Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: November 6, 2015

I. Bouazizi
Samsung Research America
May 5, 2015

MPEG Media Transport Protocol (MMTP)
draft-bouazizi-tsvwg-mmtp-00

Abstract

The MPEG Media Transport Protocol (MMTP) is a transport protocol that is designed to support download, progressive download, and streaming applications simultaneously. MMTP provides a generic media streaming mode by optimizing the delivery of generic media data encapsulated according to the ISO-Base Media File Format (ISOBMFF). In the file delivery mode, MMTP supports the delivery of any type of file. MMTP may used in IP unicast or multicast delivery and supports both Forward Error Correction (FEC) and retransmissions for reliable delivery of content.

Table of Contents

1. Introduction

The MMT protocol is an application layer transport protocol that is designed to efficiently and reliably transport multimedia data. MMTP can be used for both timed and non-timed media data. It supports several features, such as media multiplexing and network jitter estimation. These features are designed to deliver content composed of various types of encoded media data more efficiently. MMTP may run on top of existing network protocols such as UDP and IP. In this specification, the carriage of data formatted differently than the MMTP payload format as specified in Section 4 by MMTP is not defined. The MMT protocol is designed to support a wide variety of applications and does not specify congestion control. The congestion control function is left for the application implementation. MMTP supports the multiplexing of different media data such as ISOBMFF files from various Assets over a single MMTP packet flow. It delivers multiple types of data in the order of consumption to the receiving entity to help synchronization between different types of media data without introducing a large delay or requiring large buffer. MMTP defines two packetization modes, Generic File Delivery mode as specified in Section 4.2 and the ISOBMFF mode as specified in Section 4.1. The former defines a mode for packetizing media data based on the size of the payload to be carried and the latter defines a mode for packetizing media data based on the type of data to be carried in the payload. MMTP supports simultaneous transmission of packets using the two different modes in a single delivery session. MMTP also provides means to calculate and remove jitter introduced by the underlying delivery network, so that constant end-to-end delay for data delivery can be achieved. By using the delivery timestamp field in the packet header, jitter can be precisely estimated without requiring any additional signalling information and protocols.

2. Rationale

MMTP provides a generic media transport protocol that inherently supports virtually any media type and codec. For this purpose, MMTP is designed to support a limited set of payload types agnostic to the media type or coding format, but providing generic information to serve the needs of different multimedia delivery services. The MMTP payload format is defined as a generic payload format for the packetization of media data. It is agnostic to media codecs used for encoded media data, so that any type of media data that are encapsulated in the ISOBMFF format can be packetized into MMTP payloads. The MMTP payload format also supports fragmentation and aggregation of data to be delivered. MMTP supports both streaming and download modes, where the streaming mode is optimized for packetized streaming of ISO-Base Media File formatted files (ISOBMFF mode) and the download mode allows for flexible delivery of generic files (GFD mode). In addition, MMTP delivers streaming support data such as Application Layer Forward Error Correction (AL-FEC) repair data.

2.1. Difference to RTP

The RTP protocol was initially designed to support multi-party real-timed communication conferencing over the Internet. Key concern at that time was scalability of RTP to a large number of participants and dealing with media synchronization. Consequently, the RTP protocol is a mixture of transport and presentation layer functions. RTP supports a wide range of media types and codecs through the definition of codec-specific payload formats.

A set of issues arise when deploying RTP for media delivery, some of which are provided in the following list:

Lack of Multiplexing: RTP usually requires two separate ports for every media session. Rich media services have several service components, each of which would require an RTP/RTCP port pair. Although some level of multiplexing is possible in RTP (i.e. RTP and RTCP multiplexing as defined in [RFC5761], it is not clear that all RTP implementations support it and this still does not solve the problem. This is one of the reasons the industry is moving towards HTTP-based streaming where a single port is used. MMTP uses a single port and multiplexes all media streams of a service as well as the related signaling and any non-real time objects into a single MMTP flow that is self-contained and self-describing.

Costly Server Maintenance One of the major issues with RTP is the costly operation of dedicated streaming servers that need to be updated to support any new media codecs. The server must be upgraded to support the new payload format for any new media codec that the service provider wishes to use. MMTP solves this issue by supporting a single payload format for media streaming based on the ISOBMFF file format. Any media codec that can be encapsulated into the ISOBMFF file format can be streamed without any modifications by an MMT server.

Coupling Delivery and Presentation RTP carries the presentation timestamp of the encapsulated media data, which corresponds to the sampling instant of the first media sample/sub-sample contained in the packet payload. As the delivery timestamp is not provided in RTP, it is often assumed that the presentation timestamp is equal to the delivery timestamp. This coupling may make sense for real-time conferencing use cases but is generally not useful for streaming of on-demand content as the receiver will not be aware of the exact delivery time and will usually use external media for controlling the presentation time (so that the RTP timestamp will only be use for intra-media synchronization). MMTP decouples media delivery and media presentation completely by carrying only the delivery timestamp at the MMTP protocol level. The presentation time is controlled by external Presentation Information that may as well be carried as part of the MMTP flow, whereas the intra-synchronization is provided by the ISOBMFF file format. The delivery timestamp may be used for de-jittering, retransmissions, and other purposes.

3. Packet Header Field

The MMTP header is of variable size, where the size of the header may be deduced from the header flags. In the MMTP header, all integer fields are carried in "big-endian" or "network order" format, so that the most significant byte is first. Bits marked as "reserved" (r) MUST be set to 0 by the sender and ignored by receivers in this version of the specification. Unless otherwise noted, numeric constants in this specification are in decimal form (base 10). The format of the MMTP header is depicted in Figure 1.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|C|FEC|r|X|R|RES|   type    |          packet_id            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             timestamp                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      packet_sequence_number                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            packet_counter                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       header_extension                    ....
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           payload data                    ....
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: MMTP Header

The function and length of each field in the MMTP header is specified as follows:

version (V): 2 bits

indicates the version number of the MMTP protocol. This field shall be set to "00" to comply with this specification.

packet_counter_flag (C): 1 bit

"1" in this field indicates that the packet_counter field is present.

FEC_type (FEC): 2 bits

indicates whether the payload carries FEC source data or repair data. Valid values of this field are listed in Table 1 below. Depending on the FEC scheme, additional payload header may be added, for instance to identify the contained symbol(s).

reserved (r): 1 bit

reserved for future use.

extension_flag (X): 1 bit

when set to "1" this flag indicates that the header_extension field is present.

RAP_flag (R): 1 bit

when set to "1" this flag indicates that the payload contains a Random Access Point (RAP) to the data stream of that data type. The exact semantics of this flag are defined by the data type itself. The RAP_flag shall be set to mark data units of Fragment Type value "0" and "1" and for data units that contain a sync sample or a fragment thereof in the case of timed media and for the primary item of non-timed data.

reserved (RES): 2 bits

reserved for future use.

type: 6 bits

this field indicates the type of payload data. Payload type values are defined in Table 2.

packet_id: 16 bits

this field is an integer value that can be used to identify related media data, for example media that belong to the same media asset. The packet_id is unique throughout the lifetime of the delivery session and for all MMTP flows delivered by the same MMTP sender.

packet_sequence_number: 32 bits

an integer value that is used to distinguish between packets that have the same packet_id. The value of this field should start from an arbitrary value and shall be incremented by one for each new MMTP packet. It wraps around to "0" after the maximum value is reached.

timestamp: 32 bits

specifies the time instance of MMTP packet delivery based on UTC. The format is the "short-format" as defined in clause 6 of [RFC5905], NTP version 4. This timestamp specifies the sending time at the first byte of MMTP packet. It is required that an MMTP sender should provide accurate time information that are synchronized with UTC.

packet_counter: 32 bits

an integer value for counting MMTP packets. It is incremented by 1 when an MMTP packet is sent regardless of its packet_id value. This field starts from arbitrary value and wraps around to "0" after its maximum value is reached.

header_extension:

this field contains user-defined information. The header extension mechanism is provided to allow for proprietary extensions to the payload format to enable applications and media types that require additional information to be carried in the payload format header. The header extension mechanism is designed in a such way that it may be discarded without impacting the correct processing of the MMTP payload. The header extension shall have the format as shown in Figure 2. This specification does not specify any particular header extension.

Value	Description
0	MMTP packet without AL-FEC protection
1	MMTP packet with AL-FEC protection (FEC source packet)
2	MMTP packet for repair symbol(s) (FEC repair packet)
3	Reserved for future use

Table 1: FEC Type

Value	Data type	Definition of data unit
0x00	ISOBMFF file	The packet carries a media-aware fragment of the ISOBMFF file
0x01	Generic object	The packet contains a generic object such as a complete ISOBMFF file or an object of another type or a chunk thereof.
0x02	signalling message	one or more signalling messages or a fragment of a signalling message. The syntax and semantics of signalling messages are out of scope of the current memo.
0x03	repair symbol	The packet carries a single complete FEC repair symbol
0x04-0x1F	reserved	reserved for ISO use
0x20-0x3F	reserved	reserved for private use

Table 2: Data type and definition of data unit

3.1. MMTP Header Extension

```
 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             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   header_extension_value                 .... 
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: MMTP Header Extension

The function and length of each field in the MMTP header extension is as follows:

type: 16 bits

indicates the unique identification of the following header extension.

length: 16 bits

indicates the length of header_extension_value field in byte.

header_extension_value

provides the extension information. The format of this field is out of scope of this specification.

4. The MMTP payload

The MMTP payload is a generic payload format to packetize and carry media data such as ISOBMFF files, generic objects, and other information for consumption of a media service using the MMT protocol. The appropriate MMTP payload format shall be used to packetize ISOBMFF files, and generic objects. An MMTP payload may carry complete ISOBMFF files or fragments of ISOBMFF files, signalling messages, generic objects, repair symbols of AL-FEC schemes, etc. The type of the payload is indicated by the type field in the MMT protocol packet header. For each payload type, a single data unit for delivery as well as a type specific payload header are defined. For example, fragment of an ISOBMFF file (e.g. a data unit) is considered as a single data unit when MMTP payload carries ISOBMFF file fragments. The MMT protocol may aggregate multiple data units with the same data type into a single MMTP payload. It can also fragment a single data unit into multiple MMTP packets. The MMTP payload consists of a payload header and payload data. Some data types may allow for fragmentation and aggregation, in which case a single data unit is split into multiple fragments or a set of data units are delivered in a single MMTP packet. Each data unit may have its own data unit header depending on the type of the payload. For types that do not require a payload type specific header no payload type header is present and the payload data follows the MMTP header immediately. Some fields of the MMTP packet header are interpreted differently depending on the payload type. The semantics of these fields will be defined by the payload type in use.

4.1. The ISOBMFF Mode

The delivery of ISOBMFF files to MMT receivers using the MMT protocol requires a packetization and depacketization procedure to take place at the MMTP sender and MMTP receiver, respectively. The packetization procedure transforms an ISOBMFF file into a set of MMTP payloads that are then carried in MMTP packets. The MMTP payload format allows for fragmentation of the MMTP payload to enable the delivery of large payloads. It also allows for the aggregation of multiple MMTP payload data units into a single MMTP payload, to cater for smaller data units. At the receiving entity depacketization is performed to recover the original ISOBMFF file data. Several depacketization modes are defined to address the different requirements of the overlaying applications. It the payload type is 0x00, the ISOBMFF file is fragmented in a media aware way allowing the transport layer to identify the nature and priority of the fragment that is carried. A fragment of an ISOBMFF file may either be ISOBMFF file metadata, a Movie Fragment metadata, a data unit, or a non-timed media data item.

4.1.1. MMTP payload header for ISOBMFF mode

The payload type specific header is provided in Figure 3.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             length            |  FT   |T|f_i|A| frag_counter  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          sequence_number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           DU_length           |          DU_Header      ....
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         DU payload                      ....
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Structure of the MMTP payload header for the ISOBMFF mode

For payload that carries a data unit, the DU header is specified depending on the value of the T flag indicating wether the carried data is timed or non-timed media. For timed media (i.e. when the value of T is set to "1") the DU header fields are shown in Figure 4. For non-timed media (T is set to "0"), the DU header is defined as shown in Figure 4.

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                movie_fragment_sequence_number                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         sample_number                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            offset                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    priority   |  dep_counter  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: The DU header for a timed-media data unit

For non-timed media, the DU header fields are shown in 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            item_ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: The DU header for a non-timed media data unit

length: 16 bits indicates the length of payload excluding this field in byte.

Fragment Type (FT): 4 bits this field indicates the fragment type and its valid values are shown in Table 3.

Timed Flag (T): 1 bit this flag indicates if the fragment is from an ISOBMFF file that carries timed (value 1) or non-timed media (value 0).

Fragmentation Indicator (f_i) : 2 bits this field indicates the fragmentation indicator contains information about fragmentation of data unit in the payload. The four values are listed in Table 4. If the value is set to "00", the aggregation_flag can be presented.

FT	Description	Content
0	ISOBMFF metadata	contains the ftyp, mmpu, moov, and meta boxes as well as any other boxes that appear in between.
1	Movie fragment metadata	contains the moof box and the mdat box, excluding all media data inside the mdat box.
2	a data unit	contains a sample or sub-sample of timed media data or an item of non-timed media data.
3~15	Reserved for private use	reserved

Table 3: Data type and definition of data unit

fragmentation indicator	Description
00	Payload contains one or more complete data units.
01	Payload contains the first fragment of data unit
10	Payload contains a fragment of data unit that is neither the first nor the last part.
11	Payload contains the last fragment of data unit.

Table 4: Values for fragmentation indicator

The following flags are used to indicate the presence of various information carried in the MMTP payload. Multiple bits can be set simultaneously.

aggregation_flag (A: 1 bit)

when set to "1" indicates that more than 1 data unit is present in the payload, i.e. multiple data units are aggregated.

fragment_counter (frag_count: 8 bits)

this field specifies the number of payload containing fragments of same data unit succeeding this MMTP payload. This field shall be "0" if aggregation_flag is set to "1".

sequence_number (32 bits)

the sequence number of the ISOBMFF to which this fragment belongs.

DU_length (16 bits)

this field indicates the length of the data following this field. When aggregation_flag is set to "0", this field shall not be present. When aggregation_flag is set to "1", this field shall appear as many times as the number of the data units aggregated in the payload and preceding each aggregated data unit.

DU_Header

The header of the data unit, which depends on the FT field. A header is only defined for the media unit fragment type, with different semantics for timed and non-timed media as identified by the T flag.

movie_fragment_sequence_number (32 bits)

the sequence number of the movie fragment to which the media data of this data unit belongs. (see [isopart12] sub-clause 8.5.5)

sample_number (32 bits)

the sample number of the sample to which the media data of the data unit. (see [isopart12] sub-clause 8.8.8)

offset (32 bits)

offset of the media data of this data unit inside the referenced sample.

subsample_priority (priority: 8 bits)

provides the priority of the media data carried by this data unit compared to other media data of the same ISOBMFF file. The value of subsample_priority shall be between "0" and "255", with higher values indicating higher priority.

dependency_counter (dep_counter: 8 bits)

indicates the number of data units that depend on their media processing upon the media data in this data unit.

Item_ID (32 bits)

the identifier of the item that is carried as part of this data unit.

For the FT types "0" and "1", no additional DU header is defined.

4.2. Generic File Delivery Mode

MMTP also supports the transport of generic files and Assets and uses payload type "0x01" as defined in Table 3. An Asset consists of one or more files that are logically grouped and share some commonality for an application, e.g. Segments of a Dynamic Adaptive Streaming over HTTP (DASH) Representation, a sequence of ISOBMFF files, etc. In the generic file delivery (GFD) mode, an Asset is transported by using MMTP"s GFD payload type. Each file delivered using the GFD mode requires association of transport delivery information. This includes, but is not limited to information such as the transfer length. Each file delivered using the GFD mode may also have associated content specific parameters such as Name, Identification, and Location of file, media type, size of the file, encoding of the file or message digest of the file. In alignment with HTTP/1.1 protocol as defined in [RFC2616], each file within one generic Asset may have assigned any meta-information about the entity body, i.e. the delivered file. The details are also defined in Section 4.2.1.

4.2.1. GFD Information

In the GFD mode, each file gets assigned the following parameters:

- o the asset to which each object belongs to. Objects that belong to the same asset are considered as logically connected, e.g. all DASH segments of a Representation and also across Representations that extend over multiple DASH Periods and which carry pieces of the same content.

- o Each object is associated with a unique identifier within the scope of the packet_id.

- o each object is associated with a CodePoint. A CodePoint associates a specific object and object transport properties. Packets with the same TOI shall have the same CodePoint value. For more details see 0.

4.2.1.1. GFD Table

The GFD table provides a list of CodePoints as defined in Section 4.2.1.2. Each CodePoint gets dynamically assigned a CodePoint value. Table 5 shows the structure and semantics of the GFD table.

Element or Attribute Name	Use	Description
GFDTable CodePoint	1..N	The element carries a GFDTable defines all CodePoints in the MMTP session

Table 5: GFD Table

Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements: minOccurs..maxOccurs (N=unbounded) Elements are bold; attributes are non-bold and preceded with an @

4.2.1.2. CodePoints

A CodePoint value can be used to obtain following information:

- o the maximum transfer length of any object delivered with this CodePoint signalling

In addition, a CodePoint may include following information

- o the actual transfer length of the objects

- o any information that may be present in the entity-header as defined in [RFC2616] section 7.1.

- o A Content-Location-Template as defined in Section 4.2.1.3 using the TOI and packet_id parameter, if present. The TOI and packet_id may be used to generate the Content-Location for each TOI and packet_id. If such a template is present, the processing in Section 4.2.1.3 shall be used to generate the Content-Location and the value of the URI shall be treated as the Content-Location field in the entity-header.

- o Specific information on the content, for example how the content is packaged, etc.

Within one session, at most 256 CodePoints may be defined. The definition of CodePoints is dynamically setup in the MMTP Session Description. The CodePoint semantics are described in Table 6.

Element or Attribute Name	Use	Description
@value	M	defines the value of the CodePoint in the MMTP session as provided in the CodePoint value of the MMTP packet header containing the GFD payload. The value shall be between 1 and 255. The value 0 is reserved.
@fileDeliveryMode	M	specifies the file delivery mode according to Section 4.2.
@maximumTransferLength	M	specifies the maximum transfer length in bytes of any object delivered with this CodePoint in this MMTP session.
@constantTransferLength	OD default: 'false'	specifies if all objects delivered by this CodePoint have constant transfer length. If this attribute is set to TRUE, all objects shall have transfer length as specified in the @maximumTransferLength attribute.
@contentLocationTemplate	O	specifies a template to generate the Content-Location of the entity header.
EntityHeader	0..1	specifies a full entity header in the format as defined in [RFC2616] section 7.1. The entity header applies for all objects that are delivered with the value of this CodePoint.

Table 6: CodePoint Semantics

Legend: For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory. For elements:

minOccurs..maxOccurs (N=unbounded) Elements are bold; attributes are non-bold and preceded with an @

4.2.1.3. Content-Location Template

A CodePoint may include a @contentLocationTemplate attribute. The value of @contentLocationTemplate attribute may contain one or more of the identifiers listed in Table 7. In each URL, the identifiers from Table 7 shall be replaced by the substitution parameter defined in Table 7. Identifier matching is case-sensitive. If the URL contains unescaped $ symbols which do not enclose a valid identifier then the result of URL formation is undefined. The format of the identifier is also specified in Table 7. Each identifier may be suffixed, within the enclosing "$" characters following this prototype: %0[width]d The width parameter is an unsigned integer that provides the minimum number of characters to be printed. If the value to be printed is shorter than this number, the result shall be padded with zeros. The value is not truncated even if the result is larger. The @contentLocationTemplate shall be authored such that the application of the substitution process results in valid URIs. Strings outside identifiers shall only contain characters that are permitted within URLs according to [RFC3986].

$Identifier$	Substitution parameter	Format
$$	Is an escape sequence, i.e. "$$" is replaced with a single "$"	not applicable
$PacketID$	This identifier is substituted with the value of the packet_id of the associated MMT flow.	The format tag may be present.If no format tag is present, a default format tag with width=1 shall be used.
TOI	This identifier is substituted with the Object Identifier of the corresponding MMTP packet containing the GFDpayload.	The format tag may be present. If no format tag is present, a default format tag with width=1 shall be used.

Table 7: Identifiers for URL templates

4.2.1.4. File metadata

Files can be transported using the GFD mode of the MMT protocol. Furthermore, the GFD mode can also be used to transport entities where an entity is defined according to section 7 of [RFC2616]. An entity consists of meta-information in the form of entity-header fields and content in the form of an entity-body (the file), as described in section 7 of [RFC2616]. This enables that files may get assigned information by inband delivery in a dynamic fashion. For example, it enables the association of a Content-Location, the Content-Size, etc. The file delivery mode shall be signaled in the CodePoint.

Value $Identifier$	Description	Definition
1	The transport object is a file	in Section 4.2.1.4.1
2	The delivered object is an entity consisting of an entity-header and the file	in Section 4.2.1.4.2

Table 8: File Delivery Modes for GFD

4.2.1.4.1. Regular File

In case of the regular file, the object represents a file. If the CodePoint defined in the GFD table contains entity-header fields or entity-header fields can be generated, then all of these entity-header fields shall apply to the delivered file.

4.2.1.4.2. Regular Entity

In case of the regular entity, the object represents an entity as defined in section 7 of [RFC2616]. An entity consists of entity-header fields and an entity-body. If the CodePoint defined in the GFD table contains entity-header fields or entity-header fields can be generated, then all of these entity-header fields apply to the delivered file. If the entity-header field is present in both locations, then the entity header field in the entity-header delivered with the object overwrites the one in the CodePoint.

4.2.1.5. MMTP payload header for GFD mode

The GFD mode of MMTP delivers regular files. When delivering regular files, the object represents a file. If the CodePoint defined in the MMTP Session description contains entity-header fields or entity-header fields can be generated, then all of these entity-header fields shall apply to the delivered file. The payload packets sent using MMTP shall include a GFD payload header and a GFD payload as shown in Figure 6. In some special cases a MMTP sender may need to produce packets that do not contain any payload. This may be required, for example, to signal the end of a session.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C|L|B|      CP       |  RES    |                TOI            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            TOI                |        start_offset           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          start_offset                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Generic File Delivery payload                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

MMTP payload header for GFD mode

Figure 6

The GFD payload header as shown in Figure 6 and has a variable size. Bits designated as "padding" or "reserved" (r) MUST by set to 0 by MMTP sender s and ignored by receivers. Unless otherwise noted, numeric constants in this specification are in decimal form

C (1 bit)

indicates that this is the last packet for this session.

L (1 bit)

indicates that this is the last delivered packet for this object.

B (1 bit)

indicates that this packet contains the last byte of the object.

CodePoint (CP: 8 bits)

An opaque identifier that is passed to the packet payload decoder to convey information on the packet payload. The mapping between the CodePoint and the actual codec is defined on a per session basis and communicated out-of-band as part of the session description information.

RES (5 bits)

a reserved field that should be set to "0".

Transport Object Identifier (TOI: 32 bits)

The object identifier should be set to a unique identifier of the generic object that is being delivered. The mapping between the object identifier and the object information (such as URL and MIME type) may be done explicitly or implicitly. For example a sequence of DASH segments may use the segment index as the object identifier and a numerical representation identifier as the packet_id. This mapping may also be performed using a signalling message

start_offset (48 bits)

the location of the current payload data in the object.

5. Protocol Operation

In this section, we describe the behavior of an MMTP receiver and of an MMTP sender when operating the MMTP protocol using different payload types.

5.1. General Operation

An MMTP session consists of one MMTP transport flow. MMTP transport flow is defined as all packet flows that are delivered to the same destination and which may originate from multiple MMTP senders. In the case of IP, destination is the IP address and port number. A single Package may be delivered over one or multiple MMTP transport flows. A single MMTP transport flow may deliver data from multiple Packages. An MMTP transport flow may carry multiple Assets. Each Asset is associated with a unique packet_id within the scope of the MMTP session. MMTP provides a streaming-optimized mode (the ISOBMFF mode) and a file download mode (the GFD mode). The Asset is delivered as a set of related objects denoted as an object flow. Object may either be an ISOBMFF file, file or signalling message. Each object flow shall either be carried in ISOBMFF mode or GFD mode, however, the delivery of one Package may be performed using a mix of the 2(two) modes, i.e. some Assets may be delivered using the ISOBMFF mode and others using the GFD mode. The MMTP packet sub-flow is the subset of the packets of an MMTP packet flow that share the same packet_id. The object flow is transported as an MMTP packet sub-flow. The ISOBMFF mode supports the packetized streaming of an ISOBMFF file. The GFD mode supports flexible file delivery of any type of file or sequence of files. MMTP is suitable for unicast as well as multicast media distribution. To ensure scalability in multicast/ broadcast environments, MMTP relies mainly on FEC instead of retransmissions for coping with packet error. Before joining the MMTP session, the MMTP receiver should obtain sufficient information to enable reception of the delivered data. This minimum required information is specified in Section 6. MMTP requires MMTP receivers to be able to uniquely identify and de-multiplex MMTP packets that belong to a specific object flow. In addition, MMT receivers are required to be able to identify packets carrying AL-FEC repair packets by interpreting the type field of the MMTP packet header. The MMTP receiver shall be able to simultaneously receive, de-multiplex, and reconstruct the data delivered by MMTP packets of different types and from different object flows. A single MMTP packet shall carry exactly one MMTP payload.

5.2. Delivery ISOBMFF objects

The ISOBMFF mode is used to transport ISOBMFF files sent by a sending entity to a receiving entity.

5.2.1. MMTP sender operation

5.2.1.1. Timed Media Data

The packetization of an ISOBMFF file that contains timed media may be performed in a ISOBMFF file format aware mode or ISOBMFF file format agnostic mode. In the media format agnostic mode, the ISOBMFF file is packetized into data units of equal size (except for the last data unit, of which the size may differ) or predefined size according to the size of MTU of the underlying delivery network by using GFD mode as specified in Section 4.2. It means that the packetization of the ISOBMFF file format agnostic mode only consider the size of data to be carried in the packet. The type field of MMTP packet header specified in Section 4.1 is set to "0x00" to indicate that the packetization is format agnostic mode. In the format agnostic mode the packetization procedure takes into account the boundaries of different types of data in ISOBMFF file to generate packets by using ISOBMFF mode as specified in Section 4.1. The resulting packets shall carry delivery data units of either ISOBMFF file metadata, movie fragment metadata, or a data unit. The resulting packets shall not carry more than two different types of delivery data units. The delivery data unit of ISOBMFF file metadata consists of the "ftyp" box, the "mmpu" box, the "moov" box, and any other boxes that are applied to the whole ISOBMFF file. The FT field of the MMTP payload carrying a delivery data unit of the ISOBMFF file metadata is set to "0x00". The delivery data unit of movie fragment metadata consists of the "moof" box and the "mdat" box header (excluding any media data). The FT field of the MMTP payload carrying a delivery data unit of movie fragment metadata is set to "0x01". The media data, data units in "mdat" box of the ISOBMFF file, is then split into multiple delivery data units in a format aware way. This may for example be performed with the help of the MMT hint track. The FT field of the MMTP payload carrying a delivery data unit is set to "0x02". Each data unit is prepended with a data unit header, which has the syntax and semantics as defined in section Section 4.1.1. It is followed by the media data of the data unit. This procedure is described by Figure 7.

```
+------+ +------+ +------+   +------+  +--------+-------------------------+
| ftyp | | mmpu | | moov |   | moof |  |mdat_hdr|          mdat           |
+------+ +------+ +------+   +------+  +--------+-------------------------+
|                        |   |                  |   ...    |    |
|                        |   |                  |          |    |
|                        |   |                  |          |    |
+------------------------+   +------------------+          +----+
|  ISOBMFF metadata      |   | Fragment metadata|   ...    | DU |
+------------------------+   +------------------+          +----+
```

Payload generation for timed media

Figure 7

5.2.1.2. Non-Timed Media Data

The packetization of non-timed media data may also be performed in two different modes. In the ISOBMFF file format agnostic mode, the ISOBMFF file is packetized into delivery data units of equal size (except for the last data unit, of which the size may differ) or or predefined size according to the size of MTU of the underlying delivery network by using GFD mode as specified in Section 4.2. The type field of MMTP packet header specified in Figure 1 is set to "0x00" to indicate that the packetization is format agnostic mode. In the format agnostic mode, the ISOBMFF file shall be packetized into the packet containing delivery data units of either ISOBMFF file metadata or data unit by using ISOBMFF mode as defined in Section 4.1. The delivery data unit of the ISOBMFF file metadata contains the "ftyp" box, the "moov" box, and the "meta" box and any other boxes that are applied to the whole ISOBMFF file. The FT field of the MMTP payload carrying a delivery data unit of the ISOBMFF file metadata is set to "0x01". Each delivery data unit contains a single item of the non-timed media. The FT field of the MMTP payload carrying a delivery data unit is set to "0x02". Each item of the non-timed data is then used to build a data unit. Each data unit consists of a data unit header and the item's data. The data unit header is defined in Section 4.1.1.

```
+-----+ +-----+ +-----+ +-----+   +---------+    +-----------------------+
|ftyp| |mmpu| |moov| |meta|   | item #1 |    |      item #2          |
+-----+ +-----+ +-----+ +-----+   +---------+    +-----------------------+
|                         |   |         |    |                       |
|                         |   |         |    |                       |
|                         |   |         |    |                       |
+-------------------------+   +---------+    +-----------------------+
|   ISOBMFF metadata      |   |  DU     |    |          DU           |
+-------------------------+   +---------+    +-----------------------+
```

Payload generation for non-timed media

Figure 8

5.2.2. MMTP receiver operation

The depacketization procedure is performed at an MMTP receiver to rebuild the transmitted ISOBMFF file. The depacketization procedure may operate in one of the following modes, depending on the application needs:

ISOBMFF mode:

> in the ISOBMFF mode, the depacketizer reconstructs the full ISOBMFF file before forwarding it to the application. This mode is appropriate for non-time critical delivery, i.e. the ISOBMFF file's presentation time as indicated by the presentation information document is sufficiently behind its delivery time.

Fragment mode:

> in the Fragment mode, the depacketizer reconstructs a complete fragment including the fragment metadata and the "mdat" box with media samples before forwarding it to the application. This mode does not apply to non-timed media. This mode is suitable for delay-sensitive applications where the delivery time budget is limited but is large enough to recover a complete fragment.

Media unit mode:

> in the media unit mode, the depacketizer extracts and forwards media units as fast as possible to the application. This mode is

applicable for very low delay media applications. In this mode, the recovery of the ISOBMFF file is not required. The processing of the fragment media data is not required but may be performed to resynchronize. This mode tolerates out of order delivery of the fragment metadata data units, which may be generated after the media units are generated. This mode applies to both timed and non-timed media. Using the data unit sequence numbers, it is relatively easy for the receiver to detect missing packets and apply any error correction procedures such as ARQ to recover the missing packets. The payload type may be used by the MMTP sender to determine the importance of the payload for the application and to apply appropriate error resilience measures.

5.3. Delivering Generic Objects

The files delivered using the GFD mode may have to be provided to an application, for example Presentation Information documents or a Media Presentation Description as defined in ISO/IEC 23009-1 may refer to the files delivered using MMTP as GFD objects. The file shall be referenced through the URI provided or derived from Content-Location, either provided in-band as part of an entity header or as part of a GFDT. In certain cases, the files have an availability start time in the application. In this case the MMTP session shall deliver the files such that the last packet of the object is delivered such that it is available latest at the receiver at the availability start time as announced in the application. Applications delivered through the GFD mode may impose additional and stricter requirements on the sending of the files within a MMTP session.

5.3.1. MMTP sender operation

If more than one object is to be delivered using the GFD mode, then the MMTP sender shall use different TOI fields. In this case each object shall be identified by a unique TOI scoped by the packet_id, and the MMTP sender shall use that TOI value for all packets pertaining to the same object. The mapping between TOIs and files carried in a session is either provided in-band or in a GFDT. The GFD payload header as defined in Section 4.2.1.5 shall be used. The GFD payload header contains a CodePoint field that shall be used to communicate to a MMTP receiver the settings for information that is established for the current MMTP session and may even vary during a MMTP session. The mapping between settings and Codepoint values is communicated in the a GFDT as described in Section 4.2.1.1. Let T > 0 be the Transfer-Length of any object in bytes. The data carried in the payload of a packet consists of a consecutive portion of the object. Then for any arbitrary X and any arbitrary Y > 0 as long as

X + Y is at most T, an MMTP packet may be generated. In this case the followings shall hold:

1. The data carried in the payload of a packet shall consist of a consecutive portion of the object starting from the beginning of byte X through the beginning of byte X + Y.

2. The start_offset field in the GFD payload header shall be set to X and the payload data shall be added into the packet to send.

3. If X + Y is identical to T,

    * the payload header flag B shall be set to "1".

    * else

    * the payload header flag B shall be set to "0".

The following procedure is recommended for a MMTP sender to deliver an object to generate packets containing start_offset and corresponding payload data.

1. Set the byte offset counter X to "0"

2. For the next packets to be delivered set the length in bytes of a payload to a value Y, which is

    * reasonable for a packet payload (e.g., ensure that the total packet size does not exceed the MTU), and

    * such that the sum of X and Y is at most T, and

    * such that it is suitable for the payload data included in the packet

3. Generate a packet according to the rules a to c from above

4. If X + Y is equal to T,

    * set the payload header flag B to "1"

    * else

    * set the payload header flag B to "0"

    * increment X = X + Y

    * goto 2

The order of packet delivery is arbitrary, but in the absence of other constraints delivery with increasing start_offset number is recommended. Note that the transfer length may be unknown prior to sending earlier pieces of the data. In this case, T may be determined later. However, this does not affect the sending process above. Additional packets may be sent following the rules in 1 to 3 from above. In this case the B flag shall only be set for the payload that contains the last portion of the object.

5.3.2. GFD Payload

The bytes of the object are referenced such that byte 0 is the beginning of the object and byte T-1 is the last byte of the object with T is the transfer length (in bytes) of the object. The data carried in the payload of an MMTP packet shall consist of a consecutive portion of the object starting from the beginning of byte X and ending at the beginning of byte X + Y where

1. X is the value of start_offset field in the GFD payload header

2. Y is the length of the payload in bytes

Note that Y is not carried in the packet, but framing shall be provided by the underlying transport protocol.

5.3.3. GFD Table Delivery

When GFD mode is used, the GFD table (GFDT) shall be provided. A file that is delivered using the GFD mode, but not described in the GFD table is not considered a 'file' belonging to the MMTP session. Any object received with an unmapped CodePoint should be ignored by a MMTP receiver. Other ways of delivery the GFD table may possible, but out of scope of this specification.

5.3.4. MMTP receiver operation

The GFDT may contain one or multiple CodePoints identified by different CodePoint values. Upon receipt of each GFD payload, the receiver proceeds with the following steps in the order listed.

1. The MMTP receiver shall parse the GFD payload header and verify that it is a valid header. If it is not valid, then the GFD payload shall be discarded without further processing.

2. The MMTP receiver shall parse the CodePoint value and verify that the GFDT contains a matching CodePoint. If it is not valid, then the GFD payload shall be discarded without further processing.

3. The MMTP receiver should process the remainder of the payload, including interpreting the other payload header fields appropriately, and using the source_offset and the payload data to reconstruct the corresponding object as follows:

    1. The MMT receiving can determine from which object a received GFD payload was generated by using the GFDT., and by the TOI carried in the payload header.

    2. Upon receipt of the first GFD payload for an object, the MMTP receiver uses the Maximum Transfer Length received as part of the GFDT to determine the maximum length T’ of the object.

    3. The MMTP receiver allocates space for the T’ bytes that the object may require.

    4. The MMTP receiver also computes the length of the payload, Y, by subtracting the payload header length from the total length of the received payload.

    5. The MMTP receiver allocates a Boolean array RECEIVED[0..T’-1] with all T entries initialized to false to track received object symbols. The MMTP receiver keeps receiving payloads for the object block as long as there is at least one entry in RECEIVED still set to false or until the application decides to give up on this object.

    6. For each received GFD payload for the object (including the first payload), the steps to be taken to help recover the object are as follows:

    7. Let X be the value of the source_offset field in the GFD payload header of the MMTP packet. and let Y be the length of the payload, Y, computed by subtracting the MMTP packet and GFD payload header lengths from the total length of the received packet.

    8. The MMTP receiver copies the data into the appropriate place within the space reserved for the object and sets RECEIVED[X ... X+Y-1] = true.

    9. If all T entries of RECEIVED are true, then the receiver has recovered the entire object.

    10. Once the MMTP receiver receives a GFD payload with the B flag set to 1, it can determine the transfer length T of the

object as X+Y of the corresponding GFD payload and adjust the boolean array RECEIVED[0..T'-1] to RECEIVED[0..T-1].

6. Session Description information

The MMTP session description information may be delivered to receivers in different ways to accommodate different deployment environments. Before a receiver is able to join an MMTP session, the receiver needs to obtain the following information:

The destination information. In an IP environment, the destination IP address and port number.

An indication that the session is an MMTP session

The version number of the MMT protocol used in the MMTP session

Additionally, the MMTP session description information should contain the following information:

The start and end time of the MMTP session.

7. Congestion Control

All transport protocols used on the Internet are required to address congestion control. MMTP provides for means to the sender to adjust its sending rate to the available bandwidth. Feedback mechanisms from the client, sent as part of MMT signaling, give the sender the necessary information to estimate the available bandwidth. A description file of the content that is being streamed is also available at the sender to assist with the selection of alternative representations and in stream thinning through selection of an appropriate operation point. The MMTP sender SHALL make use of this available information to timely react to congestion.

8. IANA Considerations

This internet draft includes no request to IANA.

9. Security Considerations

Lower layer protocols may eventually provide all the security services that may be desired for applications of MMTP, including authentication, integrity, and confidentiality. These services have been specified for IP in [RFC4301].

10. References

10.1. Normative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[isopart12] ISO/IEC, "Information technology High efficiency coding and media delivery in heterogeneous environments Part 12: File Format", 2008, <http://www.mpeg.org/>.

[mmt] ISO/IEC, "Information technology High efficiency coding and media delivery in heterogeneous environments Part 1: MPEG media transport (MMT)", 2014, <http://www.mpeg.org/>.

10.2. Informative References

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

Author’s Address

Imed Bouazizi
Samsung Research America
Richardson, TX
US

Phone: +1 972 763 7023
Email: i.bouazizi@samsung.com