

Second edition  
2017-08

Corrected version  
2017-10

---

---

**Information technology — High  
efficiency coding and media delivery  
in heterogeneous environments —**

**Part 13:  
MPEG media transport  
implementation guidelines**

*Technologies de l'information — Codage à haute efficacité et livraison  
des médias dans des environnements hétérogènes —*

*Partie 13: Lignes directrices de mise en oeuvre du transport des  
médias MPEG*



Reference number  
ISO/IEC TR 23008-13:2017(E)  
**Documents**  
AVA Document Sharing Hub  
© ISO/IEC 2017



**COPYRIGHT PROTECTED DOCUMENT**

© ISO/IEC 2017, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
Ch. de Blandonnet 8 • CP 401  
CH-1214 Vernier, Geneva, Switzerland  
Tel. +41 22 749 01 11  
Fax +41 22 749 09 47  
copyright@iso.org  
www.iso.org

# Contents

Page

|                                                                                 |           |
|---------------------------------------------------------------------------------|-----------|
| <b>Foreword</b>                                                                 | <b>v</b>  |
| <b>Introduction</b>                                                             | <b>vi</b> |
| <b>1 Scope</b>                                                                  | <b>1</b>  |
| <b>2 Normative references</b>                                                   | <b>1</b>  |
| <b>3 Terms, definitions, symbols and abbreviated terms</b>                      | <b>1</b>  |
| <b>4 Overview</b>                                                               | <b>1</b>  |
| 4.1 System overview                                                             | 1         |
| 4.2 Normative parts                                                             | 2         |
| <b>5 MMT function deployments</b>                                               | <b>3</b>  |
| 5.1 General                                                                     | 3         |
| 5.2 Object reconstruction                                                       | 3         |
| 5.2.1 General                                                                   | 3         |
| 5.2.2 Recovery in MPU mode                                                      | 4         |
| 5.2.3 Recovery in GFD mode                                                      | 5         |
| 5.3 Default assets                                                              | 6         |
| 5.4 Low-delay live streaming                                                    | 6         |
| 5.5 Parallel processing in MMT sending and receiving entities                   | 9         |
| 5.5.1 Processing in MMT sending entity                                          | 9         |
| 5.5.2 Processing in MMT receiving entity                                        | 11        |
| 5.6 MPU streaming for live services                                             | 13        |
| 5.6.1 MPU packetization                                                         | 13        |
| 5.6.2 Sending of MPU and signalling message                                     | 16        |
| 5.7 Fast MMT session acquisition                                                | 17        |
| 5.8 Referencing and processing non-timed data                                   | 18        |
| 5.8.1 General                                                                   | 18        |
| 5.8.2 Resource grouping and referencing                                         | 19        |
| 5.8.3 Receiver handling                                                         | 19        |
| 5.9 Media adaptation for quality control in MMTP                                | 19        |
| 5.9.1 General                                                                   | 19        |
| 5.9.2 Parameters for media adaptation                                           | 19        |
| 5.9.3 Adaptation operation of MMT entity                                        | 20        |
| 5.10 Hybrid delivery in MMT                                                     | 20        |
| 5.10.1 General                                                                  | 20        |
| 5.10.2 Classification of hybrid delivery                                        | 20        |
| 5.10.3 Technical elements for hybrid delivery                                   | 21        |
| 5.11 Example of detailed implementation of MMT                                  | 22        |
| 5.11.1 Use case: Combination of MMT and MPEG-2 TS for synchronized presentation | 22        |
| 5.11.2 Use case: True real-time video streaming over lossy channel              | 22        |
| 5.12 HRBM signalling for hybrid delivery                                        | 23        |
| 5.12.1 Hybrid delivery from the single MMT sending entity                       | 23        |
| 5.12.2 Hybrid delivery from the multiple MMT sending entities                   | 25        |
| 5.13 Error resilience in MMT protocol                                           | 27        |
| 5.14 Delay-constrained ARQ                                                      | 28        |
| 5.14.1 General                                                                  | 28        |
| 5.14.2 Delivery-time constrained ARQ                                            | 28        |
| 5.14.3 Arrival-deadline constrained ARQ                                         | 29        |
| 5.15 Application layer forward error correction (AL-FEC)                        | 31        |
| 5.15.1 FEC decoding method for ssbg_mode2                                       | 31        |
| 5.15.2 Usage of two-stage FEC coding structure                                  | 35        |
| 5.15.3 MPU mapping to source packet block                                       | 37        |
| 5.15.4 FEC for hybrid service                                                   | 38        |
| 5.16 Delivery of encrypted MPUs                                                 | 40        |

|          |                                                                            |           |
|----------|----------------------------------------------------------------------------|-----------|
| 5.17     | HRBM message updating .....                                                | 41        |
| 5.17.1   | General .....                                                              | 41        |
| 5.17.2   | HRBM message sending schedule .....                                        | 41        |
| 5.17.3   | Use case .....                                                             | 42        |
| 5.18     | MMTP packet with padded data .....                                         | 42        |
| <b>6</b> | <b>Use cases for MMT deployment .....</b>                                  | <b>44</b> |
| 6.1      | General .....                                                              | 44        |
| 6.2      | Delivery of DASH Presentations using MMT .....                             | 44        |
| 6.2.1    | General .....                                                              | 44        |
| 6.2.2    | Delivery of the MPD .....                                                  | 44        |
| 6.2.3    | Delivery of the data segments .....                                        | 44        |
| 6.3      | Client operation for DASH service delivered through MMT Protocol .....     | 45        |
| 6.3.1    | Delivery of MPD with MMTP .....                                            | 45        |
| 6.3.2    | Delivery and consumption of DASH Segments with MMTP .....                  | 45        |
| 6.4      | Hybrid of MMT and DASH over heterogeneous network .....                    | 47        |
| 6.5      | MMT caching for effective bandwidth utilization .....                      | 49        |
| 6.5.1    | Overview of MMT caching middlebox architecture .....                       | 49        |
| 6.5.2    | Content-based caching of MMT media .....                                   | 49        |
| 6.5.3    | MPU sync protocol between server and caching middlebox .....               | 52        |
| 6.5.4    | MMT cache manifest .....                                                   | 56        |
| 6.6      | Usage of ADC signalling message .....                                      | 58        |
| 6.6.1    | General .....                                                              | 58        |
| 6.6.2    | Operation in MMT sending entity .....                                      | 58        |
| 6.6.3    | Operation in MANE router .....                                             | 58        |
| 6.6.4    | Example operation in MMT receiving entities .....                          | 58        |
| 6.6.5    | QoE multiplexing gain and bottleneck coordination .....                    | 58        |
| 6.7      | MMT deployment in Japanese broadcasting systems .....                      | 62        |
| 6.7.1    | General .....                                                              | 62        |
| 6.7.2    | Broadcasting systems using MMT .....                                       | 62        |
| 6.7.3    | Media transport protocol .....                                             | 64        |
| 6.7.4    | Signalling information .....                                               | 69        |
| 6.7.5    | Start-up procedure of broadcasting service .....                           | 78        |
| 6.8      | Conversion of MMTP stream to MPEG-2 TS .....                               | 80        |
| 6.8.1    | Overview of conversion operation .....                                     | 80        |
| 6.8.2    | Restrictions to MMTP packets .....                                         | 80        |
| 6.8.3    | Calculation of PTS, DTS .....                                              | 80        |
| 6.8.4    | Restriction related to MPEG-2 T-STD .....                                  | 81        |
| 6.8.5    | Packet field conversion rule .....                                         | 81        |
| 6.8.6    | PSI conversion rules .....                                                 | 82        |
| 6.9      | MMT service provisioning at conventional broadcast environment .....       | 84        |
| 6.10     | Usage of multimedia configuration for interface switching management ..... | 86        |
|          | <b>Bibliography .....</b>                                                  | <b>87</b> |

## Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see [www.iso.org/directives](http://www.iso.org/directives)).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see [www.iso.org/patents](http://www.iso.org/patents)).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: [www.iso.org/iso/foreword.html](http://www.iso.org/iso/foreword.html).

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC/TR 23008-13:2015), which has been technically revised.

The main changes compared to the previous edition are as follows:

- areas on MMT deployment case in Japan and conversion between MMTP and MPEG-2 TS stream have been improved by adding new guidelines.

A list of all parts in the ISO/IEC 23008 series can be found on the ISO website.

This corrected version of ISO/IEC TR 23008-13:2017 incorporates the following corrections:

- headers have been corrected and now read “ISO/IEC TR” instead of “ISO/TR”.

## **Introduction**

This document provides guidelines for implementation and deployment of multimedia systems based on the ISO/IEC 23008 series. These guidelines include the following:

- guidelines on usage of MMT functions;
- guidelines on deployment use cases designed based on ISO/IEC 23008-1.

# Information technology — High efficiency coding and media delivery in heterogeneous environments —

## Part 13: MPEG media transport implementation guidelines

### 1 Scope

This document provides technical guidelines for implementing and deploying systems based on ISO/IEC 23008-1.

### 2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 23008-1:2014, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 1: MPEG media transport (MMT)*

### 3 Terms, definitions, symbols and abbreviated terms

For the purposes of this document, the terms, definitions, symbols and abbreviated terms given in ISO/IEC 23008-1 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

### 4 Overview

#### 4.1 System overview

This clause describes the exemplary but typical system overview of MPEG Media Transport (MMT) as shown in [Figure 1](#).

The media origin provides A/V media or generic files to MMT sending entity in the form of Packages or Assets which are defined in ISO/IEC 23008-1. A Package is comprised of Assets, Presentation Information and Transparent Characteristics, etc. Physically, an Asset is a group of MPUs or generic files.

The MMT sending entity fragments MPU/generic files and generates MMTP packets to deliver A/V media data itself. Concurrently, it also generates signalling message for the successful delivery and presentation of A/V media included on that MMTP packet flow.

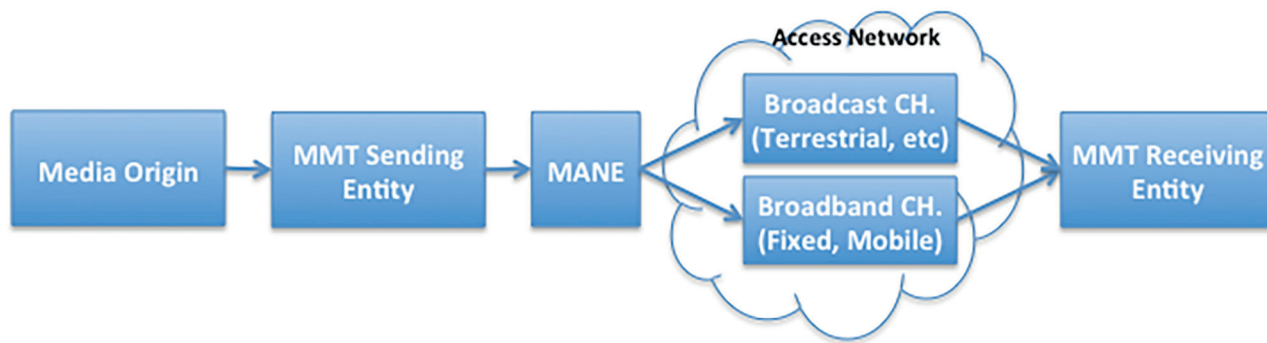


**Figure 1 — Example of MMT-based media distribution chain**

The MMT Aware Network Element (MANE) may be any network element, such as media caches and routers, that are aware of MMTP and has augmented functions for its own purposes to utilize tools from MMT.

Then, MMTP packets can be transmitted through either or both of broadcast channel and broadband channel at its own environment and scenarios.

The CI information provides presentation information, such as the location of media objects, as well as the timing and relation of the media objects that MMT receiving entity has to follow. This information is provided by MMT sending entity and also pushes related MMTP packet flow to the MMT receiving entity. It means It fully controls the media streaming session, i.e. it manages the on-time delivery, playback and temporal/spatial presentation information of the media.

## 4.2 Normative parts

ISO/IEC 23008-1 specifies a set of tools to enable advanced media transport and delivery services. [Figure 2](#) depicts the end-to-end architecture and illustrates the different functional tools and their relationships. Moreover, it shows interfaces between existing protocols and standards defined by ISO/IEC 23008-1 and those defined in other specifications. The tools spread over three different functional areas, Media Processing Unit (MPU) format, delivery and signalling, defined in ISO/IEC 23008-1 as follows.

- The Media Processing Unit (MPU) defines the logical structure of media content format of the data units to be processed by an MMT entity and their instantiation with ISO Base Media File Format as specified in ISO/IEC 14496-12.
- The delivery function defines an application layer transport protocol and a payload format. The MMTP transport protocol provides enhanced features for delivery of multimedia data, e.g. multiplexing and support of mixed use of streaming and download delivery in a single packet flow. The payload format is defined to enable the carriage of encoded media data which is agnostic to media types and encoding methods.
- The signalling function defines formats of signalling messages to manage delivery and consumption of media data.

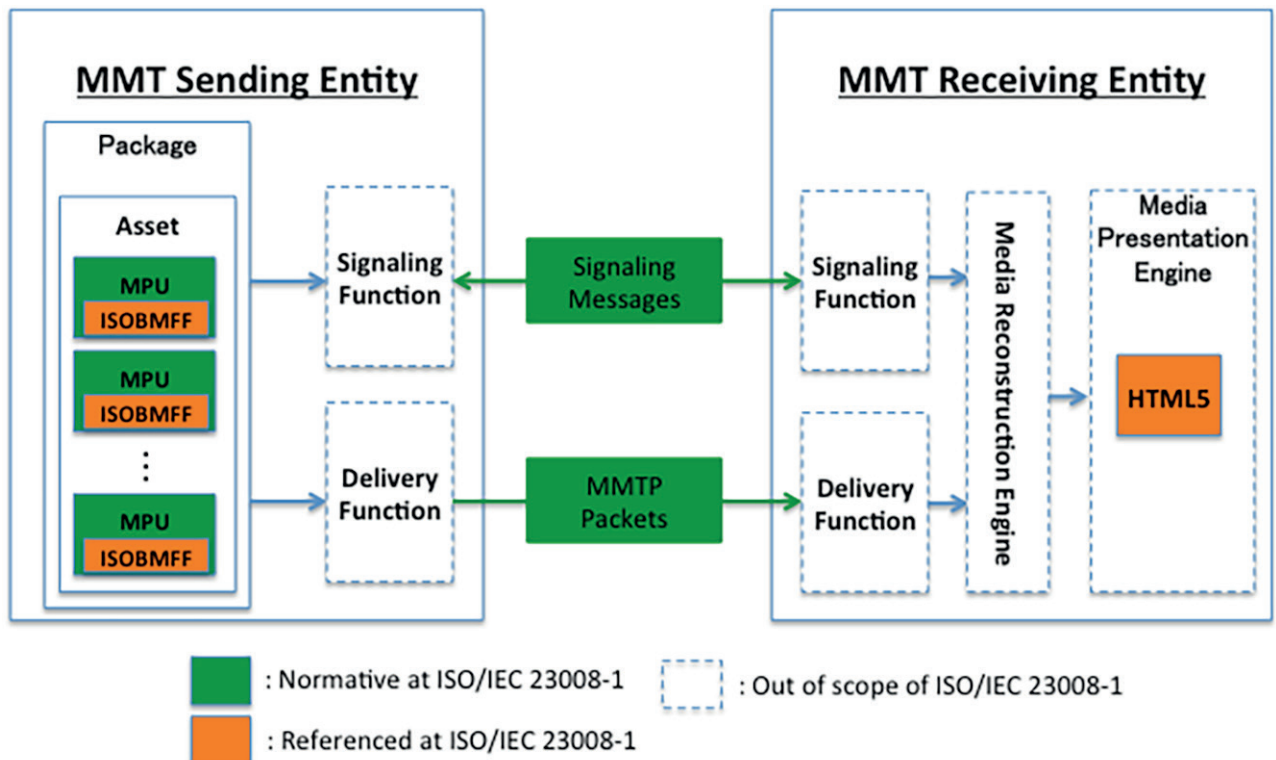


Figure 2 — MMT functions deployment

Other aspects, such as client implementations for media reconstruction and presentation itself, are not defined as normative parts of the ISO/IEC 23008-1.

## 5 MMT function deployments

### 5.1 General

This clause gives implementation guidelines on general MMT deployment based on basic functionalities provided by MMT standard itself. This subclause intends, in particular, to guide implementers to make best use of the specification for the basic topics such as, but not limited to the following:

- low delay media consumption;
- media adaptation;
- hybrid delivery;
- error recovery.

### 5.2 Object reconstruction

#### 5.2.1 General

MMTP is designed to deliver object flows that may be multiplexed together in the same MMTP flow. The objects of an object flow are usually related to each other, meaning that the application is likely to consume all objects of an object flow, if the flow or one of its objects is of interest to that application.

Depending on the delivery mode, the recovery of the object may differ. The GFD mode usually requires that the full object is recovered prior to its delivery to the application. However, the application may request that correctly received contiguous byte ranges of the object are forwarded to the application.

The MPU mode is used to deliver MPUs and usually operates on movie fragments. Alternatively, the application may request that each received MFU is forwarded to the application without additional delay. It may also require that the complete MPU be reconstructed prior to forwarding it to the application.

### 5.2.2 Recovery in MPU mode

When operating in the MPU mode, the object flow consists of MPUs of the same asset. Each MPU is a single object in the object flow and shares the same packet\_id as all MPUs of the same asset.

The MMT receiving entity performs the following steps.

- a) Receive MMTP packet.
- b) Check if packet\_id is equal to the packet\_id of the object flow of interest, discard packet and go to step a) if it does not belong to an object flow of interest.
- c) Assert that type of the MMTP packet is MPU.
- d) If fragmentation flags are set (different than "00")
  - 1) if fragmentation flag is equal to "11", attempt to recover packet and if successful go to step f), or
  - 2) add packet to the list of packet fragments based on the MMTP sequence number and go to step a).
- e) If Aggregation flag A is set, extract all aggregated data units and proceed to step g) for each extracted data unit.
- f) If object map with the same MPU\_sequence\_number does not exist, create a new object map for the MPU with that sequence number.
- g) Check fragment type (FT) of the MPU payload header.
  - 1) If FT is MPU metadata
    - i) Check if MPU metadata is already received.
      - I) If yes, discard the MPU metadata as being a duplicate, or
      - II) insert MPU metadata at the beginning of the object map
        - optionally, forward MPU metadata to application.
    - ii) Go to step a).
  - 2) If FT is Fragment metadata
    - i) Check if movie fragment with the same movie\_fragment\_sequence\_number already exists.
      - I) If no, create a placeholder for the movie fragment in the object map, or
      - II) check if Fragment metadata has already been received.
        - If yes, discard fragment metadata as being a duplicate,
        - otherwise, insert fragment metadata at the beginning of the fragment placeholder.
      - III) Go to step a).
  - 3) If FT is MFU
    - i) If fragment placeholder with sequence number movie\_fragment\_sequence\_number does not exist in the object map of the MPU with sequence number MPU\_sequence\_number, then create movie fragment placeholder in the object map of the MPU.

- ii) If timed metadata flag is set
  - I) insert payload in the fragment placeholder in the correct order based on the sample\_number and offset values;
  - II) check if movie fragment is complete, and
    - if yes, forward fragment to the application.
  - III) Go to step a).
- iii) If timed metadata flag is not set
  - I) insert payload in the item in the object map based on the item item\_ID,
  - II) recover item information from MPU metadata for the recovered item and forward the item to the application, and
  - III) go to step a).

The sender may send the movie fragment out of order, i.e. sending the movie fragment header after sending all the media units that are contained in that movie fragment. At the receiver side, step g.3.i ensures that the movie fragment is recovered appropriately by reordering the received data using the MPU\_sequence\_number and the movie\_fragment\_sequence\_number. This is necessary if the receiver is operating in the Fragment mode or MPU mode, where only complete movie fragments or complete MPUs are forwarded to the application. When operating in the very low delay mode, the receiver will forward every single MFU to the application. In this case, it has to make sure that the content supports this operation, so that MFUs will be self-describing and self-contained. In particular, the receiver should be able to recover the presentation timestamp of that MFU payload using the sample number, fragment\_sequence\_number and MPU\_sequence\_number,

For fragments and items that cannot be recovered correctly by the time the fixed end to end delivery delay passes, error concealment is performed on the movie fragment or the partially recovered item.

### 5.2.3 Recovery in GFD mode

When operating in the GFD mode, the object flow consists of a set of related files. The files of the same object flow share the same packet\_id. The application forwards each recovered file or contiguous byte range of a file to the application. The receiver creates an object map to recover each file separately.

The operation of the MMTP receiver is as follows.

- a) Receive MMTP packet.
- b) Check if packet\_id is equal to the packet\_id of the object flow of interest, discard packet and go to step a) if it does not belong to an object flow of interest
- c) Assert that type of the MMTP packet is GFD.
- d) If object map with same TOI does not exist, create new object map for the file with that TOI.
- e) Insert payload in the correct place in the object map using the start\_offset information.
- f) It is recommended that chunks of contiguous byte ranges that lie between two MMTP packets with the RAP flag R set to 1 be forwarded to the application. Applications may choose to forward sufficiently large contiguous byte ranges whenever they are recovered correctly.
- g) If complete TOI is recovered
  - 1) extract metadata from the transport object or from the GFD table, and
  - 2) forward file to the application.

### 5.3 Default assets

In order to cater for basic receivers with limited processing capabilities and also to facilitate fast channel tune-in, an alternative and simple way for service consumption has been devised that can function without the need for more advanced and highly demanding presentation information solutions (such as HTML 5). It is of course not possible to achieve the same level of service complexity and richness with the basic solution, but it enables receivers to quickly tune in to the channel and, if needed, to completely avoid processing the complex presentation information.

MMT provides the tools to identify default service components and to enable the receiver to consume them in a synchronized manner by processing the MMT signalling information. Default service components are usually the main video stream together with the default audio stream.

An MMT receiver that wants to achieve fast tune in or wants to bypass processing the presentation information checks the MP table for the MMT package of interest and identifies the assets of that MMT package that are marked as default assets. It then starts receiving and reconstructing the default assets by first locating the asset using the `MMT_general_location_info` and looking for the MPU metadata information as a starting point for the reconstruction. The MPU header is necessary as it delivers the information about the used media codecs and any applied encryption.

Each MPU of a default asset provides its presentation time, which can be used for synchronized playback of the media components. This is done using the MPU timestamp descriptor, which assigns an NTP playback timestamp for the MPU with sequence number `mpu_sequence_number`.

If the MMT receiver decides later on to consume the presentation information, it might stop relying on timing information provided in the MP table and use the presentation information instead.

### 5.4 Low-delay live streaming

MMT streaming is based on the MPU concept, which in turn is an ISO-based media file format (ISOBMFF) with certain restrictions. However, the usage of the ISOBMFF may give the impression of high end-to-end delay, which may seem not suitable for live broadcast. This however, is not true. This subclause shows how low-delay live streaming may be performed using MMT.

Live streaming requires real-time media encoding and transmission of the encoded media to a set of receivers. In such scenarios, end-to-end delay has a significant impact on the perceived quality by the end user.

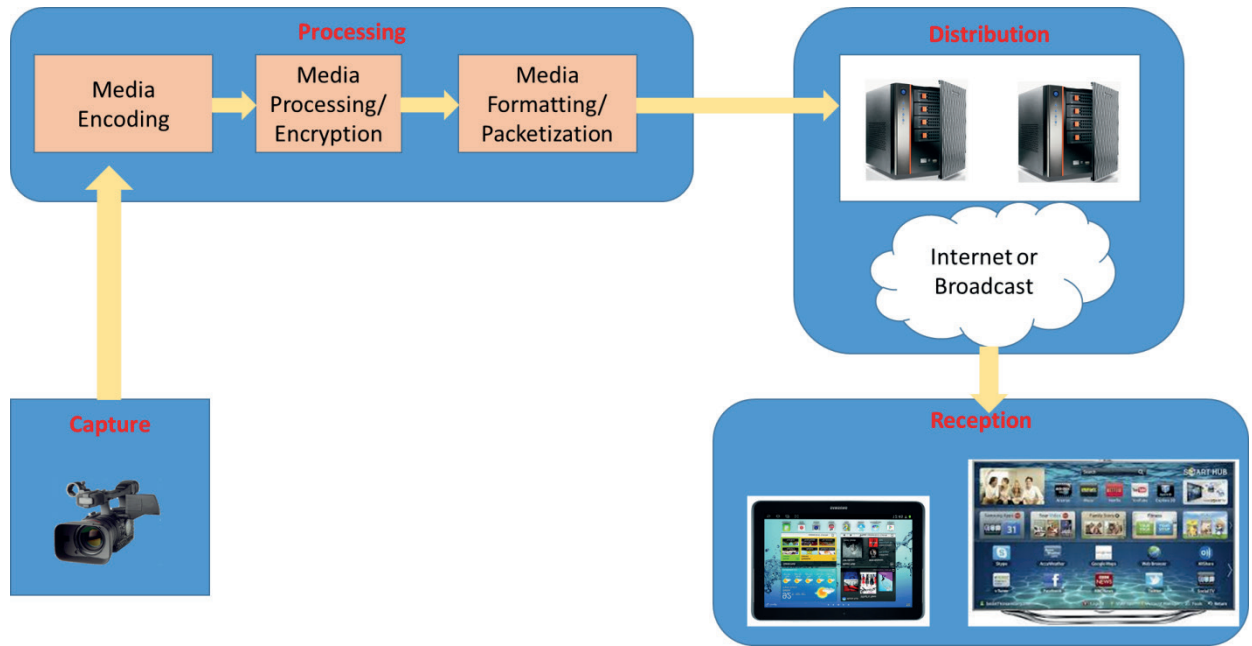


**Figure 3 — Example of a broadcast scenario**

After media encoding and any other related processing (such as encryption), the media data is formatted according to the transmission protocol in use and the packets are then sent down to the receivers as shown in [Figure 3](#). In the case of MMT, MMTP is the transport protocol used for media streaming. MMTP operates on MPUs in the MPU mode, which is the most appropriate mode for streaming. Theoretically, MPU should be completed to start packetizing it and send to the client. However, in real implementation, there are ways to further optimize generation of MPU and packetization of it to minimize delay by starting packetization and delivery of MPU before completion of generation of MPU.

The MPU mode of MMTP is designed to operate in a very low-delay mode and without any restrictions on the MPU size. An MPU is streamed progressively as soon as media data becomes available in a way similar to RTP streaming. Each media unit, such as an AVC NAL unit or an AAC audio frame, is encapsulated into an MMTP packet that also contains the MPU payload header. The MMTP packet looks as shown in [Figure 4](#).

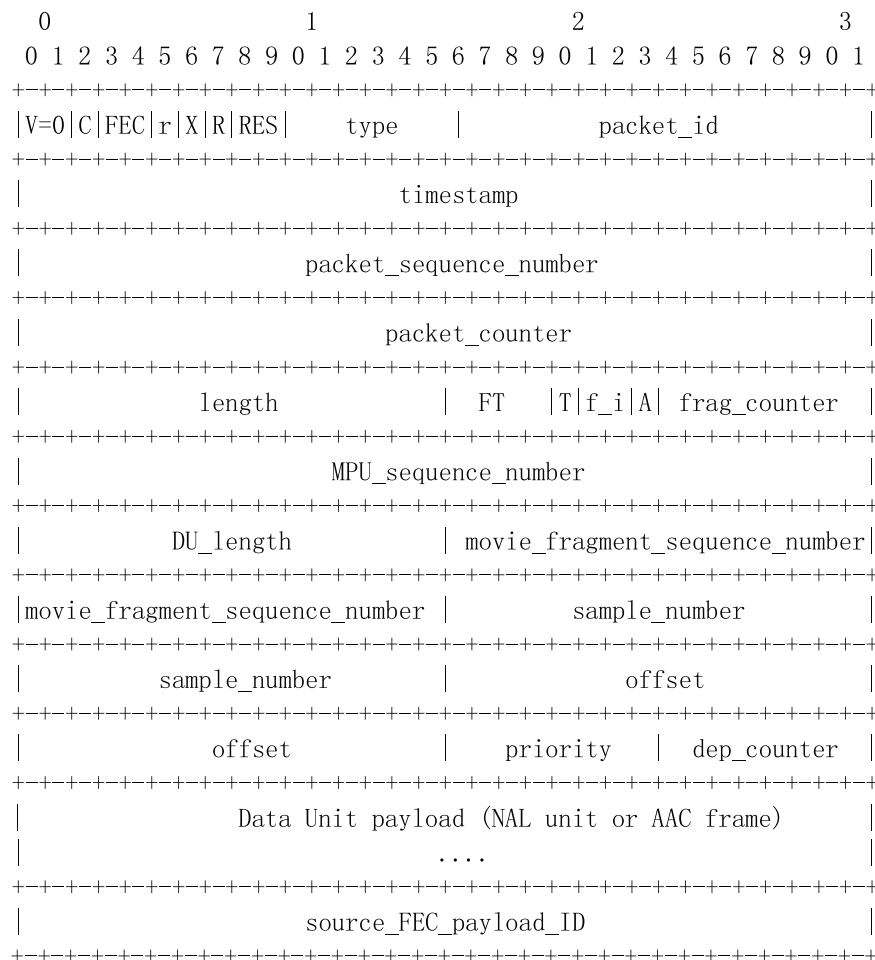


Figure 4 — MMTP packet

As can be seen from the packet header, all fields of the packet can be generated immediately at the streaming server, i.e. no need to delay the transmission of the media unit. In particular, the fields *movie\_fragment\_sequence\_number*, the *sample\_number* and the *offset* of the media unit in that sample are all known a priori for each media unit, even before generation of the complete movie fragment. The *priority* and *dep\_counter* fields may be decided based on the encoder configuration parameters. Most of the case, encoder predefines encoding structure, how many P-frames will be coded between I-frames and how many B-frames will be coded between I- and P-frames, before encoding starts. Therefore, high priority can be assigned for the NAL Units containing intra-coded macroblocks. *dep\_counter* can be set based on the coding structure. If multi-path encoding is performed at the encoder, such configuration could be altered during actual encoding process to improve encoding quality. However, in that case, actual coding structure is known before the last path which generates final compressed data. Therefore, the precise dependency structure can also be known before encoding of each video frame is completed.

The MPU metadata is a data which would need to be sent before transmission of compressed media data for low-delay processing at the client. It consists of the *ftyp* and *moov* boxes, where the latter does not contain any media sample tables and serves as the initialization data, which could be known before encoding is started. Consequently, the MPU metadata may be generated a priori with the knowledge of the media encoder configuration.

The movie fragment metadata contains the *moof* box which provides the timing information for the samples in the movie fragment, as well as their offset. The fragment metadata is constructed progressively and will be ready at the end of the movie fragment. This information is not required for the generation and delivery of the media units and will be sent out of order after all the media data of that movie fragment is transmitted.

At the receiver side, the receiver may either consume the media data immediately upon reception of a media unit or it may reconstruct the movie fragment first. In both cases, the overall delay is reduced down to the duration of a movie fragment or less than that.

The reconstruction of the movie fragment at the receiver side is straightforward. All media data that belongs to that particular movie fragment (based on the MPU\_sequence\_number and the movie\_fragment\_sequence\_number) is first collected progressively to build the mdat box. Finally, after reception of the movie fragment metadata, the fragment can be recovered fully. Any missing media data will be corrected by either marking it as lost or fixing the movie fragment metadata appropriately.

This operation mode is very close to that of RTP streaming and ensures minimal end-to-end delay. It still maintains the characteristics of streaming an ISO/BMFF file in a generic and media-independent way.

Another important aspect needs to be considered for low-delay streaming, in particular broadcast application is using very short duration for MPU. In a broadcast application, the client needs to quickly find a starting point of decoding. To support it, conventional broadcasting service repeatedly transmits initialization information for decoder and use very short duration for GOP. In MMT, as each MPU is self-contained, by defining the length of MPU as the period of repetition of decoder initialization parameter in conventional broadcast application, delay can be maintained same as that of conventional broadcast application.

MMTP allows for operation at very low end-to-end delay, in a way suitable to and required by several applications such as live broadcast applications.

## 5.5 Parallel processing in MMT sending and receiving entities

### 5.5.1 Processing in MMT sending entity

MMT protocol performs parallel generation of MMTP flows and signalling message generation/processing. The data packet processing part of MMT protocol performs the packetization of the media data using either the MPU mode for MPUs or the GFD mode for generic files. The generated source data is encapsulated into MMTP packets and transmitted to Transport layer. The signalling message generation part of MMTP processes CI, ADC and other data from the MMT Package and encapsulates them into signalling messages and packetizing and passing them to the Transport layer as shown in [Figure 5](#).

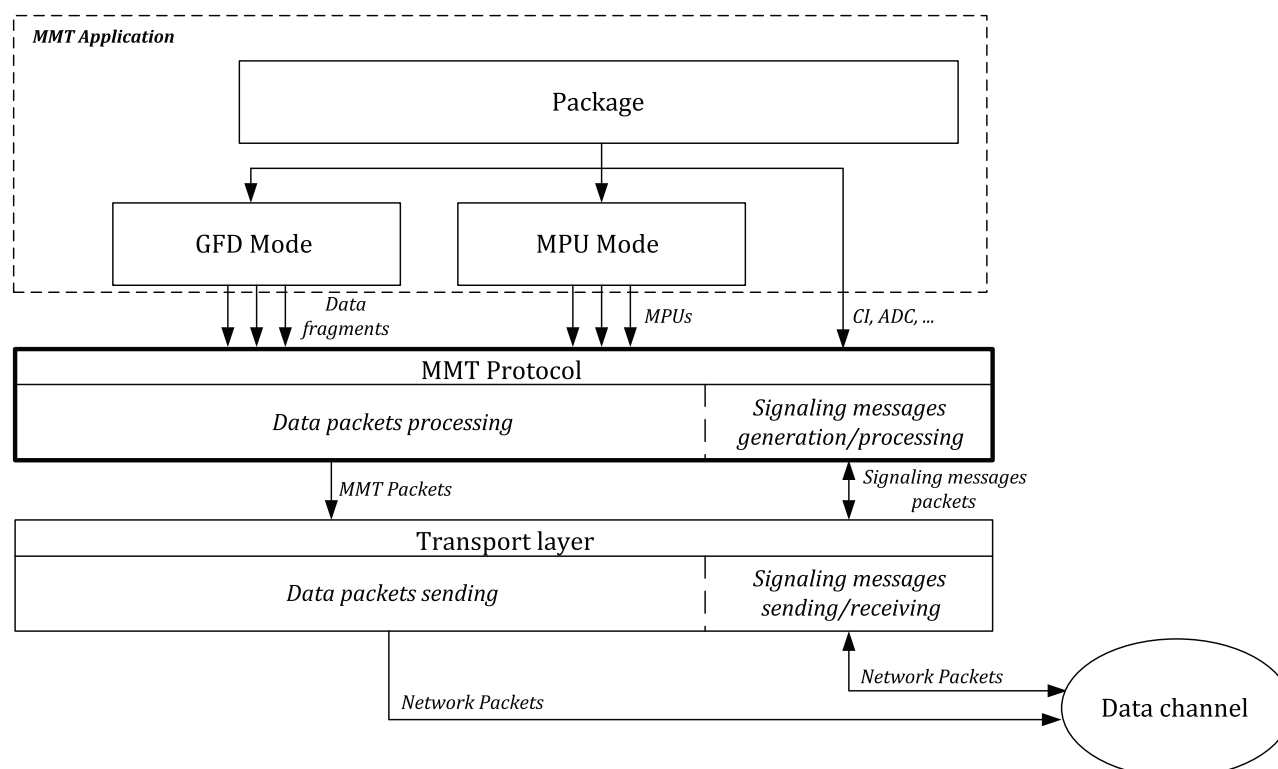


**Figure 5 — MMT sending entity structure**

More detailed architecture of data packet processing part of MMTP is presented in [Figure 6](#). MMTP packets are stored in separate buffers for each data flow after their processing with the help of data flow controller. After that, these MMT packets are passed to the corresponding MMT FEC scheme for protection. Each MMT FEC scheme returns repair symbols with repair FEC payload IDs and source FEC payload IDs. After that, the repair symbols are packetized into FEC repair packets and passed to the Transport layer. The identification of each FEC encoded flow and specifying of FEC coding structure and FEC code are provided by the FEC configuration information.

FEC source packets and their FEC configuration information for each data flow are passed to the corresponding MMT FEC scheme for protection. The MMT FEC scheme uses FEC code(s) for the repair symbol generation. Then FEC source and repair packets are delivered to the MMT receiving entity.

The data flow controllers of MMT sending entity may perform both encapsulation and packetization functions.

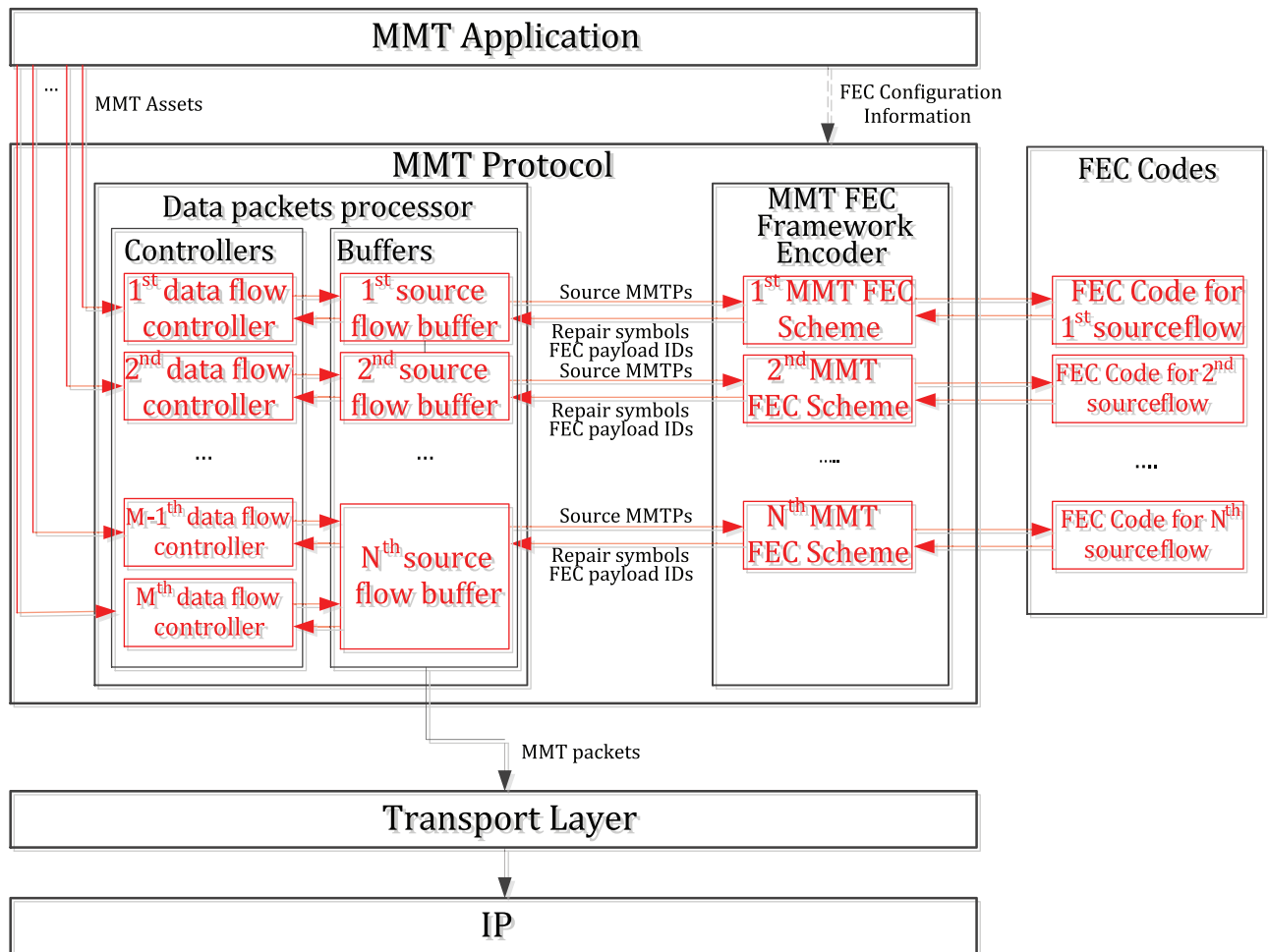


Figure 6 — Architecture for AL-FEC (MMT sending entity)

### 5.5.2 Processing in MMT receiving entity

MMT protocol performs parallel processing of MMT packet data flows and generation/processing of signalling messages. The data packet processing part of MMT protocol receives MMTP packets from the transport layer and transfers them into the corresponding data flow processor. Each data flow performs recovering of lost FEC source packets and then passes them for generic object reconstruction and/or MPU reconstruction, which happen in parallel. The signalling message receiving part of MMTP processes incoming signalling message packets and passes them for signalling message reconstruction. The reconstruction of generic objects and MPUs from different assets and of signalling messages is also performed in parallel as shown in [Figure 7](#).

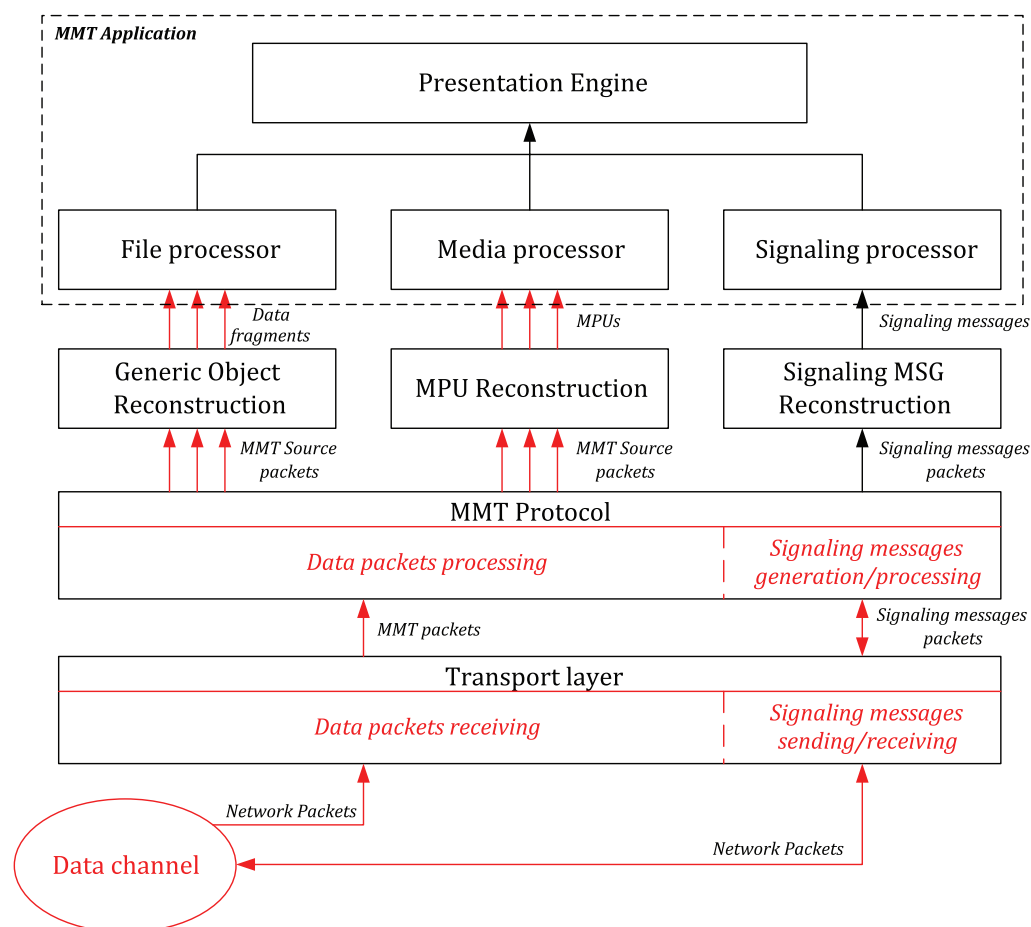


**Figure 7 — MMT receiving entity structure**

At MMT receiving entity, MMT protocol passes each FEC source flow and its associated FEC repair flow(s) to the corresponding MMT FEC scheme. After passing of flows to the MMT FEC schemes, each of the given schemes returns recovered source MMT packets. MMT packets are stored in separate buffers for each data flow and processed by separate data flow controllers. The outlined architecture is depicted in [Figure 8](#).

Data flow controllers of MMT receiving entity unite functions of de-capsulator, de-packetizer and de-jitter.

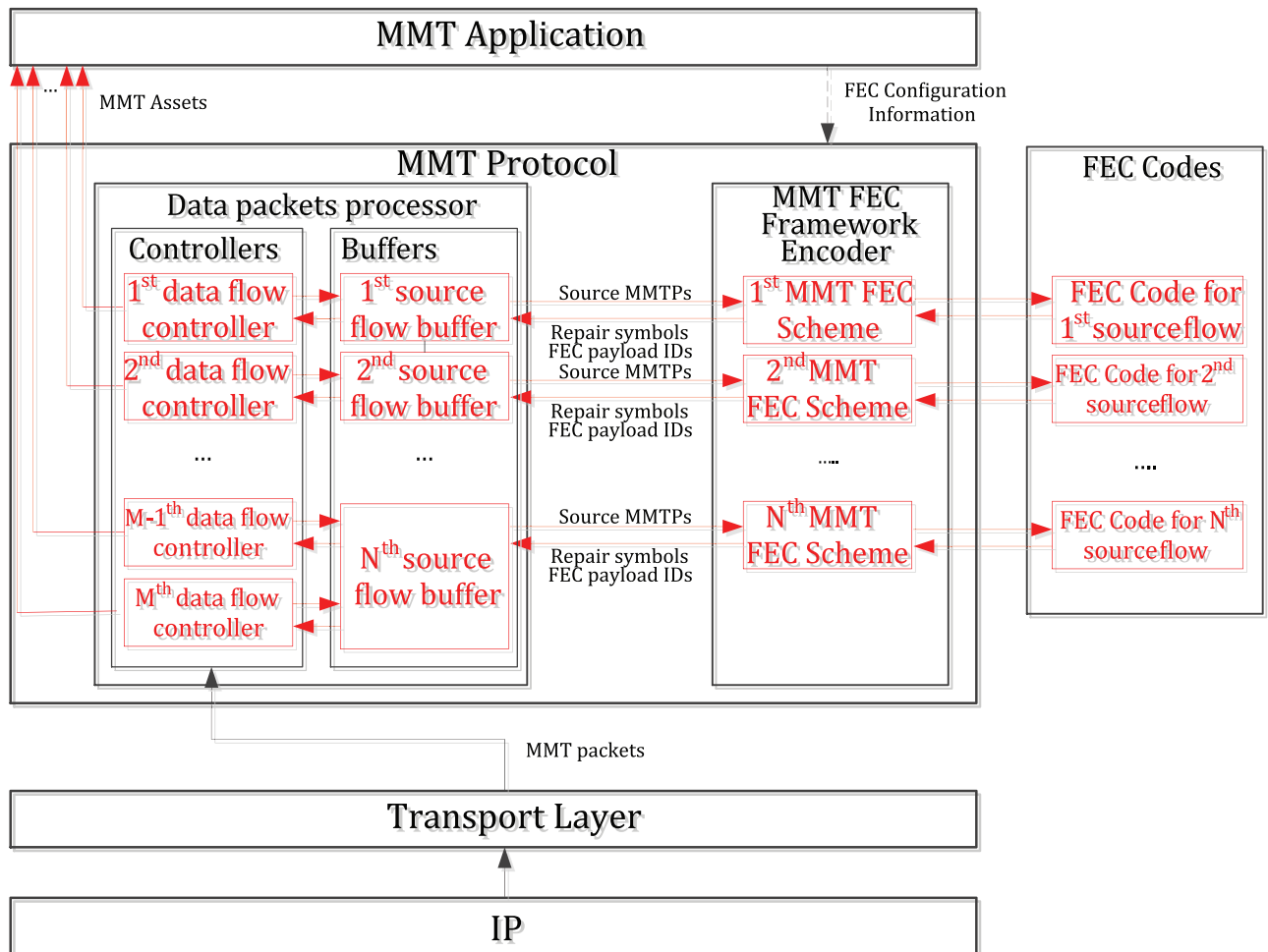


Figure 8 — Architecture for AL-FEC (MMT receiving entity)

## 5.6 MPU streaming for live services

### 5.6.1 MPU packetization

#### 5.6.1.1 MPU fragment building block

In MMT, there are two kinds of MPU structures: the timed data and the non-timed data. MMTP supports streaming modes, where the streaming mode is optimized for packetized streaming of ISO Base Media File formatted files. The MPU mode supports the packetized streaming of an MPU.

[Figure 9](#) depicts an MPU fragment building block for packetized streaming of an MPU. This MPU fragment building block helps to prepare for the packetization of an MPU into MMTP packets. The process of creating the MMTP packet passes by two steps: generating the MMTP payload and generating the MMTP packet.

According to the MMT specification, the format of the MMTP payload takes into account the boundaries of data in the MPU to generate packets using the MPU mode. The MPU metadata delivery data unit consists of the “ftyp” box, the “mmpu” box, the “moov” box and any other boxes that are applicable to the whole MPU. The FT field of the MMTP payload carrying a delivery data unit from an MPU metadata is set to “0x00”. The fragment metadata delivery data unit 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, MFUs stored in the “mdat” box of an MPU, is then split into multiple delivery data units in a media-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 of movie fragment metadata is set to 0x01. The media data, MFUs stored in the “mdat” box of an MPU, is then split into multiple delivery data units in a media-aware way. This may, for example, be performed with the help of the MMT hint track.

from an MFU is set to “0x02”. Each MFU is prepended with an MFU header. It is followed by the media data of the MFU.

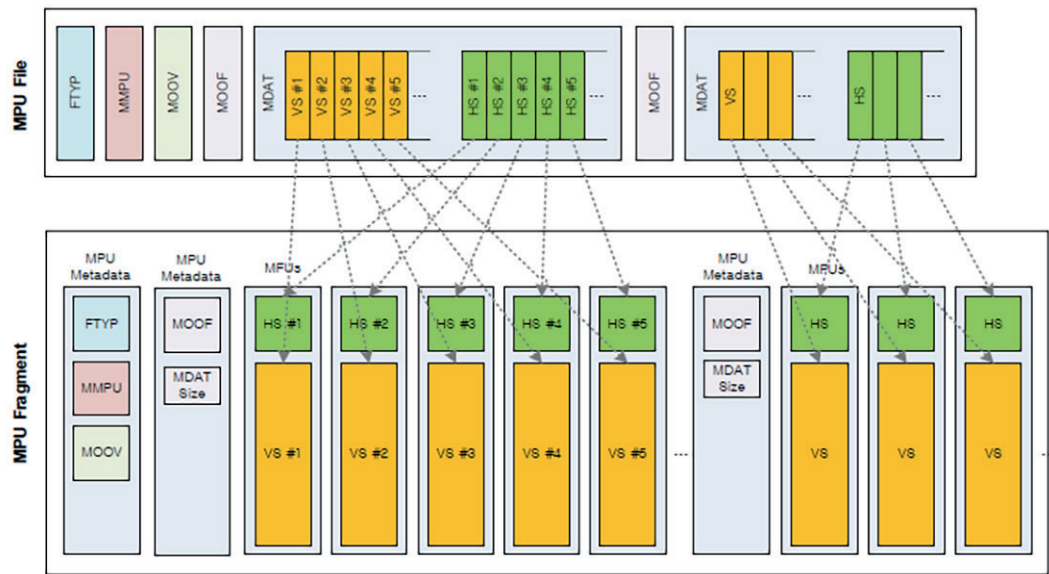


Figure 9 — MPU fragment building block

5.6.1.2 MMT broadcast sending procedure for low latency

5.6.1.2.1 General

When considering the low latency requirement of broadcasting services, the broadcasting system may consider reducing the delay resulting from both the transmission and the reception procedure, while also considering the requirements on random access to the broadcast channel. In order to support this use case, the MMT receiving entity has to also receive the related presentation information. The MMT sending entity may also consider sending the signalling information periodically during the MPU delivery as shown in [Figure 10](#).

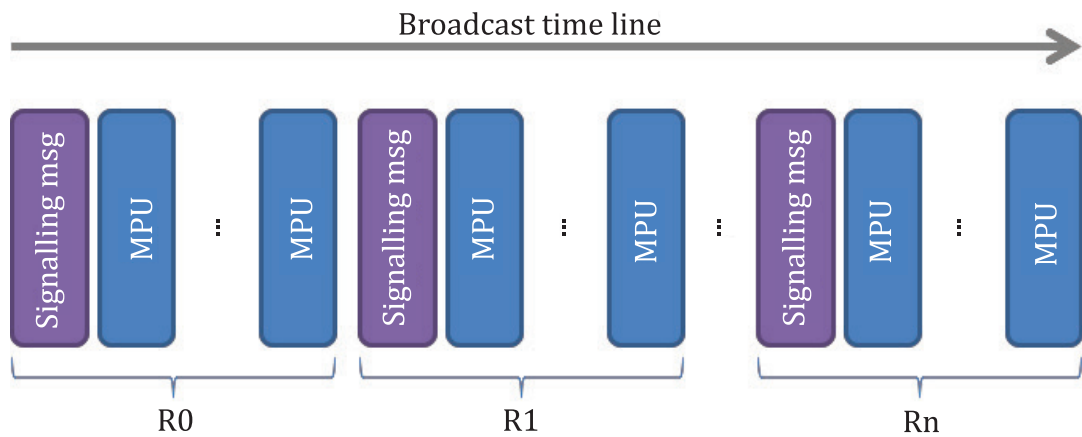


Figure 10 — Periodical signalling information transmission

### 5.6.1.2.2 MPU sending procedure

In order to reduce the end-to-end latency up to the reconstruction of the MPU, the MMT sending entity may change the transmission order of fragments comprising one MPU, which means, for example, transmission of MFUs precede that of fragment metadata. Additionally, for reliable transmission, the MMT sending entity may send the MPU metadata frequently during the transmission of the related MFUs.

When building the MPU fragment building block, the delivery data unit for fragment can be considered as the following two cases.

The first case is that the delivery data unit is MPU. Fragmentation indicator contains information about fragmentation of data unit in the payload, especially boundary information of delivery data unit. When the MPU is delivery data unit, the MMT receiving entity prepares the reconstruction of MPU in MMT delivery layer. The fragmentation indicator indicates that MPU metadata is the first of delivery data unit. The MMT receiving entity can wait to reconstruct the MPU until the fragment indicator which indicating the last of it is detected.

The second case is that the delivery data unit is either of MPU metadata, fragment metadata or MFU. In this case, the MMT receiving entity can process the reconstruction of each type of delivery data unit in MMT delivery layer for fast processing of delivery data and managed buffer status.

### 5.6.1.2.3 Signalling message sending procedure

Whenever an MMT receiving entity wants to present media resources in MPUs, it needs to receive the related signalling messages as soon as possible after joining a broadcast channel.

The channel reception can only be initiated after getting the related Presentation Information (PI). In order to minimize the channel switch time as short as possible, the frequency of sending the CI can be per-MPU basis.

However, there can be the time interval between consecutive MPUs in MMT broadcast, which effectively corresponds to the length of MPU playing time. Usually, signalling messages are much smaller than MPUs. Therefore, the sending time of signalling messages is trivial compared with that of the MPUs. It means when an MMT receiving entity randomly accesses the service, it is very highly probable that it will meet the MPU directly rather than signalling message. In that case, MMT sending entity can send the signalling message behind the MPU to make use of those received MPUs as shown in [Figure 11](#).

In this case, the channel process instance is initialized as soon as possible.

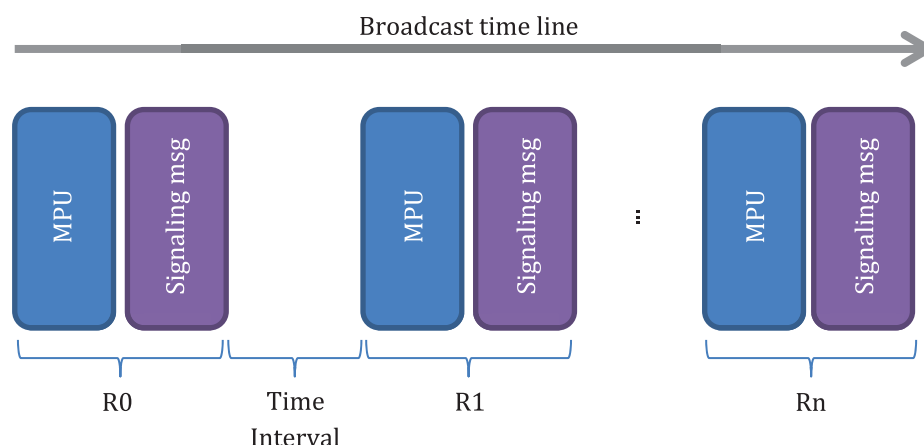


**Figure 11 — Signalling message behind the MPU**

### 5.6.2 Sending of MPU and signalling message

After channel process instance initialization, the media player should be initialized by MPU. Normally, the first completed MPU can initialize media player because the media player initialization only needs the head information of MPU.

When the MMT receiving entity randomly access to the broadcast channel, there are two cases.

The first case is that MMT receiving entity access to the channel while a certain MPU transmission is ongoing and the other case is that the MMT receiving entity access to the channel during time interval between consecutive MPUs as shown in [Figure 12](#).

#### Case 1

In this case, normally the first MPU received is not complete MPU because the MMT receiving entity could not receive packets transmitted before it joins the channel. In order to initialize the media player, it needs a complete MPU, therefore it needs to wait for the next one whole MPU data to arrive. In that case, the expected wait time before media player initialization,  $\Delta T$ , can be calculated using [Formula \(1\)](#):

$$\Delta T = T1 + T2 + T3 \quad (1)$$

where

- T1 is the residual time to finish receiving first MPU since MMT receiving entity joins to the channel;
- T2 is the time interval between two consecutive MPUs which is brought by the length of MPU playing time;
- T3 is the receiving time of the next first MPU (because the signalling message sending time is very short compared with that of MPU, it may not be considered).

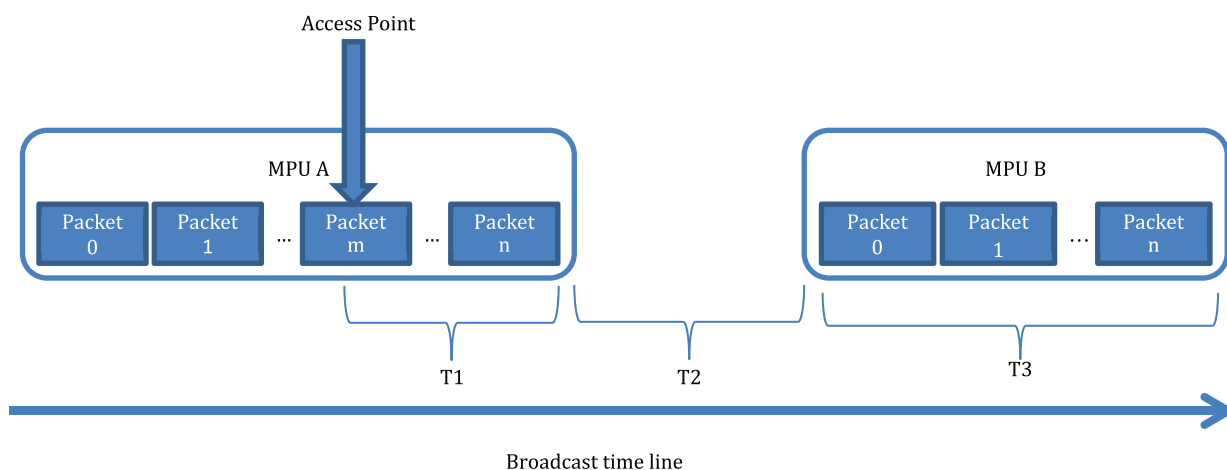


**Figure 12 — Random access in the middle of MPU (case 1)**

In this case, the MMT receiving entity can easily find a way to play the first MPU, by using the signalling message sent right after the MPU, then the wait time of T2 and T3 can be saved.

In MMT protocol, the payloads are generated according to the following description (see [Figure 13](#)):

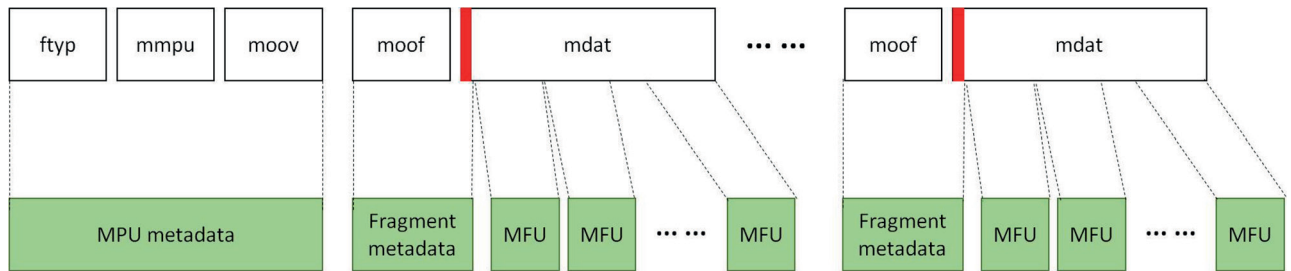


**Figure 13 — MMTP payload generation**

In this case, only one moof is included in one MPU. So MMT receiving entity can classify all packets as three categories. The first category belongs to MPU metadata. The second category belongs to fragment metadata and the third one belongs to MFU. MFU is considered as a (sub)sample of media.

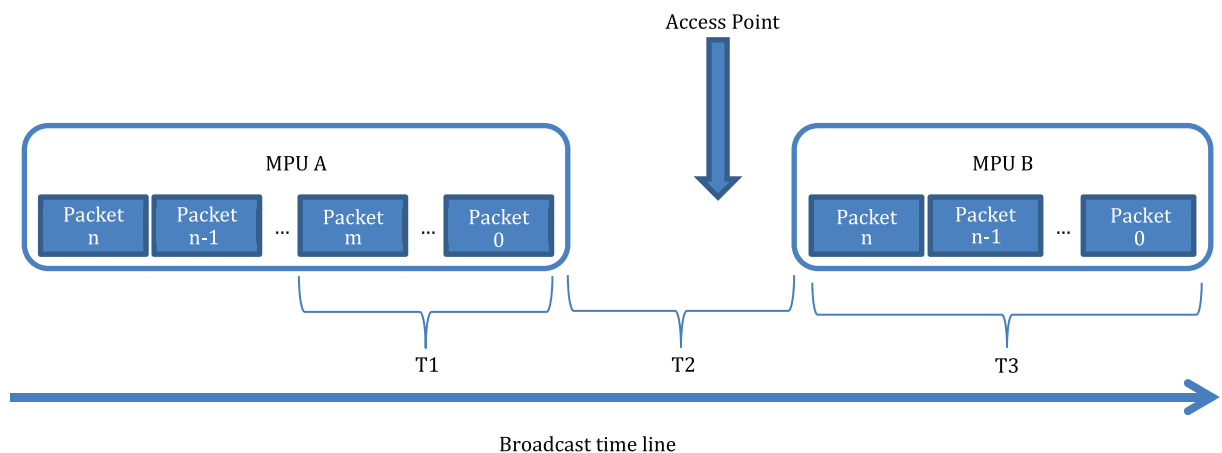


**Figure 14 — Random access in the middle of interval (case 2)**

Moreover, in this case, it can be ensured that the MPU metadata and the fragment metadata can be received by terminals with the highest probability. Then the waiting time should be as shown in [Formula \(2\)](#):

$$\Delta T = T1 \quad (2)$$

### Case 2

In this case, the first MPU received should be a complete MPU as shown in [Figure 14](#). The total waiting time should be part of T2 and whole T3. It should be as shown in [Formula \(3\)](#):

$$\Delta T < T2 + T3 \quad (3)$$

## 5.7 Fast MMT session acquisition

In multimedia streaming environment, user switches between media which are very huge size when they are consuming the contents. In such scenarios, low-delay session acquisition has a significant impact on the perceived quality by the end user.

MMT streaming is based on the MPU concept as well as other kinds of data (e.g. media data itself and signalling messages) being fragmented and multiplexed into a sequence of MMTP packets for transmission.

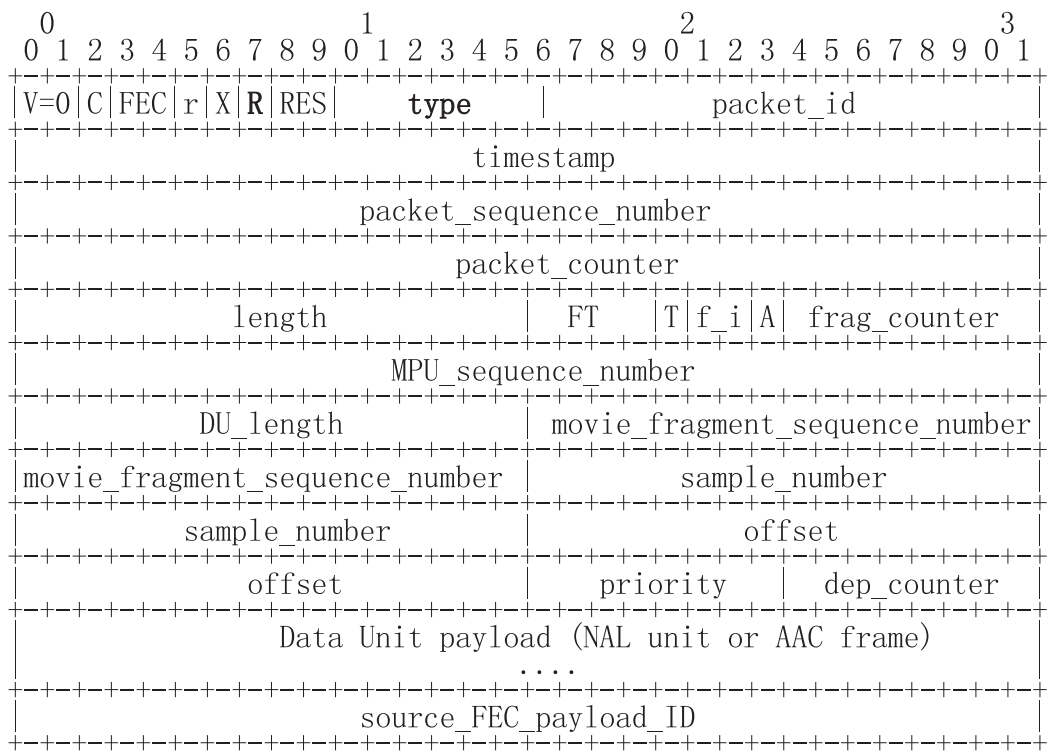


Figure 15 — Structure of MMTP packet

In the current MMTP packet header, the fields of *RAP\_flag* (R: 1 bit) and *type* (6 bits) can be used for fast session acquisition as shown in [Figure 15](#).

The *RAP\_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* should be set to mark data units of MPU Fragment Type value 0 and 1 and for MFUs that contain a sync sample or a fragment thereof, in the case of timed media, and for the primary item of non-timed MPUs. RAP can be MPU metadata, signalling message or frames which contain I-frames.

The *type* indicates the type of payload data, i.e. MPU, generic object, signalling message. As one more further step when MMTP packet includes MPU data (*type*=0x00), MMT receiving entity can also find out whether it has MPU metadata by checking MPU Fragment Type (FT: 4 bits) which is in the payload header indicating the fragment type as defined in ISO/IEC 23008-1:2014, Table 4.

For fast accessing of the IDR frames after constructing MPU, MMT receiving entity can be processed to parse “moov” box in the MPU metadata which indicated the presentation time for sample and also processed to get starting time from MPU\_timestamp descriptor or begin time specified in MPEG-CI.

If *RAP\_flag* is set to “0”, it means the corresponding MMT packet contains no high priority random access data. Then MMT receiving entity can discard the MMTP packets without *RAP\_flag* set to “1” for efficient buffer management before meeting the *RAP\_flag* set to “1” or processing it depending on its policy or MMT receiving entity situation.

5.8 Referencing and processing non-timed data

5.8.1 General

MMTP provides a mode for the grouped transmission of non-timed media resources, such as resources of a website. This mode facilitates the referencing, reception and consumption of media resources that are closely related. For instance, a receiver that is consuming a website would only need to locate a single resource and receive it instead of receiving the root resource, wait for the application to parse it and detect the referenced resources and then request the MMTP receiver to receive them and make

them available. On the sender side, this removes the necessity of describing every single resource and its location in the signalling.

### 5.8.2 Resource grouping and referencing

The MMTP sender is instructed to send a set of related non-timed media resources with a designated primary resource/entry point. The primary resource is usually the resource that connects all other resources or that constitutes the entry point for the application. Based on this information, the MMTP sender decides to encapsulate these related resources and send them together.

The MMTP sender sets the MP table according to the following rules.

- The `identifier_type` of the `identifier_mapping` syntax element is set to 0x01.
- The `URL_count` field is at least equal to 1.
- The first URL in the list should be the URL of the primary resource/entry point.
- It is recommended that only the URL of the primary item is provided in the MP table for the corresponding asset.
- The `asset_type` is set to “mmpu”.

It then creates the MPU by setting the primary resource as the primary item of the MPU and the remaining resources as additional items. The sender should set all resource metadata, such as URL and MIME type of the resource as part of the item information box.

The presentation information is authored to reference the primary resource.

### 5.8.3 Receiver handling

Based on the information received as part of the presentation information, the receiver detects a referenced resource. It builds the URL for that resource and requests the MMTP receiver to make that resource available. The MMTP receiver checks the MP table to locate the asset that carries that resource. It finds out that the asset is of type “mmpu” and lists the URL of the requested resource. It then uses the `packet_id` of that asset to receive and reconstruct the MPU.

Once the MPU that contains the primary resource is reconstructed, the receiver extracts all embedded resources together with their metadata and makes them available to the application, e.g. through a web cache. The application requesting the primary resource will later find all related resources available and quickly accessible from the cache.

## 5.9 Media adaptation for quality control in MMTP

### 5.9.1 General

This subclause shows how MMT protocol can transmit media streams adaptively to environment changes, such as network congestions, while also minimizing the service quality degradation.

### 5.9.2 Parameters for media adaptation

MMT streaming is to deliver MPUs which is a media file based on ISO-based media file format (ISO-BMFF). To transmit the MPU, it is fragmented and packetized into MMTP packets which have three different payload fragments, MPU metadata, fragment metadata and MFU.

Each MPU has “ftyp”, “sidc”, “mmpu”, “moov” and “moof” box in MPU metadata. Among them, “mmpu” box includes asset identifier, MPU information and *is\_complete* parameter indicating whether this MPU has all MFUs described by the MFU structure or not. The “moov” box contains all codec configuration information for decoding and presentation of media data and especially MMT hint track providing the information to convert encapsulated MPU to MMTP payloads and MMTP packets.

Those MMTP hint track provides two important parameters: *priority* and *dependency\_counter*. *priority* indicates the priority of the MFU relative to other MFUs within a MPU and *dependency\_counter* indicates the number of MFUs whose decoding is dependent on this MFU.

These two parameters can be utilized to judge the importance of the MFUs in an MPU.

### 5.9.3 Adaptation operation of MMT entity

When MMT sending entity transmits media streams to MMT receiving entity, there may be cases that MMT sending entity can intentionally skip some portion of it. For example, MMT sending entity can know poor network condition or lack of resources in the MMT receiving entity side through some feedback. Then MMTP packets will be dropped anyway according to network condition by MANE or other network entities if there is no adaptation operation. In streaming service case, omission of some media may not affect critically to user experience.

MMT sending entity can skip transmission of some MFUs to accommodate to network condition. The MMT sending entity needs information to decide which MFUs they need to omit. In that case, *priority* and *dependency\_counter* can provide the priority information between MFUs. MMT sending entity should drop MFUs with the lowest priority and highly independent from other MFUs first by comparing *priority* and *dependency\_counter*. For example, MFUs including RAP information will have high priority value and MFUs corresponding I-frame will have *dependency\_counter* value.

MMT sending entity also has to announce to MMT receiving entity side that some of MFUs are intentionally dropped. If that information is not provided to MMT receiving entity side, they will wait for MFUs which are intentionally omitted or try to request again. When *is\_complete* is set as "0", MMT receiving entity should recognize there are more than one missing MFUs and should reconstruct MPU through current MFUs without requesting MFUs in that MPU. For that reason, MMT sending entity has to set *is\_complete* as "0" to acknowledge that current MPU has some missing MFUs to MMT receiving entity side when some MFUs are omitted intentionally.

## 5.10 Hybrid delivery in MMT

### 5.10.1 General

This subclause provides MMT implementation information when MMT Assets are delivered on hybrid networks.

Hybrid delivery in this subclause is defined as simultaneous delivery of one or more content components over more than one different types of network. One example is that one media component is delivered on broadcast channels and the other media component is delivered on broadband networks. The other example is that one media component is delivered on broadband networks and the other media component is delivered on another broadband networks.

### 5.10.2 Classification of hybrid delivery

The basic concept of hybrid delivery is to combine media components on different channels. However, in practice, there are several scenarios for hybrid delivery. The classification can be classified as follows.

- Live and non-live:
  - combination of streaming components (see [Figure 16](#));
  - combination of streaming component with pre-stored component (see [Figure 17](#));
- presentation and decoding:
  - combination of components for synchronized presentation (see [Figure 16](#));

- combination of components for synchronized decoding (see [Figure 18](#));
- same transport schemes and different transport schemes:
  - combination of MMT components;
  - combination of MMT component with another-format component such as MPEG-2 TS.

NOTE The case in which another-format component is encapsulated into MMT-format is excluded.

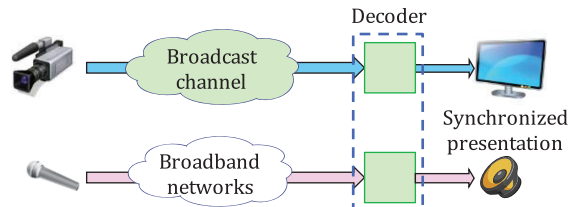


**Figure 16 — Combination of streaming components for presentation**

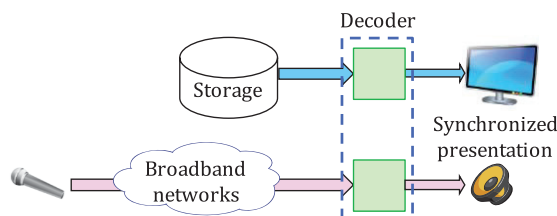


**Figure 17 — Combination of streaming component with pre-stored component for presentation**

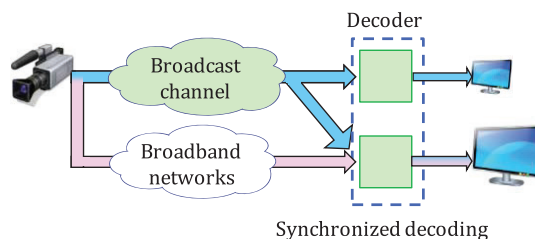


**Figure 18 — Combination of components for decoding**

The so-called seamless switching can be categorized into hybrid delivery of streaming components for presentation since switching components is possible when both components are synchronized for presentation.

### 5.10.3 Technical elements for hybrid delivery

While many types of information are specified in MMT signalling messages and MMT-PI, three types of information are mainly required for hybrid delivery:

- MMT asset information;

NOTE When an MMT component is combined with another-format component, the latter component can be neither an asset nor an MPU.

- information on spatial relationships among media components;
- signalling message for media consumption.

[5.11](#) provides detailed information on the implementation of MMT for individual cases categorized in [5.10.2](#).

## 5.11 Example of detailed implementation of MMT

### 5.11.1 Use case: Combination of MMT and MPEG-2 TS for synchronized presentation

#### 5.11.1.1 General

This subclause describes the case in which MMT and MPEG-2 TS are used as a transport scheme in broadband networks and broadcast channels, respectively.

#### 5.11.1.2 MMT asset information

In order to identify the type and location of media components of MPEG-2 TS in broadcast channels, the following signalling messages are required in MMT in broadband networks:

- `MMT_general_location_information` syntax in MP table.

This syntax specifies the address of a media component of MPEG-2 TS in broadcast channels. In the case of combination with MPEG-2 TS in broadcast channels, MPEG-2 TS location is used.

A typical example is to identify `network_id` (16 bits assigned by SDO), `MPEG-2_transport_stream_id` (16 bits assigned by the operator) and `MPEG-2_PID` (13 bits assigned by the operator). By using these IDs, a media component in broadcast channels can be identified.

#### 5.11.1.3 Information on temporal relationships among media components

MPEG-2 TS components have timestamps based on STC. MMT components have timestamps based on UTC. To synchronize these different types of timestamps in MMT and MPEG-2 TS, Clock Relation Information messages are required.

- Clock Relation Information message can carry a set of `STC_sample` and `NTP_timestamp_sample` that are identical timing.

At an MMT compliant receiving entity, the STC based clock in MPEG-2 TS can be converted to the wall clock based on UTC by processing the Clock Relation Information. An MMT component in broadband networks and an MPEG-2 TS component in broadcast channels are presented in synchronized manner since both components can share the same time domain as wall clock.

### 5.11.2 Use case: True real-time video streaming over lossy channel

#### 5.11.2.1 General

This subclause describes the case in which the client starts replaying video within 150 ms after requesting a video to the server<sup>[7]</sup>. It is assumed that de-jittering delay is zero.

#### 5.11.2.2 Main features

**MFU-by-MFU processing:** In the server and client, video data is processed MFU by MFU rather than MPU by MPU. An MPU is not reconstructed in the client. An MFU is replayed before receiving its MPU fully. In the server, video data is packetized and sent as soon as being parsed as part of MPU. At the same time, the sent packets are stored in the FEC encoder buffer. There is no bi-directional frame, i.e. only intra- and predictive frames.

**Short FEC period:** FEC period is independent of borders of MPU and MFU. It is much shorter than an MPU period. Number of data symbols,  $k$ , is fixed ( $k = 49$  in the implementation) whenever the number is reached, repair packets are calculated and sent [10 repair packets,  $(n, k) = (59, 49)$  in the

implementation]. When symbol size  $T$  is 800 bytes and packet size is 1 500 bytes, repair packets are sent in every 24 to 25 packets, which is about 100 ms long on average for 3 Mbps video  $[(49 \text{ symbols/FEC}) \times (1\,500 \times 8 \text{ bits/packet}) / (2 \text{ symbols/packet}) / 3 \text{ Mbps}]$ .

**Centralized timing control:** In the server and client, timing is controlled by the session object.

In every MFU period, the client session object orders to retrieve an MFU from FEC buffer and decode and replay it. The client session object orders to flush the receiver socket buffer to the FEC ring buffer from time to time [in every MFU period (= 33 ms) in the implementation].

### 5.11.2.3 Ring-buffer in the client

The ring buffer is composed of  $N$  FEC associate groups. The number  $N$  is determined in terms of de-jittering delay plus some margin in order to prevent buffer overflow. Here, “buffer overflow” means over-writing lately arrived packets on the packets which are not read by the FEC decoder yet [ $N = 50$ , in the implementation;  $N = 50$  allows about 5 s de-jittering delay and total buffer size becomes 2,2 Mbytes =  $3 \text{ Mbps} / (8 \text{ bits/byte}) \times 5 \text{ s} \times (59/49)$ ].

### 5.11.2.4 FEC/deFEC performance and delay

Code rate is 49/59, which is enough to recover up to 15 % packet loss. The code rate could be adaptive to channel condition. Mostly, deFEC is successful at most,  $k+2$  symbols. Theoretical failure rate at  $k+2$  is  $10^{-4}$ .

Since data packets are sent immediately after being stored in the FEC encoder buffer, total delay FEC encoding and decoding is one associate group period (about 100 ms in the implementation).

Complexity of FEC encoding is very low. Because unlikely as Raptor Q, all the coefficients are stored in the coefficient table and calculation in the server is just to calculate weighted sum of symbols [that is, a repair packet is calculated by just GF(256) multiplication of a  $T$  by  $k$  array and  $k$  by 1 column vector]. Complexity of FEC decoding is also very low since systematic FEC has been used. If  $L$  symbols are lost, then, total calculation includes one  $L$  by  $L$  inverse matrix calculation and GF(256) multiplication of an  $L$  by  $L$  array and an  $L$  by  $T$  array ( $T = 800$ ,  $k = 49$ ,  $L < 9$ , in the implementation).

## 5.12 HRBM signalling for hybrid delivery

### 5.12.1 Hybrid delivery from the single MMT sending entity

As shown in [Figure 19](#), the main station is composed of the broadcast tower and at least one of the broadcast services are provided to any broadcasting service using the hybrid delivery.

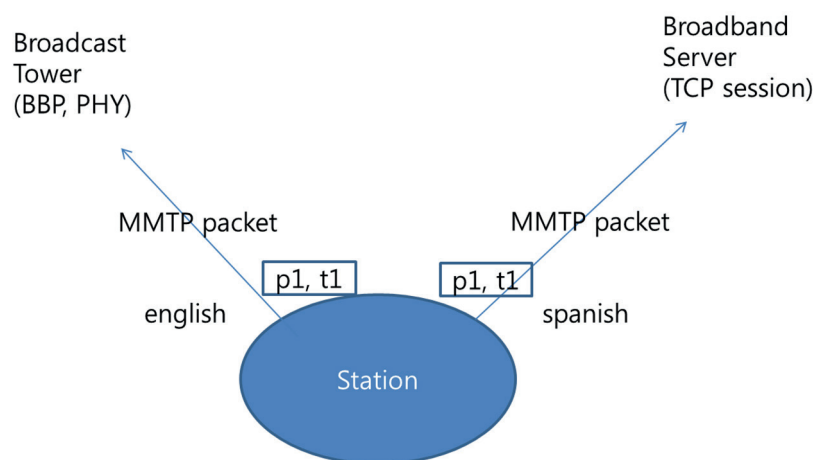


Figure 19 — Single station for hybrid delivery