Input sources. Different UDP ports should be used in this case, each devoted to a single source:

m=audio 5004 RTP/AVP 96
c=IN IP4 171.64.92.160
a=rtpmap: 96 mwpp/44100/1
m=audio 5006 RTP/AVP 97
c=IN IP4 171.64.92.160
a=rtpmap: 97 mwpp/44100/1

7. Session Description Protocol for MPEG-4 generic transport

This section describes Session Description Protocol (SDP) definitions for the MPEG-4 generic RTP payload format [3] [4]. Note that MWPP as defined in this memo creates valid MPEG-4 generic RTP packets; only SDP customization is necessary.

The MIME name for this packetization is mpeg4-generic. The SDP rtpmap attribute is declared as

a=rtpmap: <payload> mpeg4-generic/<srate>/<rj>

The definitions of srate and rj are identical to the descriptions in Section 6.

The SDP fmpt command configures mpeg4-generic for MWPP transport, as shown below:

a=fmpt: <payload> streamtype=5; profile-level-id=15; mode=mwpp;

To signal SingleSL mode, we omit the ConstantSize and SizeLength format parameters from the fmpt command. If the MPEG 4 audio codec requires configuration data be sent via SDP, AudioSpecificConfig() may be added.

8. Security Considerations

Cryptographic authentication of incoming RTP and RTCP packets is highly recommended when using MWPP. Without such protections, attackers could forge MIDI commands into an ongoing streams, potentially damaging speakers and eardrums. An attacker could also craft RTP and RTCP packets to exploit known bugs in the client, and take effective control of a client machine.
9. Congestion Control

MWPP has congestion control issues that are unique for an RTP audio packetization. In certain applications such as network musical performance [6], the packet rate is linked to the gestural rate of a human performer.

MWPP implementations SHOULD sense the MIDI wire protocol stream for command patterns that result in excessive packet rates, and filter these streams as part of MWPP to reduce the packet rate.

Appendix A.1. Chapter P: MIDI Program Change

A channel journal contains Chapter P if a MIDI Program Change command on this channel is present in the MIDI command section of an earlier packet, back to and including the checkpoint packet. Figure A.1.1 shows the format for Chapter P.

```
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|   PROGRAM   |C| BANK-COARSE |F| BANK-FINE   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.1.1 -- Chapter P Format

The chapter has a fixed size of 24 bits. The PROGRAM field indicates the program value of the most recent MIDI Program Change command sent on this channel. The S bit is set to 1 if this most recent Program Change command did not appear in the previous packet in the stream (i.e. packet N-1, if the recovery journal is a part of packet N).

If a MIDI Control Change command for the Bank Select Coarse controller was sent before this Program Change command, the C bit is set to 1, and the BANK-COARSE field is the Bank Select Coarse controller value that was sent. The F bit and BANK-FINE field code the Bank Select Fine value in the same manner. The BANK-COARSE and BANK-FINE fields may reflect Control Change commands sent before the checkpoint packet.

Appendix A.2. Chapter W: MIDI Pitch Wheel

A channel journal contains Chapter W if a MIDI Pitch Wheel command on this channel is present in the MIDI command section of an earlier packet.
packet, back to and including the checkpoint packet. Figure A.2.1 shows the format for Chapter W.

```
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|     FIRST   |R|    SECOND   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.2.1 -- Chapter W Format

The chapter has a fixed size of 16 bits. The FIRST and SECOND fields are the 7-bit values of the first and second data bytes of the most recent Pitch Wheel command sent on this channel. The S bit is set to 1 if this most recent Pitch Wheel command did not appear in the previous packet in the stream. The R bit is reserved and set to 0.

Appendix A.3. Chapter N: MIDI NoteOff and NoteOn

A channel journal contains Chapter N if a MIDI Note On or a Note Off command on this channel is present in the MIDI command section of an earlier packet, back to and including the checkpoint packet.

In the description that follows, we consider MIDI Note On commands with zero velocity to be MIDI Note Off commands.

Figure A.3.1 shows the format for Chapter N.

```
  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 8 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|B|   LENGTH    |  LOW  | HIGH  |S|   NOTENUM   |Y|  VELOCITY   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|   NOTENUM   |Y|  VELOCITY   | ....                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   BITFIELD    |   BITFIELD    |     ....      |   BITFIELD    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.3.1 -- Chapter N Format

Chapter N codes information about Note On and Note Off commands by coding information about the MIDI note numbers referenced by these
commands. The chapter consists of a 2-octet header, and at least one of the following data structures:

- A variable-length note list, coding Note On information.
- A variable-length bitfield, coding Note Off information.

Information about a specific MIDI note number may appear in the note list (if the note number last appears in a Note On command) or the bitfield (if the note number last appears in a Note Off command) but never both.

The header for Chapter N, reproduced in Figure A.3.2, codes the size of the note list and bitfield structures.

```
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|B| LENGTH |   LOW  | HIGH  |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```  

Figure A.3.2 -- Chapter N Header

The 7-bit LENGTH field codes the number of 2-octet note logs in the note list. Zero is a valid value for LENGTH, and codes the empty note list.

The 4-bit fields LOW and HIGH determine the number of bitfield bytes that follow the note logs. A bitfield byte codes NoteOff information for eight consecutive MIDI note numbers, with the MSB representing the lowest note number. The MSB of the first bitfield byte codes the note number 16*LOW; the MSB of the last bitfield byte codes the note number 16*HIGH.

A 1 in a bit position codes that a Note Off command happened more recently than a Note On command for this note number, and that this Note Off command occurred in the MIDI command section of an earlier packet, back to and including the checkpoint packet. Note that because Chapter N codes the presence of a Note Off command using a single bit, the Note Off velocity value is not recorded.

If LOW is less than or equal to HIGH, there are (HIGH - LOW + 1) bitfield octets in the chapter. An empty bitfield structure is coded by setting LOW to 15 and HIGH to 0. The B bit is set to 1 if the MIDI command section of the previous packet did not include a Note Off command for this channel.
The note list structure consists of LENGTH 2-octet note logs. The note log structure is reproduced below.

```
0                   1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|   NOTENUM   |Y|  VELOCITY   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.3.3 -- Chapter N Note Log

A note log will exist for a note number (coded by the 7-bit NOTENUM field) if a Note On command happened more recently than a Note Off command for this note number, and if this Note On command occurred in the MIDI command section of an earlier packet, back to and including the checkpoint packet. A note number may not be represented by multiple note logs in the note list.

The 7-bit VELOCITY field codes the velocity value for this most-recent NoteOn command, and is never zero: Note On commands with zero velocity are treated as Note Off commands, and coded in the bitfield structure.

The S bit is set to 1 if the Note On command coded by the note log is not in the MIDI command section of the previous packet.

The note log does not contain the execution time of the Note On command it codes, for efficiency reasons. In lieu of a timestamp, the Y bit codes information about the execution time of the Note On command coded by the Note Log.

The Y bit is set to 1 if the most recent event coded in the MIDI command section of the packet containing the recovery journal is considered to be simultaneous with the Note On command coded by the note log. If the MIDI command section of the packet contains no events, Y is set to 1 if a hypothetical MIDI command occurring at the RTP timestamp time would be considered simultaneous. The definition of simultaneity is implementation dependent.

Appendix A.4. Chapter A: MIDI Poly Aftertouch

A channel journal contains Chapter A if a MIDI Poly Aftertouch command on this channel is present in the MIDI command section of an earlier packet, back to and including the checkpoint packet. Poly Aftertouch commands contained in packets previous to the checkpoint packet are never coded in Chapter A.
Figure A.4.1 shows the variable-length format for Chapter A.

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7
\end{array}
\]

Figure A.4.1 -- Chapter A format

The chapter consists of a 1-octet header, followed by a variable length list of 2-octet note logs. A note log appears for a note number if a Poly Aftertouch command is present for the note number in the MIDI command section of an earlier packet, back to and including the checkpoint packet. A note number may not be represented by multiple note logs in the list.

The 7-bit LENGTH field codes the number of note logs in the list, minus one. The expression \((1 + 2 \times (\text{LENGTH} + 1))\) yields the number of octets in Chapter A. The S bit in the header is set to 1 if the MIDI command section of the previous packet does not contain a Poly Aftertouch command on this channel.

Figure A.4.2 reproduces the note log structure of Chapter A.

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
0 & 1 & 2 & 3 & 4 & 5
\end{array}
\]

Figure A.4.2 -- Chapter A Note Log

The 7-bit PRESSURE field codes the pressure value of the most recent Poly Aftertouch command for the MIDI note number coded by the 7-bit NOTENUM field. The F bit is 1 if this most recent Poly Aftertouch command did not appear in the previous packet. The R bit is reserved, and is set to 0.
Appendix A.5. Chapter T: MIDI Channel Aftertouch

A channel journal contains Chapter T if a MIDI Channel Aftertouch command on this channel is present in the MIDI command section of an earlier packet, back to and including the checkpoint packet. Figure A.5.1 shows the format for Chapter T.

```
0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|S|   PRESSURE  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.5.1 -- Chapter T Format

The chapter has a fixed size of 8 bits. The 7-bit PRESSURE field holds the pressure value of the most recent Channel Aftertouch command sent on this channel. The S bit is set to 1 if this most recent Channel Aftertouch command for this channel did not appear in the previous packet in the stream.

Appendix A.6. Chapter C: MIDI Control Change

A channel journal contains Chapter C if a MIDI Control Change command on this channel is present in the MIDI command section of an earlier packet, back to and including the checkpoint packet. Control Change commands contained in packets previous to the checkpoint packet are never coded in Chapter C.

Figure A.6.1 shows the variable-length format for Chapter C.

```
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 8 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|S|  LENGTH     |F|  CONTROLLER |R| VALUE/COUNT |F| CONTROLLER  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|R| VALUE/COUNT |  ....                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure A.6.1 -- Chapter C format

The chapter consists of a 1-octet header, followed by a variable length list of 2-octet controller logs. A controller log appears for a
controller number if a Control Change command is present for the controller number in the MIDI command section of an earlier packet, back to and including the checkpoint packet. A controller number may not be represented by multiple controller logs in the list.

The 7-bit LENGTH field codes the number of controller logs in the list, minus one. The expression \((1 + 2 \times (\text{LENGTH} + 1))\) yields the number of octets in Chapter C. The S bit in the header is set to 1 if the MIDI command section of the previous packet does not contain a MIDI Control Change command on this channel.

Figure A.6.2 reproduces the note log structure of Chapter C.

```
0                   1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|  CONTROLLER |R| VALUE/COUNT |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.6.2 -- Chapter C Controller Log

The 7-bit CONTROLLER field identifies the controller number. For most controller numbers, the 7-bit VALUE/COUNT field codes the control value of the most recent Control Change command for this controller number. The F bit is 1 if this most recent Control Change command did not appear in the previous packet. The R bit is reserved, and is set to 0.

Chapter C uses a VALUE/COUNT field differently for a few controller numbers, as described below.

For the Sustain Pedal controller number, the VALUE/COUNT field has the value 0 if the most recent Sustain Pedal command codes a pedal release. However, if the most recent Sustain Pedal command codes a pedal depression, the VALUE/COUNT field codes the total number of Sustain Pedal depression commands present in the MIDI command section of all packets over the lifetime of the stream, including this most recent Sustain Pedal command. If this value exceeds 127, modulo arithmetic is used, but the value 0 is skipped.

For the All Notes Off or All Sound Off controller numbers, the VALUE/COUNT field codes the total number of commands for the controller number present in the MIDI command sections of all packets over the lifetime of the stream, including this most recent command. If this value exceeds 127, modulo arithmetic is used, but the value 0 is skipped.
Appendix B. Author Addresses

John Lazzaro (corresponding author)
UC Berkeley
CS Division
315 Soda Hall
Berkeley CA 94720-1776
Email: lazzaro@cs.berkeley.edu

John Wawrzynek
UC Berkeley
CS Division
631 Soda Hall
Berkeley CA 94720-1776
Email: johnw@cs.berkeley.edu

Appendix C. References


[8] Dominique Fober, Yann Orlarey, Stephane Letz. Real Time


Appendix D. Expiration Notice

This document expires August 9, 2002.