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

**Part 13:
MMT implementation guidance**





COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, 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
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier; Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword	vii
Introduction	ix
1 Scope	1
2 Normative references	1
3 Terms and definitions	1
4 General overview of MPEG media transport	1
4.1 System overview	1
4.2 Tools specified in ISO/IEC 23008-1	2
5 MMT function deployments	3
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	6
5.3 Default assets	6
5.4 Low-delay live streaming	7
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
5.6.3 MPU generation for SHVC-encoded video in real-time streaming	17
5.7 Fast MMT session acquisition	19
5.8 Referencing and processing non-timed data	20
5.8.1 General	20
5.8.2 Resource grouping and referencing	20
5.8.3 Receiver handling	21
5.9 Media adaptation for quality control in MMTP	21
5.9.1 General	21
5.9.2 Parameters for media adaptation	21
5.9.3 Adaptation operation of MMT entity	21
5.10 Hybrid delivery in MMT	22
5.10.1 General	22
5.10.2 Classification of hybrid delivery	22
5.10.3 Technical elements for hybrid delivery	23
5.11 Example of detailed implementation of MMT	24
5.11.1 Use case: Combination of MMT and MPEG-2 TS for synchronized presentation	24
5.11.2 Use case: Combination of MMT and HTTP streaming for synchronized decoding	24
5.11.3 Use case: Content request in advance for synchronized play-out	25
5.12 HRBM signalling for hybrid delivery	26
5.12.1 Hybrid delivery from the single MMT sending entity	26
5.12.2 Hybrid delivery from the multiple MMT sending entities	27
5.13 Error resilience in MMT protocol	29
5.14 Delay constrained ARQ	30
5.14.1 General	30
5.14.2 Delivery-time constrained ARQ	30
5.14.3 Arrival-deadline constrained ARQ	31
5.15 Delivery of encrypted MPUs	33
5.16 HRBM message updating	33
5.16.1 General	33

5.16.2	HRBM message sending schedule.....	34
5.16.3	Use case.....	34
5.16.4	HRBM buffer operation in unicast environment.....	35
5.17	MMTP packet with padded data.....	37
5.18	Constraints on signalling splicing points	39
5.18.1	General.....	39
5.18.2	Constraints on Case 1 – Asset change at the start of MPU.....	39
5.18.3	Constraints on Case 2 – Asset change at a point in MPU.....	39
5.18.4	Signal the splicing point for target assets in Case 1 and Case 2.....	40
6	Use cases for MMT deployment.....	40
6.1	General.....	40
6.2	Delivery of DASH presentations using MMT.....	40
6.2.1	General.....	40
6.2.2	Delivery of the MPD.....	41
6.2.3	Delivery of the data segments.....	41
6.3	Client operation for DASH service delivered through MMT protocol.....	42
6.3.1	Delivery of MPD with MMTP.....	42
6.3.2	Delivery and consumption of DASH segments with MMTP.....	42
6.4	Hybrid of MMT and DASH over heterogeneous network.....	44
6.5	MMT caching for effective bandwidth utilization.....	45
6.5.1	Overview of MMT caching middlebox architecture.....	45
6.5.2	Content-based caching of MMT media.....	46
6.5.3	MPU sync protocol between server and caching middlebox.....	48
6.5.4	MMT cache manifest.....	53
6.6	Usage of ADC signalling message.....	55
6.6.1	General.....	55
6.6.2	Operation in MMT sending entity.....	55
6.6.3	Operation in MANE router.....	55
6.6.4	Example operation in MMT-receiving entities.....	55
6.6.5	QoE multiplexing gain and bottleneck coordination.....	55
6.7	MMT deployment in Japanese broadcasting systems.....	58
6.7.1	General.....	58
6.7.2	Broadcasting systems using MMT.....	59
6.7.3	Media transport protocol.....	61
6.7.4	Signalling information.....	66
6.7.5	Start-up procedure of broadcasting service.....	74
6.7.6	Actual packet structure.....	77
6.8	MMT deployment in ATSC 3.0 systems.....	79
6.9	Implementation of MMT based on D-TMB in China.....	81
6.9.1	Background.....	81
6.9.2	MMT over legacy DTMB infrastructure.....	81
6.9.3	Use cases.....	81
6.10	Conversion of MMTP stream to MPEG-2 TS.....	83
6.10.1	Overview of conversion operation.....	83
6.10.2	Restrictions to MMTP packets.....	83
6.10.3	Calculation of PTS, DTS.....	83
6.10.4	Restriction related to MPEG-2 T-STD.....	84
6.10.5	Packet field conversion rule.....	85
6.10.6	PSI Conversion rule.....	86
6.11	MMT service provisioning at conventional broadcast environment.....	87
6.12	Usage of multimedia configuration for interface switching management.....	89
6.13	MMT signalling for multiple timed text assets.....	89
6.13.1	Multiple timed text assets within an MMT presentation.....	89
6.13.2	Selective spatial assignment for multiple timed text assets.....	90
6.13.3	Example of multiple timed text assets signalling in MMT.....	92
6.13.4	Carriage of TTML based timed text in MMT.....	92
6.14	Viewport-dependent baseline media profile with packed streaming for VR.....	94

7	Application layer forward error correction (AL-FEC)	97
7.1	FEC decoding method for ssbg_mode2	97
7.1.1	General	97
7.1.2	Source symbol block format for ssbg_mode2	97
7.1.3	Regionalization of source symbol Block for FEC decoding	98
7.1.4	How to choose a proper unit of data for FEC decoding	102
7.2	Usage of two stage FEC coding structure	102
7.2.1	General	102
7.2.2	Use case: Hybrid content delivery	103
7.2.3	Use case: Streaming multicasting (or broadcasting) to two different end-user groups which is under two different channel conditions each other	104
7.3	Usage of layer-aware FEC coding structure	104
7.3.1	General	104
7.3.2	Use case 1: Layered multicast streaming	105
7.3.3	Use case 2: Hybrid delivery	106
7.3.4	Use case 3: Fast zapping with long time interleaving (LA-FEC UI)	107
7.3.5	Use case 4: Prioritized transmission	107
7.4	MPU mapping to source packet block	108
7.4.1	General	108
7.4.2	Aligned MPU mapping method to source packet block	108
7.5	FEC for hybrid service	109
7.6	Usage of rate-adaptive AL-FEC	111
7.6.1	General	111
7.6.2	AL-FEC rate control	112
7.7	FEC scheme for interleaved source symbol block	115
7.7.1	General	115
7.7.2	Re-order buffer for interleaved source symbol block	115
8	MMT developments in mobile environments	115
8.1	True realtime video streaming over a lossy channel	115
8.1.1	General	115
8.1.2	Main features	115
8.1.3	Ring buffer in the client	116
8.1.4	FEC/deFEC performance and delay	116
8.1.5	Characteristics of UDP based multi-path and multi-session transmitting for true real-time video streaming	116
8.2	Dynamic asset change	119
8.3	Media adaptation for quality control	121
8.3.1	Use cases of QoS management for adaptive video service	121
8.3.2	Advanced adaptation operation of MMT entity: selective transmission	125
8.4	Transition time decision using MTR and MTN	129
8.5	Usage of DRI and DSI message	130
8.5.1	Overview	130
8.5.2	Operation in MMT sending entity or MANE	130
8.5.3	Operation in MMT receiving entity	131
8.6	Usage of dynamic media resource identification information update	131
8.6.1	General	131
8.6.2	Classification of dynamic media resource allocation	131
8.6.3	Operation in MMT sending entity	131
8.6.4	Operation in MMT receiving entity	132
8.6.5	Example operation in MANE proxy	132
8.7	MMT service list	134
8.8	Usage of DNS message for MMT URL resolution	135
8.8.1	General	135
8.8.2	Example on DNS resolution procedure	135
8.9	Usage of guide information for dynamic media resource allocation	137
8.9.1	General	137
8.9.2	Classification of guide information	137
8.9.3	Classification of dynamic media resource allocation	137

8.9.4	Operation in MANEs	138
8.9.5	Operation in MMT sending entity	138
8.9.6	Example of dynamic media resource allocation using guide information	138
8.10	Usage of DARI for supporting DNS in MMT	139
8.10.1	General	139
8.10.2	Classification of DARI proxy	139
8.10.3	Operation in MANE DNS or DNS	140
8.10.4	Operation in DARI proxy	140
8.10.5	Relation between the media resource identification in MMT and DNS message	140
Bibliography		142

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.

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).

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) or the IEC list of patent declarations received (see <http://patents.iec.ch>).

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

For an explanation of 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 www.iso.org/iso/foreword.html.

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

This third edition cancels and replaces the second edition (ISO/IEC 23008-13:2017), which has been technically revised. The main changes compared to the previous edition are as follows:

- Guidance added to show how the MMT protocol can transmit media streams adaptively to environment changes such as network congestions, while also minimizing service quality degradation.
- Guidance added to describe the scenario in which MMT and MPEG-2 TS are used as transport schemes in broadband networks and broadcast channels, respectively.
- Guidance added for constraints on signalling splicing points that are specified for changing points or splicing points on MMT assets.
- Application Layer Forward Error Correction (AL-FEC) guidance added to describe the usage of Rate-Adaptive AL-FEC, Layer-Aware (LA) FEC coding structure and FEC scheme for interleaved source symbol block.
- Broadcasting MMT deployment guidance added to describe the implementation of MMT based on D-TMB in China and MMT Deployment in ATSC 3.0 systems.
- MMT deployment guidance added to show the usage of MMT signalling for multiple timed text assets and for the viewport-dependent baseline media profile with packed streaming for VR.
- MMT developments in mobile environments guidance added to describe the usage of true real time video streaming over lossy channels, dynamic asset change, and media adaptation for quality control.
- MMT developments in mobile environments guidance added to describe the usage of signalling messages for supporting Package retransmission and dynamic media resource allocation.

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

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

This document provides guidance for implementation and deployment of multimedia systems based on ISO/IEC 23008-1. These document include the following:

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

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

Part 13: MMT implementation guidance

1 Scope

This document provides guidance 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 23008-1:2017, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 1: MPEG media transport (MMT)*

3 Terms and definitions

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

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

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

4 General overview of MPEG media transport

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.

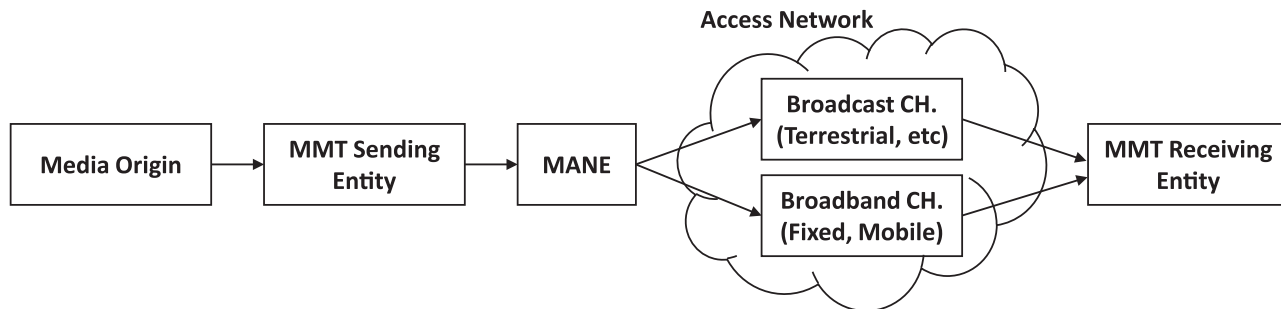


Figure 1 — Example of MMT-based media distribution chain

The MMT aware network element (MANE) may be any network element, such as, media caches and routers, that 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 the MMT receiving entity has to follow. This information is provided by the 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 Tools specified in ISO/IEC 23008-1

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.

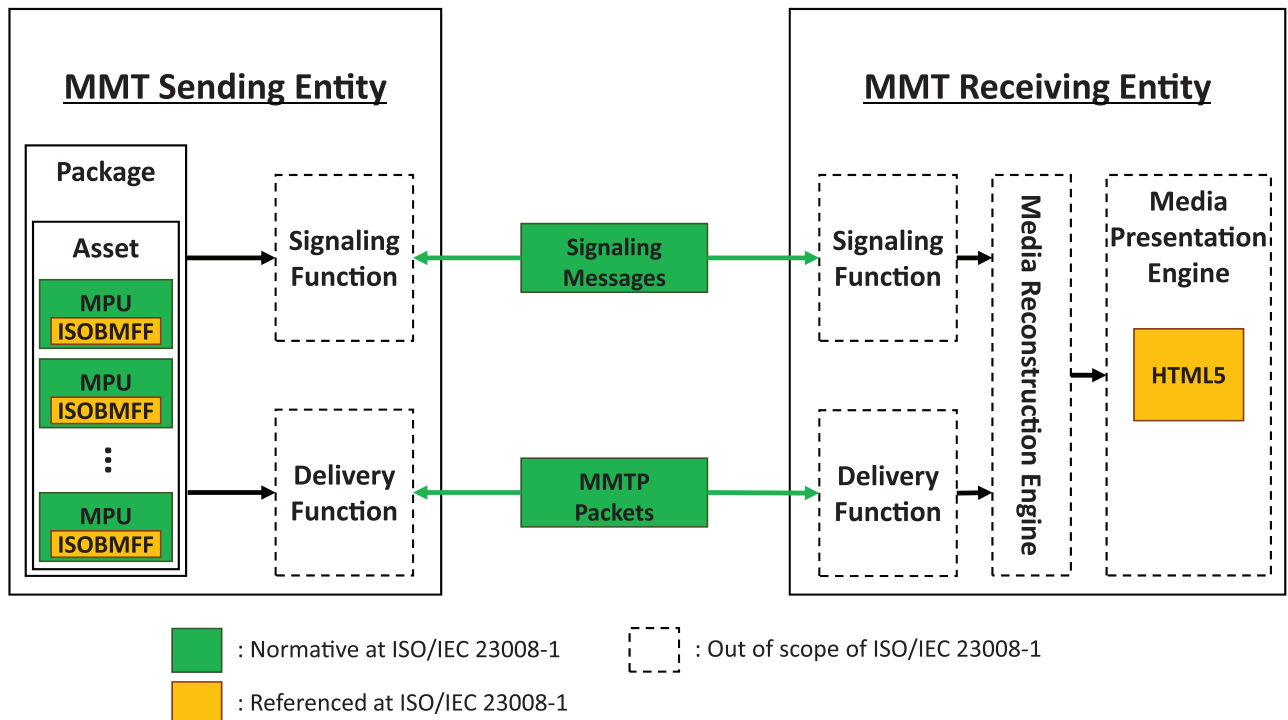


Figure 2 — MMT functions deployment

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

5 MMT function deployments

5.1 General

This clause gives implementation guidance on general MMT deployment based on basic functionalities provided by ISO/IEC 23008-1. In particular, it provides guidance on how to make best use of ISO/IEC 23008-1 for the basic topics such as, but not limited to;

- 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:

- 1) Receive MMTP packet.
- 2) Check if *packet_id* is equal to the *packet_id* of the object flow of interest, discard packet and go to step 1 if it does not belong to an object flow of interest.
- 3) Assert that type of the MMTP packet is MPU.
- 4) If fragmentation flags are set (different than '00').
 - a) If fragmentation flag is equal to '11', attempt to recover packet and if successful go to step 6.
 - b) Else add packet to the list of packet fragments based on the MMTP sequence number and goto step 1.
- 5) If aggregation flag A is set, extract all aggregated data units and proceed to step 7 for each extracted data unit.
- 6) If object map with same *MPU_sequence_number* does not exist, create new object map for the MPU with that sequence number.
- 7) Check fragment type (FT) of the MPU payload header.
 - a) If FT is MPU metadata:
 - i) Check if MPU metadata is already received:
 - 1) If yes, discard the MPU metadata as being a duplicate;
 - 2) Else insert MPU metadata at the beginning of the object map:
 - a. Optionally, forward MPU metadata to application.
 - ii Go to step 1.
 - b) If FT is fragment metadata:
 - i) Check if movie fragment with the same *movie_fragment_sequence_number* already exists:
 - 1) If no, create a placeholder for the movie fragment in the object map.
 - 2) Else, check if fragment metadata has already been received:
 - a) If yes, discard fragment metadata as being a duplicate;
 - b) Otherwise, insert fragment metadata at the beginning of the fragment placeholder.
 - 3) Go to step 1.

- c) 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:
 - 1) Insert payload in the fragment placeholder in the correct order based on the sample number and offset values.
 - 2) Check if movie fragment is complete.
 - a) If yes, forward fragment to the application.
 - 3) Go to step 1.
 - iii) If timed metadata flag is not set:
 - 1) Insert payload in the item in the object map based on the item *item_ID*.
 - 2) Recover item information from MPU metadata for the recovered item and forward the item to the application.
 - 3) Go to step 1.

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 7)c) 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.

An MFU may be fragmented by multiple fragments and the total number of fragments can be larger than the range which the fragment counter can present. It can be recognized by using the fragmentation indicator and the fragment counter. In this case, the fragmentation indicator (*f_i*) indicates that the received payload is neither the first nor the last fragment (which finally corresponds to value of '10'), but at the same time the fragment counter (*frag_count*) specifies that there is no succeeding MMTP payload (value of '0') containing fragments of same data unit. So the receiver can notice that the fragment counter is fully used but, after reaching 0, it will roll over and be reused. This is the way to provide information that an MFU is packetized by the larger number of fragments than the maximum number of fragment counter can present and to count that number. [Figure 3](#) describes the operation.

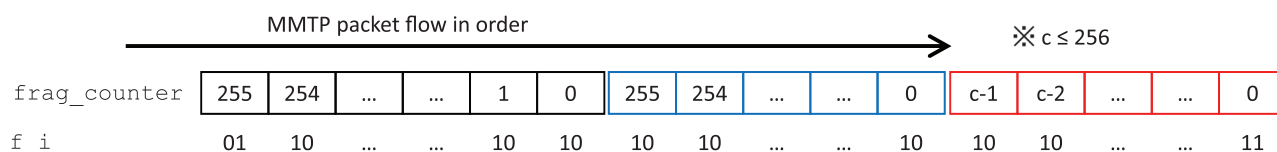


Figure 3 — Fragment counter and fragmentation indicator changes for 513~768 fragments

Figure 3 shows the changes of fragment counter and fragmentation indicator when an MFU is fragmented into '512+c' number of payloads ('c' is an integer less than or equal to the maximum number that fragment number can have. In this example, it is 256 because the size of fragment counter field is 8 bits). The fragment counter will change from 255 to 0 twice and for the last 'c' number of payloads it will start from 'c-1'. The fragmentation indicator will set to '01' for the first fragment, '11' for the last fragment, and set to '10' for the rest 'c-2' mid fragments.

5.2.3 Recovery in GFD mode

When operating in 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:

- 1) Receive MMTP packet.
- 2) Check if *packet_id* is equal to the *packet_id* of the object flow of interest, discard packet and go to step 1 if it does not belong to an object flow of interest.
- 3) Assert that type of the MMTP packet is GFD.
- 4) If object map with same TOI does not exist, create new object map for the file with that TOI.
- 5) Insert payload in the correct place in the object map using the *start_offset* information.
- 6) It is recommended that chunks of contiguous byte ranges that lie between 2 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.
- 7) If complete TOI is recovered:
 - a) Extract metadata from the transport object or from the GFD table.
 - b) 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 HTML5). It is 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 (ISO-BMFF) with certain restrictions. However, the usage of the ISO-BMFF may give the impression of high end-to-end delay, which may seem not suitable for live broadcast. This is, however, 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.

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. 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, an MPU should be completed to packetize it and send it to the client. However, in real implementation, there are ways to further optimize generation of an MPU and packetization of it to minimize delay by starting packetization and delivery of the MPU before completion of its generation (see [Figure 4](#)).

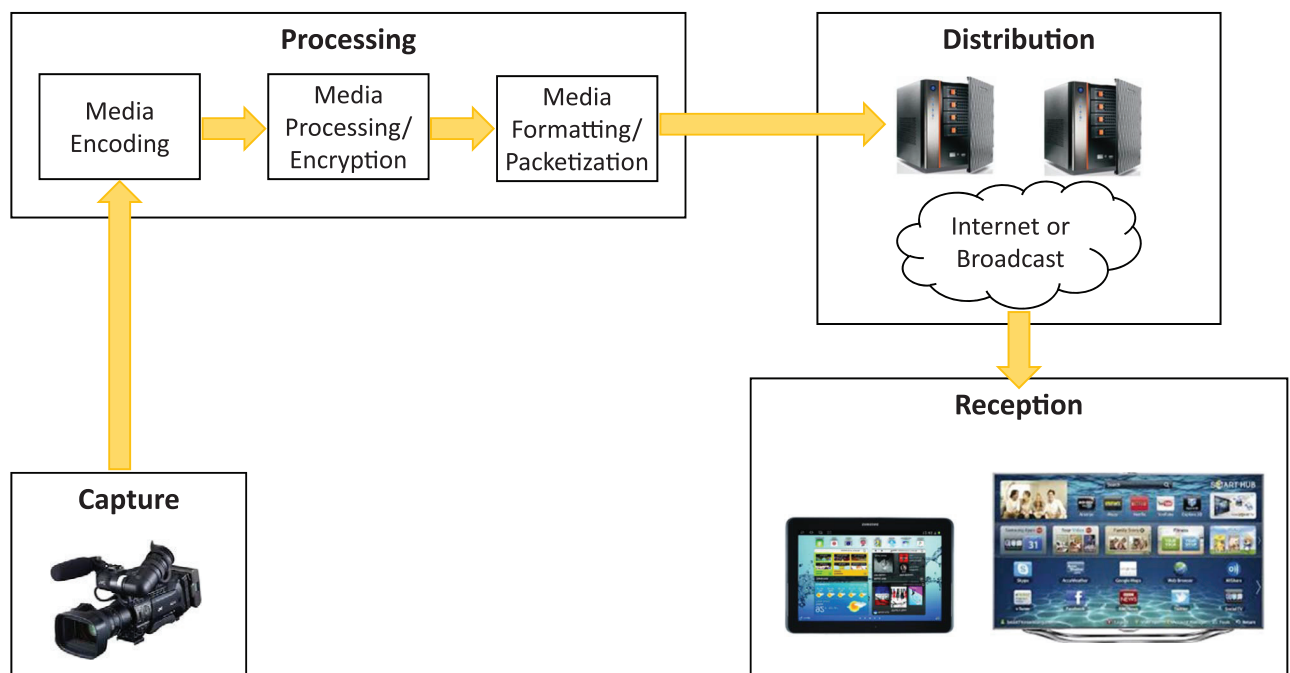


Figure 4 — Example of a broadcast scenario

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 is shown in [Figure 5](#).

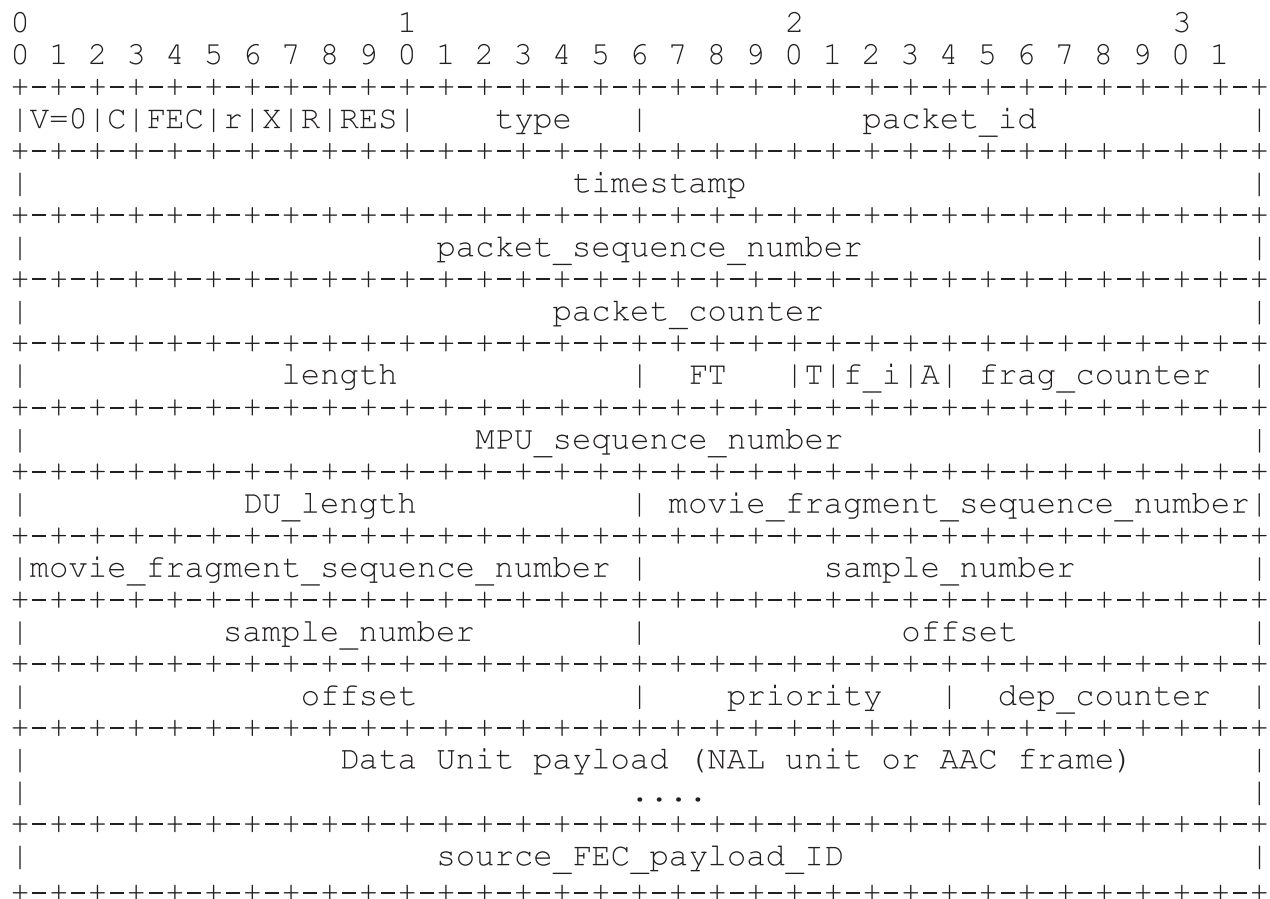


Figure 5 — MMTP packet

As can be seen from the packet header, all fields of the packet can be generated immediately at the streaming server, i.e., there is 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 already known 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 the start of encoding. Therefore, high priority can be assigned for the NAL Units containing intra coded macroblocks. The *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 the actual encoding process to improve encoding quality. However, in that case, the actual coding structure is known before the last path which generates the final compressed data. Therefore, the precise dependency structure can also be known before the 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 in advance 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 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 belong 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, that the broadcast application is using a very short duration for the MPU. In a broadcast application, the client needs to quickly find a starting point for decoding. To support it, a conventional broadcasting service repeatedly transmits initialization information for the decoder and uses 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, a delay can be maintained the same as that of conventional broadcast application.

MMTP allows for operation at very low end-to-end delay in a way that is suitable for 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

The MMT protocol performs parallel generation of MMTP flows and signalling message generation/processing (see [Figure 6](#)). The data packet processing part of the 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 the transport layer. The signalling message generation part of MMTP processes the CI, ADC and other data from the MMT package and encapsulates them into signalling messages, packetizing and passing them to the transport layer.

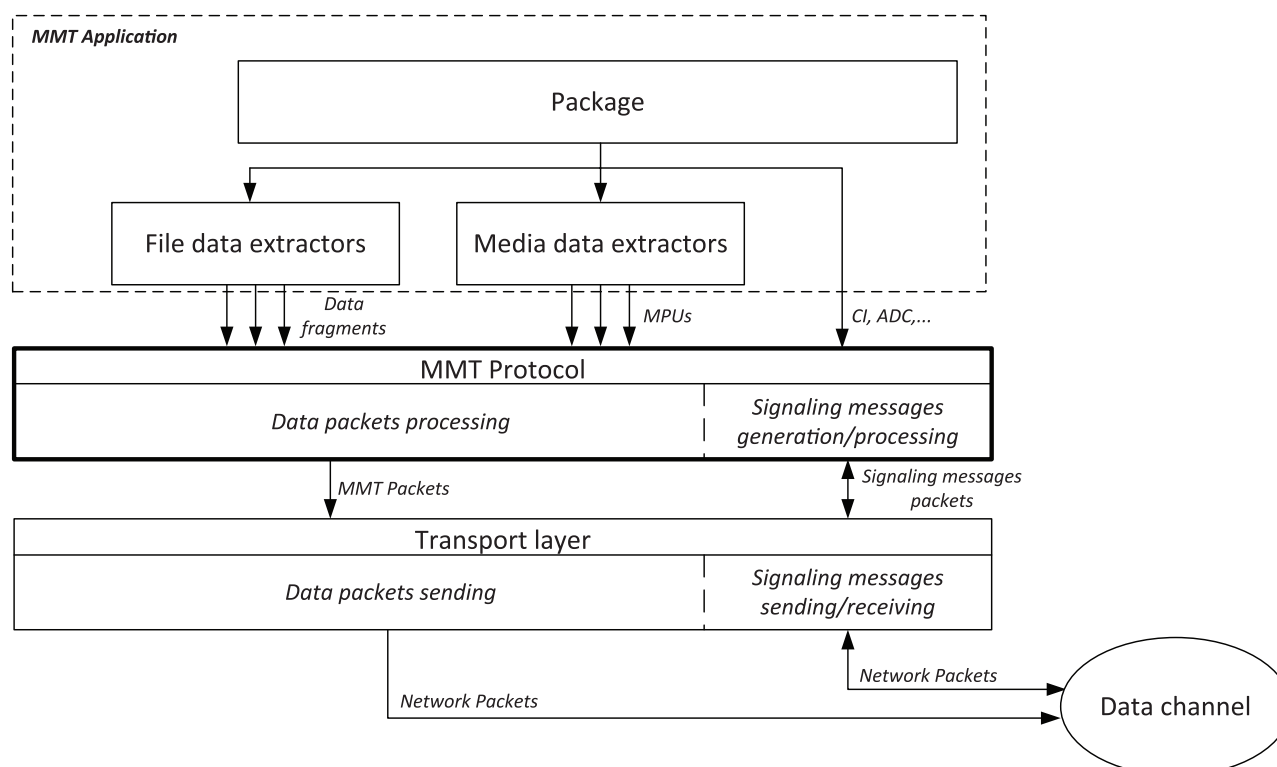


Figure 6 — MMT sending entity structure

A more detailed architecture of the data packet processing part of MMTP is presented in [Figure 7](#). MMTP packets are stored in separate buffers for each data flow after their processing with the help of a 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.

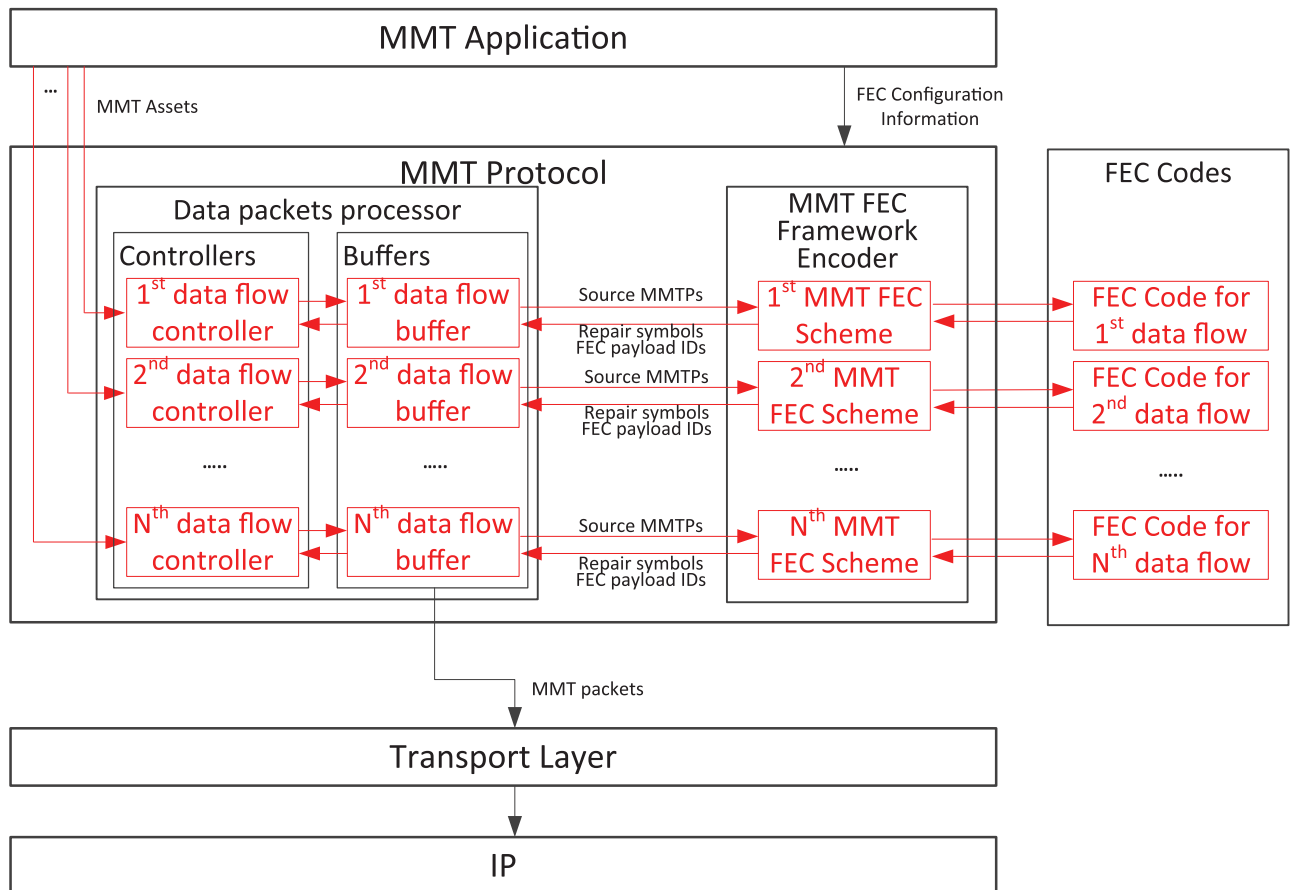


Figure 7 — Architecture for AL-FEC (MMT sending entity)

5.5.2 Processing in MMT receiving entity

The MMT protocol performs parallel processing of the MMT packet data flows and generation/processing of signalling messages (see Figure 8). The data packet processing part of the MMT protocol receives MMTP packets from the transport layer and transfers them into the corresponding data flow processor. Each data flow performs recovery of lost FEC source packets and then passes them for generic object reconstruction or/and MPU reconstruction, which happen in parallel. The signalling message receiving part of the 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.

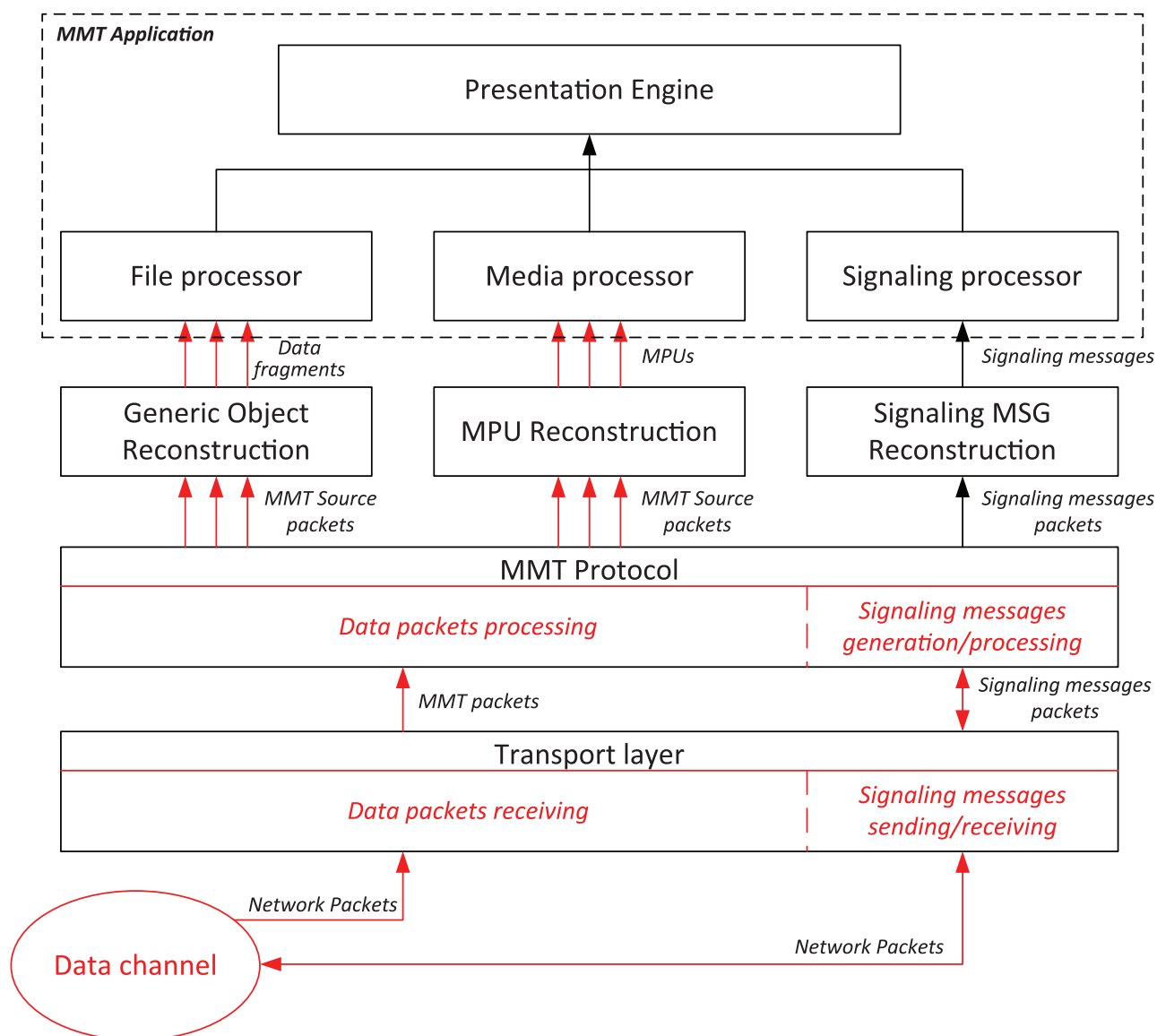


Figure 8 — MMT receiving entity structure

At MMT receiving entity, the 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 shown in [Figure 9](#).

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

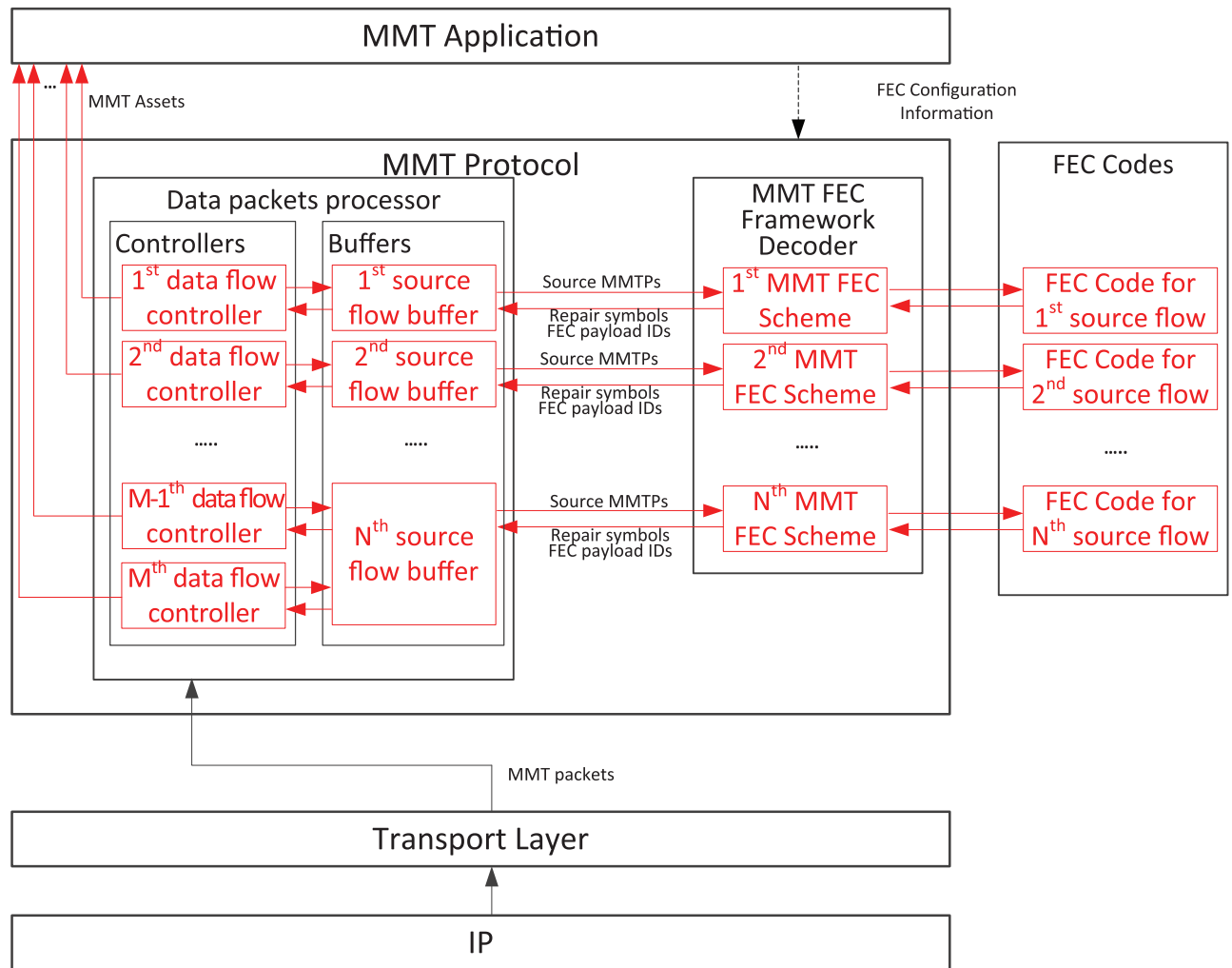


Figure 9 — Architecture for AL-FEC (MMT receiving entity)

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: timed data and non-timed data. The 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 10](#) 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 involves 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 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.

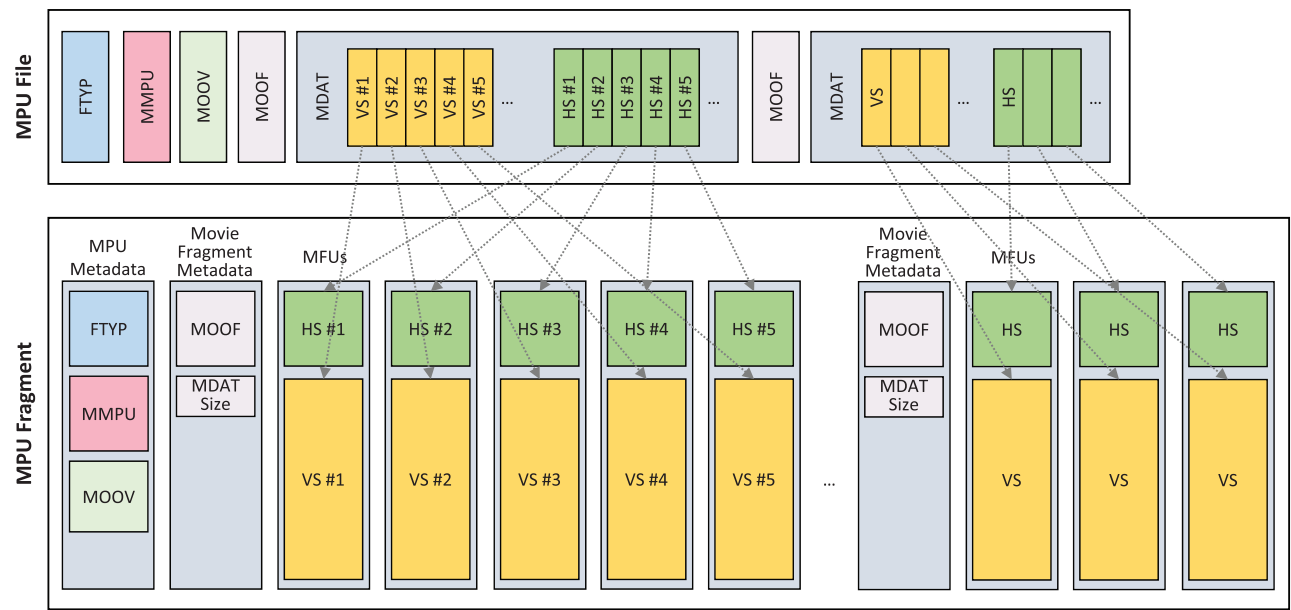


Figure 10 — MPU fragment building block

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 also has to 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 11](#).

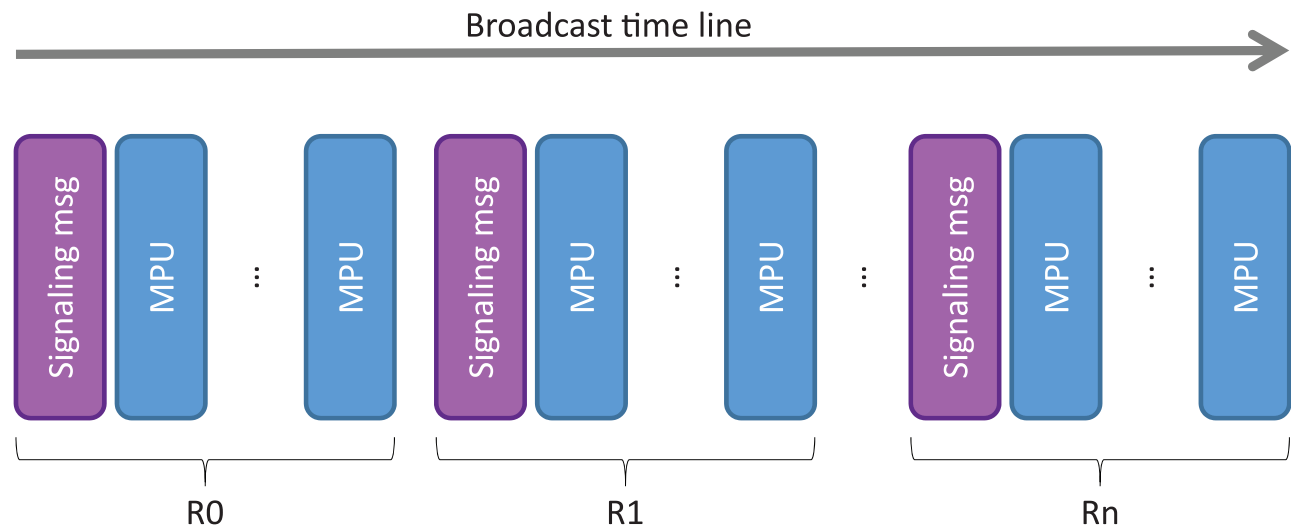


Figure 11 — Periodical signalling information transmission

5.6.1.2.2 The 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 an MPU. The fragmentation indicator contains information about fragmentation of the data unit in the payload, especially boundary information of the delivery data unit. When the MPU is a delivery data unit, the MMT receiving entity prepares the reconstruction of the MPU in the MMT delivery layer. The fragmentation indicator indicates MPU metadata is the first delivery data unit. The MMT receiving entity can wait to reconstruct the MPU until the fragment indicator which indicates the last of it is detected.
- The second case is that the delivery data unit is either kind 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 the 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, the frequency of sending the CI can be on a per-MPU basis.

However, there can be a time interval between consecutive MPUs in an MMT broadcast, which effectively corresponds to the length of MPU playing time (see [Figure 12](#)). Usually signalling messages are much smaller than MPUs. Therefore, the sending time of signalling messages is trivial compared to that of the MPUs. It means that when an MMT receiving entity randomly accesses the service, it is very highly probable that it will meet the MPU directly rather than the signalling message. In that case, the MMT sending entity can send the signalling message behind the MPU, to make use of those received MPUs.

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

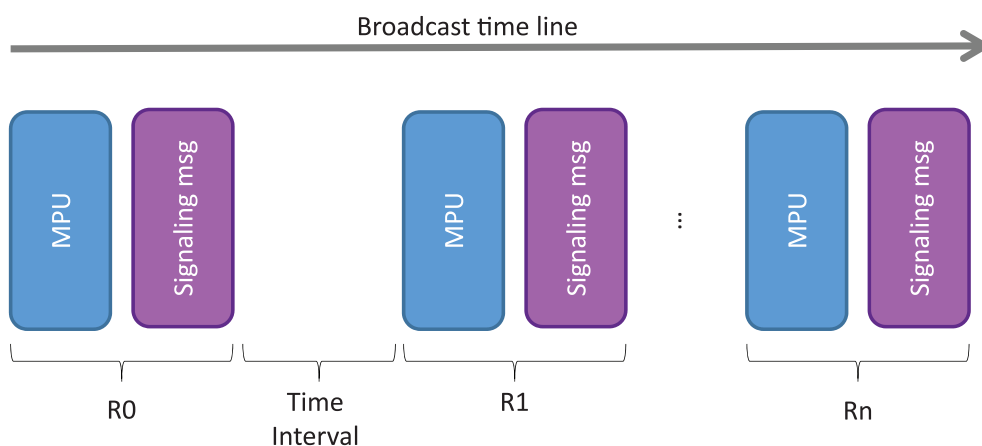


Figure 12 — Signalling message behind the MPU

5.6.2 Sending of MPU and signalling message

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

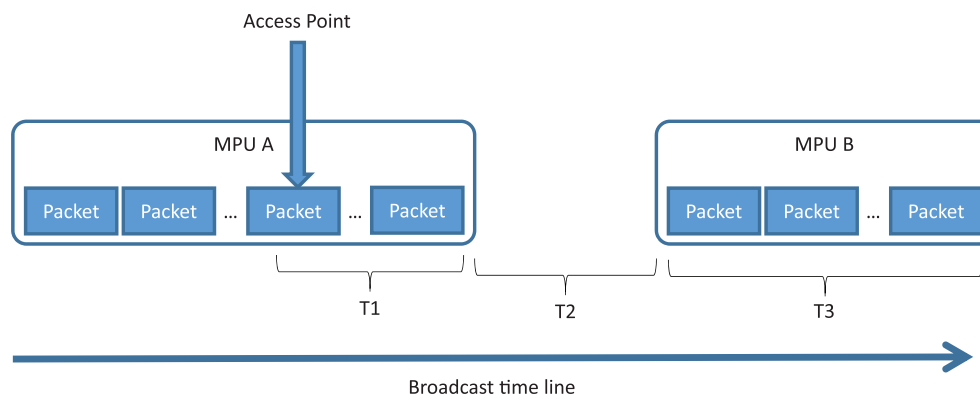
When the MMT receiving entity randomly accesses the broadcast channel, there are two cases: the MMT receiving entity accesses the channel while a certain MPU transmission is ongoing; or the MMT receiving entity accesses the channel during the time interval between consecutive MPUs.

Case 1:

In this case (see [Figure 13](#)), normally the first MPU received is not a 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 whole MPU to arrive. In that case, the expected wait time before media player initialization, Δt , can be calculated as:

$$\Delta t = t_1 + t_2 + t_3$$

where



- t_1 is the residual time to finish receiving the first MPU since the MMT receiving entity joins to the channel;
- t_2 is the time interval between two consecutive MPUs which is calculated by the length of MPU playing time;
- t_3 is the receiving time of the next MPU (because the signalling message sending time is very short compared to that of MPU, it may not be considered).

Figure 13 — 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, and MFUs containing at least one RAP (e.g., IRAP picture) should be received during t_1 , the wait time of t_2 and t_3 can be saved.

In MMT protocol, the payloads are generated according to [Figure 14](#).

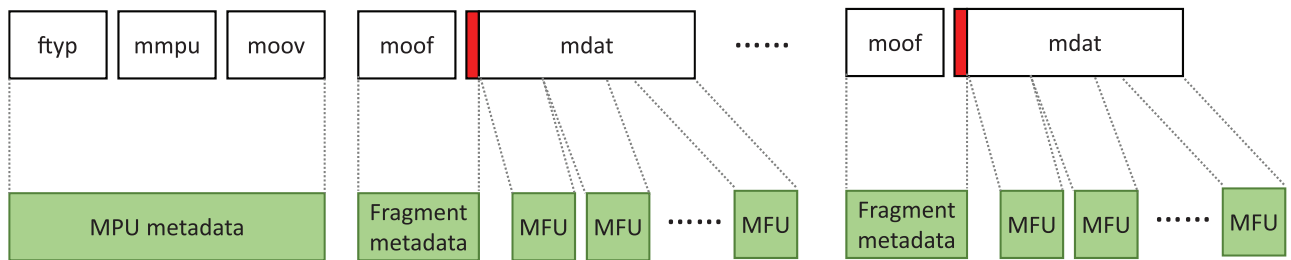


Figure 14 — MMTP payload generation

In this case (see [Figure 15](#)), only one moof is included in one MPU. So the MMT receiving entity can classify all packets in three categories: MPU metadata; fragment metadata; and MFU. MFU is considered as a (sub-) sample of media.

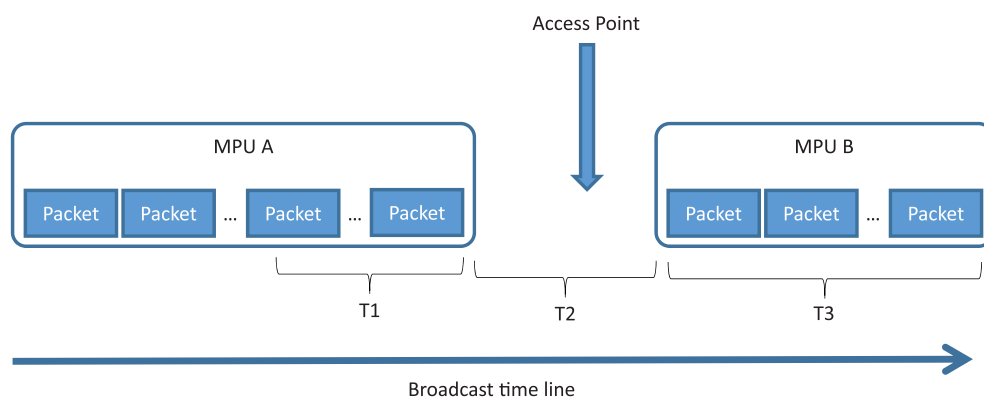


Figure 15 — Random access in the middle of interval (case 2)

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:

$$\Delta t = t_1$$

Case 2:

In this case the first MPU received should be a complete MPU. The total waiting time should be part of t_2 and whole t_3 . It should be:

$$\Delta t < t_2 + t_3$$

5.6.3 MPU generation for SHVC-encoded video in real-time streaming

5.6.3.1 General

Hybrid delivery may be a good solution to provide UHD services. Service providers (MMT sending entities) may provide a video service with various resolutions because they do not know the resolution of the user display. Therefore, the video sequence can be encoded by scalable video coding (i.e., SVC, SHVC) spatially, and the encoded video may have 2 or 3 layers depending on encoding parameters (i.e., base layer, enhancement layer 1 and enhancement layer 2). The following is the implementation guidance for MPU encapsulation to provide SHVC-encoded video in MMT.

5.6.3.2 MPU generating method for timed media with multi-layered video

The process of MPU generation is described as follows;

- 1) SHVC-encoded video stream should be extracted according to its own '*layer_id*' because each layer can be transmitted over different networks.
- 2) The extracted video streams are encapsulated into each of the MPUs and it has two tracks, i.e., a media track and a hint track.

a) media track

i) handler_type

When the '*hdlr*' box is present in the media box, '*handler_type*' is set to '*vide*'. The sample description box of this MPU has initialization information for video decoding.

ii) VideoSampleEntry

The VideoSampleEntry of the base layer has an '*hvc1*' or '*hev1*' box and the other layers (e.g., enhanced layers) have '*lhv1*' or '*lhl1*' (see ISO/IEC 14496-15:2019, 9.7.3). These boxes have encoding configuration, e.g., VPS, SPS and PPS. Decoder can get initialization information through these boxes.

b) hint track

i) has_mfus_flag

In general, it is set to '1' in case that '*handler_type*' is set to '*vide*' (see ISO/IEC 23008-1:2017, 7.1.2).

ii) timed_flag

When the media data hinted by this track is timed data, it is set to '1'.

c) MMTHSample: sequence_number, movie_fragment_sequence_number, samplenum

The values of '*sequence_number*', '*movie_fragment_sequence_number*' and '*samplenum*' of each MFU (for base, enhancement #1 and enhancement #2 layers) with different '*layer_id*' should be set to the same value for synchronization of the SHVC-encoded video (see ISO/IEC 23008-1:2017, 7.1.3). In this case, decoder can control the layered video because the MFUs have the same timestamp. The decoded video can be switched to the different scalable level according to '*layer_id*' selected by users (or decoder).

d) multiLayerInfo

i) multilayer_flag

If video is extended spatially or temporally, it is set to '0', and '*layer_id*' and '*temporal_id*' are used. Otherwise, it is set to '1' in case the video is encoded with other scalability.

ii) layer_id

If '*multilayer_flag*' is set to '0' and video data in this MFU has spatial scalability, this field is set as the same value of '*nuh_layer_id*' in the NAL header of the video sample in this MFU.

- 3) The NAL stream in the '*mdat*' box may have an IRAP picture.

5.6.3.3 Resolution-switching scenario

An SHVC-encoded video is extracted into three layers with spatial scalability, and each layer is encapsulated into MPU. The MPUs for base (HD) and the enhanced #1 (FHD) layers are delivered over

a broadcast channel, and the enhanced #2 (UHD) over a broadband network. [Figure 16](#) shows a test scenario.

1. The user can watch basically HD-resolution video at t_0 , and then the FHD service is played at t_1 if the user wants.
2. The user gets a UHD-resolution video service at t_2 , if the receiving entity can connect a HTTP session. At this time, the decoder requires parameter sets of all layers.
3. At t_n and t_{n+1} seconds, the network conditions are getting bad. The decoder decreases the video resolution to keep the video playing.



Figure 16 — Test scenario

5.7 Fast MMT session acquisition

In a multimedia streaming environment, a user switches between media of very large 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 (see [Figure 17](#)) for transmission.

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.

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. A RAP can be MPU metadata, a signalling message or frames containing i-frames.

The *type* indicates the type of payload data, i.e., MPU, generic object, signalling message. As an additional step, when an MMTP packet includes MPU data (*type*=0x00), the MMT receiving entity can also find out whether it has MPU metadata by checking the MPU fragment type (FT: 4 bits) in the payload header indicating the fragment type as defined in ISO/IEC 23008-1:2017, Table 4.

For fast access to the IDR frames after constructing MPU, the MMT receiving entity can be processed to parse the “moov” box in the MPU metadata, which indicates the presentation time for the sample, and to get the starting time from the *MPU_timestamp* descriptor or the begin time specified in in ISO/IEC 23008-11.

If the *RAP_flag* is set to “0”, it means that the corresponding MMT packet contains no high priority random access data. The MMT receiving entity can discard the MMTP packets without a *RAP_flag* set to “1” for efficient buffer management before the *RAP_flag* is set to “1”, or process it depending on its policy or MMT receiving entity situation.

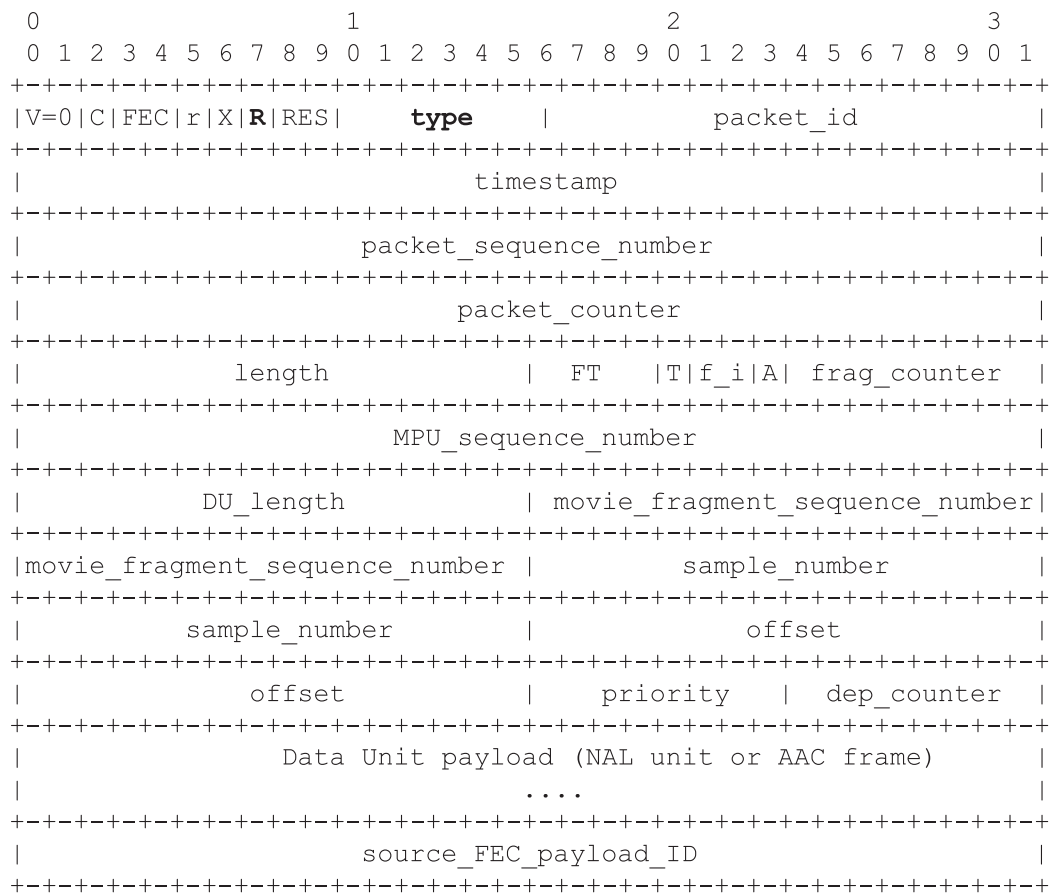


Figure 17 — Structure of MMTP packet

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 the 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 clause 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 designed to deliver MPUs, which are media files 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 an *'ftyp'*, *'sidx'*, *'mmpu'*, *'moov'* and *'moof'* box in MPU metadata. Among them, the *'mmpu'* box includes an 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 an MMT hint track providing the information to convert encapsulated MPU to MMTP payloads and MMTP packets.

The MMTP hint track provides two important parameters: *priority* and *dependency_counter*. The *priority* parameter 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 the MMT sending entity transmits media streams to an MMT receiving entity, it is possible that the MMT sending entity can intentionally skip some portion of it. For example, the MMT sending entity can know of a poor network condition or lack of resources on the MMT receiving entity side through some feedback. The MMTP packets will be dropped according to network conditions by MANE or other

network entities if there is no adaptation operation. In a streaming service case, omission of some media may not critically affect user experience.

The MMT sending entity can skip transmission of some MFUs to adapt to network conditions. 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. The MMT sending entity should drop MFUs with the lowest priority and high independence from other MFUs first by comparing *priority* and *dependency_counter*. For example, MFUs including RAP information will have a high priority value and MFUs corresponding to I-frame will have a *dependency_counter* value.

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

5.10 Hybrid delivery in MMT

5.10.1 General

This clause provides MMT implementation information when MMT assets are delivered on hybrid networks.

Hybrid delivery 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 network.

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 can be classified as follows.

- Live and non-live:
 - Combination of streaming components ([Figure 18](#)).
 - Combination of streaming component with pre-stored component ([Figure 19](#)).
- Presentation and decoding:
 - Combination of components for synchronized presentation ([Figure 18](#)).
 - Combination of components for synchronized decoding ([Figure 20](#)).
- Same transport schemes and different transport schemes:
 - Combination of MMT components.
 - Combination of MMT component with another-format component¹⁾ such as MPEG-2 TS.

1) In this instance, the case in which another-format component is encapsulated into MMT-format is excluded.

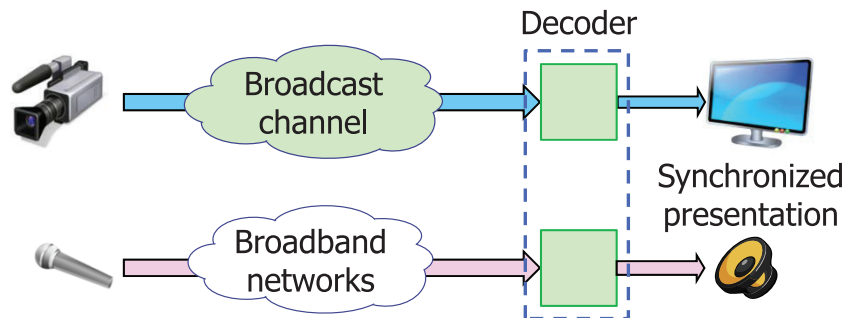


Figure 18 — Combination of streaming components for presentation

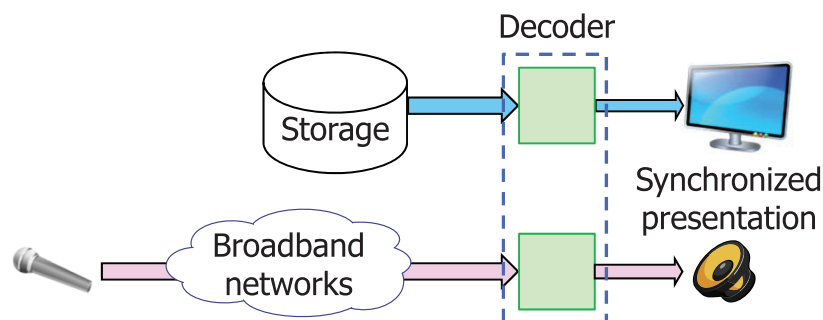


Figure 19 — Combination of streaming component with pre-stored component for presentation

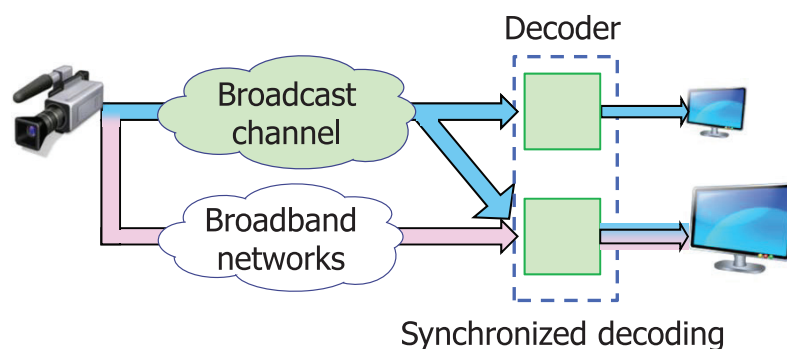


Figure 20 — Combination of components for decoding

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 information²⁾.
- Information on spatial relationships among media components.
- Signalling message for media consumption.

2) When an MMT component is combined with another-format component, the latter component may be neither an asset nor an MPU.

The following subclauses provide detailed information on the implementation of MMT for individual cases categorized in subclause [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

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.

5.11.1.1 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 the 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, the 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

5.11.1.2 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 a synchronized manner since both components can share the same time domain as a wall clock.

5.11.2 Use case: Combination of MMT and HTTP streaming for synchronized decoding

5.11.2.1 Asset information

It has to include all layer information to the signalling message.

- *MMT_general_location_information*

In the case of combination with HTTP on broadband networks, the URL location is used. The value of *location_type* in *MMT_general_location_information* is set to '0x05'. URL can indicate the address of MPD, and the list of segment location.

- *identifier_mapping*

When *MMT_general_location_information* specifies the URL for MPD, *identifier_type* is set to '0x03' and it means id in the 'Representation' element. It indicates the URL of a segment.

5.11.2.2 Information on spatial relationships among media components

It has to include all layer information to the signalling message.

— *dependency_counter*

It indicates the number of MFUs whose decoding is dependent on this MFU. The value of this field is equal to the number of subsequent MFUs in the order of *sequence_number* that may not be correctly decoded without this MFU. For example, if a scalable video is encoded two layers (one is base layer and another one is enhanced layer) and each layer is encapsulated MPU or segment following ISO-BMFF format then *dependency_counter* in *MMTHSample* of base layer is set to '1', and *dependency_counter* in *MMTHSample* of enhanced layer is set to '0'.

— *multilayer_flag*

When scalable video is used, it is set to '1', and the values of *dependency_id*, *depth_flag*, *temporal_id*, *quality_id*, *priority_id*, *view_id* and *layer_id* in 'MMTHSample' are set according to the results of video coding. If a scalable video has two layers, *layer_id* in base layer is set to '0' and *layer_id* in enhanced layer is set to '1'.

The asset descriptor in the MP table includes:

— *Dependency_descriptor*

It should be set according to *layer_id* of current asset.

5.11.3 Use case: Content request in advance for synchronized play-out

This subclause describes the case in which the transmission delay in a specific channel is too large or could not be fixed to a maximum value.

5.11.3.1 Application scenarios

When a program is transmitted in a channel (e.g., broadband channel) with very large or unpredictable transmission delay, the HRBM model cannot be used to ensure the synchronized play-out of the content. In this case the client could request the content in advance as long as it gets the content location information in *MMT_general_location_info()* and content accessible time period in *AT_descriptor()*, both of which are included in the MP table.

5.11.3.2 Content request procedure for synchronized play-out (see [Figure 21](#))

- If the available time of the content is not ready on the server side, the *AT_descriptor()* would not be included in MPT asset_descriptors.
- If the content is prepared with available time information, the descriptor would be delivered along with the MPT.
 - For the client that supports *AT_descriptor()*, it will read the *AT_descriptor()* and get the value of attributes 'location_index', 'available_begin' and 'available_end'. Then based on the *location_index*, the client may get the location information in *MMT_general_location_info()* in accordance with available time information. Thus, the client is able to request the asset content beforehand during the available time at the corresponding location.
 - For the legacy client that does not support *AT_descriptor()*, once they receive the *AT_descriptor()* with *descriptor_tag*=0x8000, it will ignore this descriptor as the tag is defined in the reserved fields. Since there is no available time provided, the client may try to request the content at the time when it thinks it's appropriate.

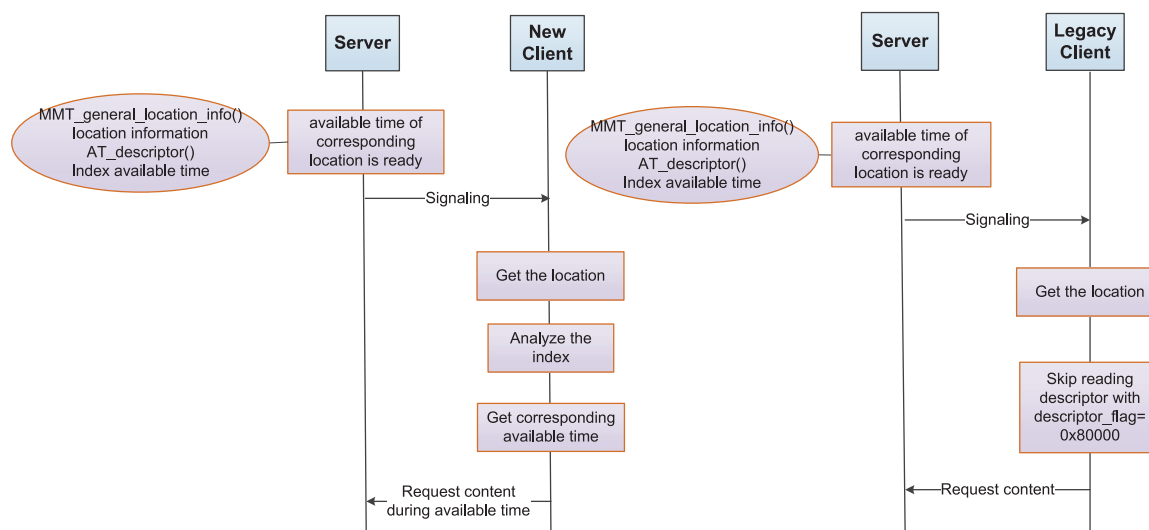


Figure 21 — Content request procedure

5.12 HRBM signalling for hybrid delivery

5.12.1 Hybrid delivery from the single MMT sending entity

As shown in Figure 22, 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.

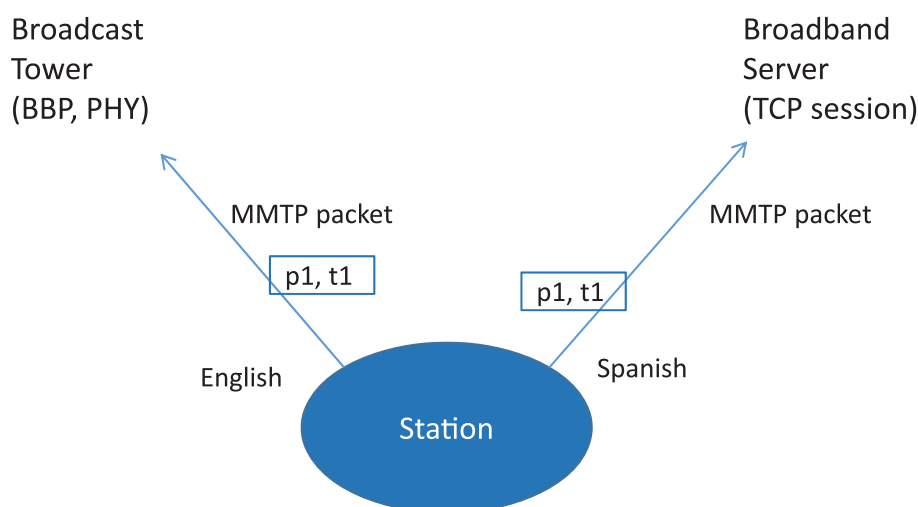


Figure 22 — Single station for hybrid delivery

The consideration of broadcast tower and broadband server is the transmitter of MMTP packets to distribute to the multiple MMT receivers.

The station can send the MMTP packets through the broadcast tower that contains the English audio streams for major broadcasting channels and can also send the MMTP packets with Spanish audio streams for secondary audio for the main channel via the broadband channel. In this case, the MMT receiver, the broadcasting tower and the broadcast server can receive the news program in the english or spanish audio by selecting their preferred alternative.

In this case, those MMTP packets are composed of each MPU that has same presentation time for rendering and sending the same time to leave from the station with the same timestamp of the MMTP packet header. Those MMT sending entities are synchronized based on UTC.

In MMT, the MMT receiving entity consists of the multiple HRBM buffers as shown in [Figure 23](#).

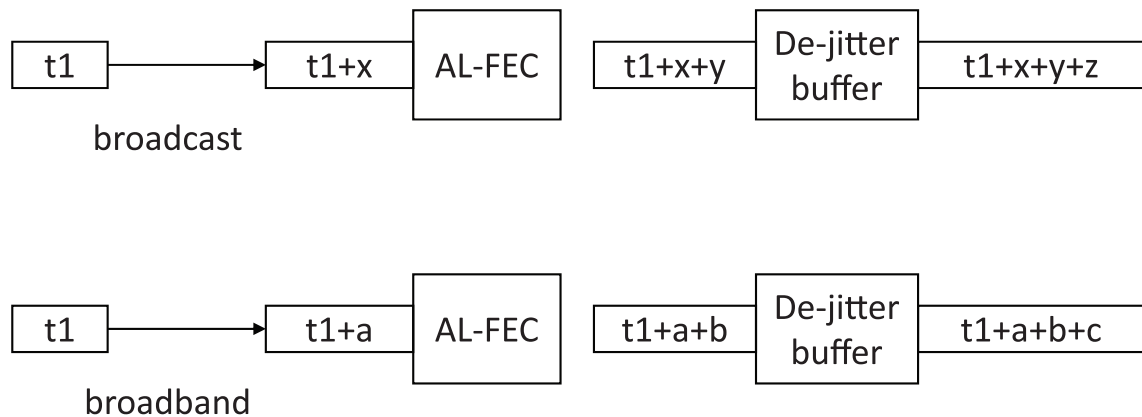


Figure 23 — HRBM buffers

Each HRBM has an AL-FEC decoding buffer and de-jitter buffer. An AL-FEC decoding buffer is used for recovering from the loss of MMTP packets in the network, and the de-jittering buffer removes the jitter of MMTP packets from the network. The goal of the HRMB structure is to provide the constant delay of the delivery system of the MMT receiving entity for synchronized play-out of the media data.

However, the lose and jitter of the MMTP packets depends on their delivery channel conditions. In this example, x is the delay of the network, y is the time for AL-FEC processing, and z is the time that the MMTP packet remains in the de-jittering buffer in the broadcasting channel. The broadband channel has also lost their MMTP packet, with a different arrival time of MMTP packets and corresponding de-jittering buffer time in the broadband channel.

For synchronized play-out of the media in hybrid delivery, the value of $t_1 + x + y + z$ needs to be equal to $t_1 + a + b + c$.

The MMT asset corresponding to the content set by each component, the play out delay (D), is determined by the value of the MMT sending entity via a signalling message to the MMT receiver. For example, the value of play out delay is the transmission network delay incurred by the characteristics of the maximum value and AL-FEC protection window time and can be calculated as the sum of:

$$D = \max(x_1, x_2, x_3, \dots) + \text{AL-FEC protection window time}$$

Here, x_1, x_2 and x_3 correspond to the nature of the transmission network delay including the propagation delay specified from *max_transmission_delay* in the HRBM message, the AL-FEC protection window time and the AL-FEC decoding processing is performed on the window interval (i.e., the AL-FEC protection window time of the AL-FEC encoding and decoding unit that performs the FEC packets corresponding to the block is the first of the packets including the transmission time of the packet being sent and the FEC packets of a packet block containing). Finally, a transmitted MMTP packet including the AL-FEC and transmission delay time is defined as the maximum value.

The "fixed end-to-end delay" is specified as the maximum value in the HRBM signalling message for the hybrid delivery network via multiple delivery paths to the MMT receiver for MMT receiving packets from the MMT sending entity within the same service that corresponds to the MMT receiver's output timing.

5.12.2 Hybrid delivery from the multiple MMT sending entities

[Figure 24](#) shows the multiple stations used for the MMT services. Station 1 is composed of the broadcast tower and station 2 has at least one of the broadcast servers that are provided to any broadcasting service using the hybrid delivery.

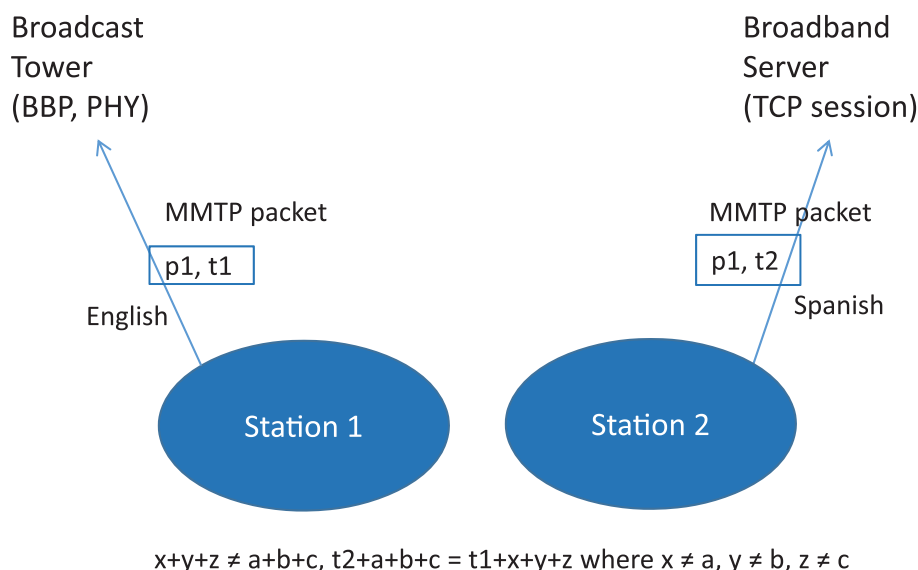
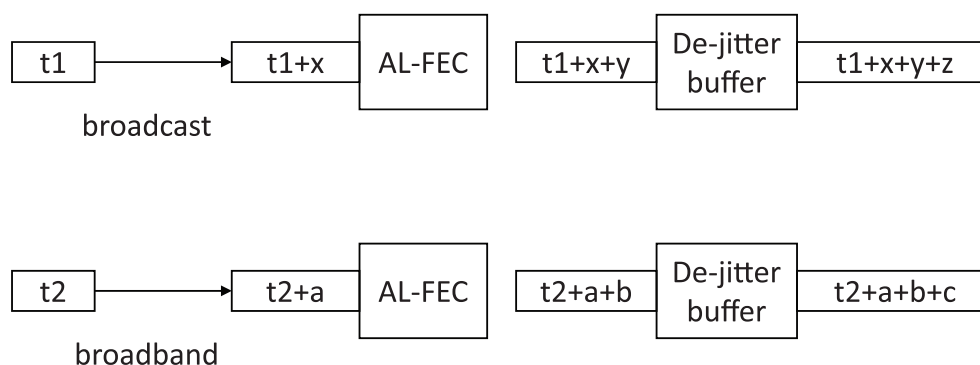


Figure 24 — Multiple stations for hybrid delivery

This also considers that the broadcast tower and broadband server is the transmitter of MMTP packets to distribute to multiple MMT receivers.

In this case, the presentation time of the MMTP packet is the same, but the sending of the MMTP packet is different (see Figure 25).



$x+y+z \neq a+b+c$, $t2+a+b+c = t1+x+y+z$ where $x \neq a$, $y \neq b$, $z \neq c$
 $t1$ for broadcast = time when the MMTP packet leaves MMTP layer
 $t2$ for broadband = time when the MMTP packet leaves MMTP layer

Figure 25 — Multiple HRBM buffers

The 'fixed_end_to_end_delay' is pre-defined as the time to consider the delay including propagation delay time between the MMT sending entity and the MMT receiving entity. The 'fixed_end_to_end_delay' is calculated by the summation of the 'max_transmission_delay' and 'FEC_protection_window_time'.

The HRBM message as described above is based on the structure of the first use case according to the maximum transmission delay defined value for each of the MMT receiving entities to the de-jitter buffer; the play-out of the time can be adjusted:

$$T_{de_jitter_out_time} = t_s + \alpha$$

Here, t_s is timestamp of the received MMTP packet in the MMT receiver. Wherein the timestamp can be calculated by the MMTP packet sending timing for the transmission delay. Then, 'fixed_end_to_end_delay' can be defined as a value in HRBM message. Based on t_s and α which are calculated by max_transmission_delay corresponding to network B, in this case, α is corresponding to the fixed_end_to_end_delay, the MMT receiving entity can control their playout time for synchronization.

In the above example, the delay of transmission A is relatively small compared to B. The MMT sending entity sends to the initial setting in the broadcasting channel, and the MMT receiving entity can analyse the network transmission delays and the jitter in transmission network B is larger than A. Here, 'max_transmission_delay' as specified in the HRBM message by the service provider corresponds to this transmission characteristic of B in the hybrid delivery network. The HRBM message may be sent periodically or sent by component content that may be transmitted by a specific event.

Consequently, the de-jitter buffer of A and the de-jitter buffer of B, according to the fixed end-to-end delay specified in HRBM signalling message, need to be specified with max_transmission_delay and can control the play out time which has the same value between the final output ($P: 't_1 + x + y + z$) of de-jitter buffer of A, and the final output ($Q: 't_2 + a + b + c$) of de-jitter buffer of B.

5.13 Error resilience in MMT protocol

MMTP is optimized for the delivery of MPUs, which are ISOBMFF files. The delivery of the MPU is performed movie fragment by movie fragment, thus enabling fast startup delays and fast access to the content. MMTP defines 3 different payload fragments for error resilience purposes:

- **MPU metadata:** this information contains the metadata of the ISOBMFF file and the MPU. The MPU metadata thus contains all codec configuration information and is crucial for the consumption of the whole MPU. The MPU mode allows marking the packets of the MPU metadata (usually only one or a few packets) so that the client can clearly identify them and recognize if it has received it correctly. To provide for random access and enhance the probability of receiving the MPU metadata, the sender should send the metadata repeatedly and periodically throughout the transmission time of that MPU.
- **Fragment metadata:** this information contains the “moof” box and the skeleton of the “mdat” box. This metadata provides information about the sample sizes and their timing and duration. This information is important for all media samples of the current fragment. However, it may be possible to recover from loss of fragment metadata and it is also possible to send it out of order. The sender may deliver the fragment metadata repeatedly and interleaved with the packets that contain the media samples of that fragment to increase the probability of correct reception and to enable random access inside a movie fragment.
- **MFU:** this contains a media unit from a sample of a particular movie fragment. The MFU also provides enough information such as the sample number, the fragment sequence number, and the position inside the media sample to position the media unit on the timeline and inside the “mdat” box. It may also contain information about the importance of that media unit for the decoding process. Based on that information, the sender, as well as intermediate MMT entities, may undertake appropriate steps to enhance error resilience respective to the priority and importance of the media unit. A media unit from a SAP is, for instance, more important than a media unit for which there are no dependencies.

One of MMTP's advantages is its ability to enable error robustness at the receiver side by enabling the client to recover from packet losses and still generate a compliant MPU when needed.

When the MPU metadata is lost, the client should keep any correctly received data from that MPU until a new copy of the MPU metadata is correctly received.

When a fragment metadata is lost, the client should use information from previous fragments about the sample durations to correctly reconstruct the lost “moof” box. It uses information from the received MFUs to recover the movie fragment segment number. The offsets of the media data may be recovered later using the start of a fragment as the baseline and the sample number and MFU sizes to reconstruct the “mdat” box as well.

When an MFU is lost, the loss can be discovered at the receiver based on the gap in the sequence numbers. The missing MFU is replaced by a null value array in the “mdat” box, if at least one MFU of the same media sample has been received correctly. If the complete sample is lost, the space occupied by that media sample may be removed completely and the information in the containing chunk, the “trun”, should be edited appropriately to adjust the sample duration of the previous sample and the sample offsets of the following samples. [Figure 26](#) shows an example of such processing.

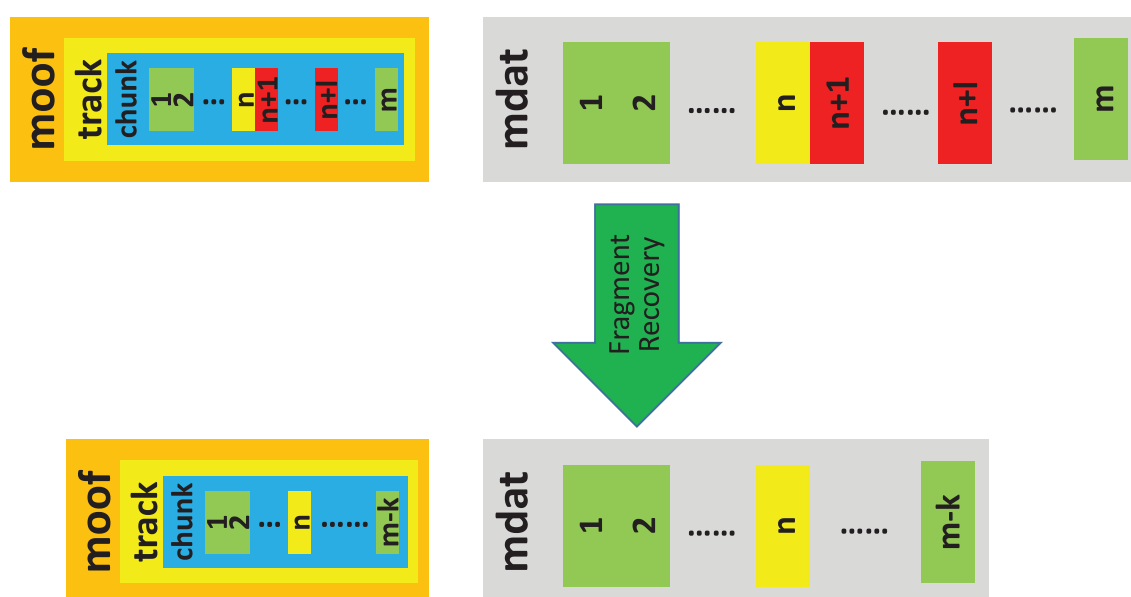


Figure 26 — Fragment recovery after packet loss

5.14 Delay constrained ARQ

5.14.1 General

This subclause provides methods on how to recover lost packets in a delay-constrained environment through ARQ. Delay constraints can be considered both on a delivery-time basis and arrival-deadline basis.

5.14.2 Delivery-time constrained ARQ

MMT employs an ARQ mechanism as an error control technique for recovery of packets which are lost during MMTP packet delivery. When an MMT receiving entity detects lost MMTP packets, it can request retransmission of those packets.

In timed media application cases, retransmitted packets which cannot arrive before play-out of media will be useless because they cannot be utilized for timely media recovery. The *arrival_deadline* is suggested as the maximum tolerable latency for the requested packets in an AF message. However, *fixed_end_to_end_delay* in HRBM message which already MMT sending entity knows can be used by a server to decide whether they will retransmit requested packets to the MMT receiving entity or not.

HRBM is used to ensure effective MMT operation under a fixed end-to-end delay and limited memory requirements for buffering of incoming MMTP packets. Especially in the de-jitter buffering phase of HRBM, the MMT receiving entity should remove MMTP packets that experience a transmission delay larger than the *fixed_end_to_end_delay* or *max_transmission_delay*. It means MMTP packets will be discarded before it is passed to de-capsulation phase into MPU or MFU, if it experiences more than that delay.

When an MMT sending entity receives an AF message requesting the lost packets, it should decide whether it will retransmit the requested packets or not based on *fixed_end_to_end_delay* in HRBM. To do that, it can use the most updated *propagation_delay* value included in the AF message and calculate expected delivery time of the MMT packets from the MMT receiving entity to the MMT sending entity by subtracting the timestamp in the packet header from the NTP time at the arrival instant of the AF message.

When retransmitted packets are expected to arrive at the MMT receiving entity eventually within *fixed_end_to_end_delay* since its the first original transmission, it will decide to transmit the requested packet, otherwise it will give up on retransmitting it.

5.14.3 Arrival-deadline constrained ARQ

5.14.3.1 Basic operation of the arrival-deadline constrained ARQ

Figure 27 shows the basic operation of the arrival-deadline constrained ARQ. In Figure 27, the arrival deadline denotes the maximum tolerable latency for the requested retransmission packet to arrive at the receiver. If the retransmitted packet arrives later than arrival deadline, even if the packet in fact arrives, it is regarded as useless and discarded.

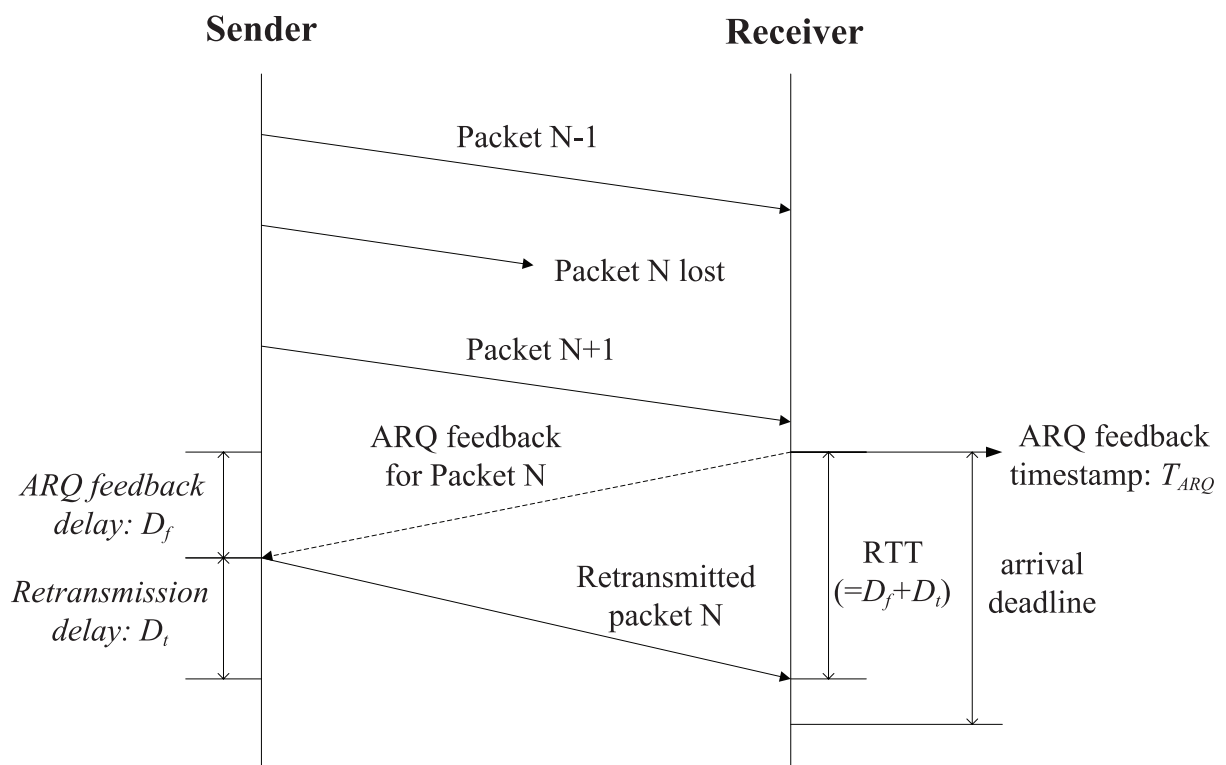


Figure 27 — Operation of the arrival-deadline constrained ARQ

Based on the above analysis, when the receiver detects the loss of a packet, the receiver sends ARQ feedback to the sender, and then the sender performs the following arrival-deadline constrained retransmission decision:

```

If (RTT < arrival_deadline)

    Sender retransmits the requested lost packet to the receiver;

Else

    Sender decides not to transmit the requested lost packet to the receiver;
    
```

In [Figure 27](#), the RTT (round-trip time) can be obtained at the beginning at the sender side by using the reception quality feedback (RQF) message. For more accurate estimation of the RTT, an up-to-date ARQ feedback delay (D_f) value can be provided to the sender by enclosing an ARQ feedback timestamp (T_{ARQ}) in the ARQ feedback message. Using the ARQ feedback timestamp, T_{ARQ} , the sender can obtain up-to-date ARQ feedback delay (D_f). And this updated D_f value can be used to compute up-to-date RTT.

5.14.3.2 Example of calculating arrival-deadline

There can be various ways to calculate the *arrival-deadline* depending on the service scenarios and preference of the service provider. The *arrival_deadline* can be obtained at the receiver side by considering the remaining amount of safely arrived packets in the receiver buffer when the ARQ feedback message is prepared for sending. [Figure 28](#) shows an example of calculating the *arrival_deadline* value at the receiver.

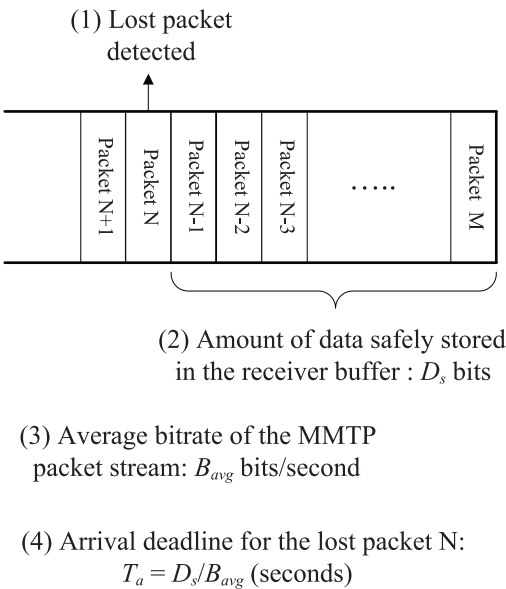


Figure 28 — Example of calculating *arrival_deadline* at the receiver

After detecting packet loss, the amount of data safely stored in the receiver buffer, D_s , is obtained. If some packets before the current lost packet N were also lost, D_s can be calculated considering the successive packets that are safely received prior to and following the previous lost packet from an affirmative point of view. Then, by using the average bitrate of the MMTP packet stream, arrival deadline for the lost packet N, T_a , is calculated as follows:

$$T_a = D_s / B_{avg} \text{ (seconds)}$$

In [Figure 28](#), with the available amount of packets in the buffer, the client can continue the presentation roughly for the duration of T_a (arrival-deadline). The requested retransmission packet should arrive until the arrival-deadline T_a .

5.15 Delivery of encrypted MPUs

Common encryption relies on different types of metadata for enabling decryption. The “*pssh*” box carries DRM specific metadata that is opaque to common encryption. The “*pssh*” box may appear inside the “*moov*” box or any “*moof*” box. Default track encryption metadata is provided in the “*tenc*” box and provides default metadata for all samples of the track. This information may be overwritten by encryption metadata for sample groups, which are provided as part of sample descriptions of the corresponding track. Finally, sample-specific encryption metadata can be provided as part of sample auxiliary information, which is either stored as part of the “*mdat*” box and pointed at by “*saiz*” and “*saio*” boxes for size and location respectively, or and as of in ISO/IEC 23001-9, sample encryption metadata will be provided as part of the “*senc*” box in track fragments “*traf*” boxes of movie fragments.

When delivering a common encryption encrypted MPU, all sample auxiliary information, whether stored as part of the “*tenc*”, “*senc*”, or the “*mdat*” is considered as part of the fragment metadata. In the latter case, the auxiliary sample information should appear in the beginning of the *mdat* box prior to any media samples. [Figure 29](#) shows how packetization occurs.

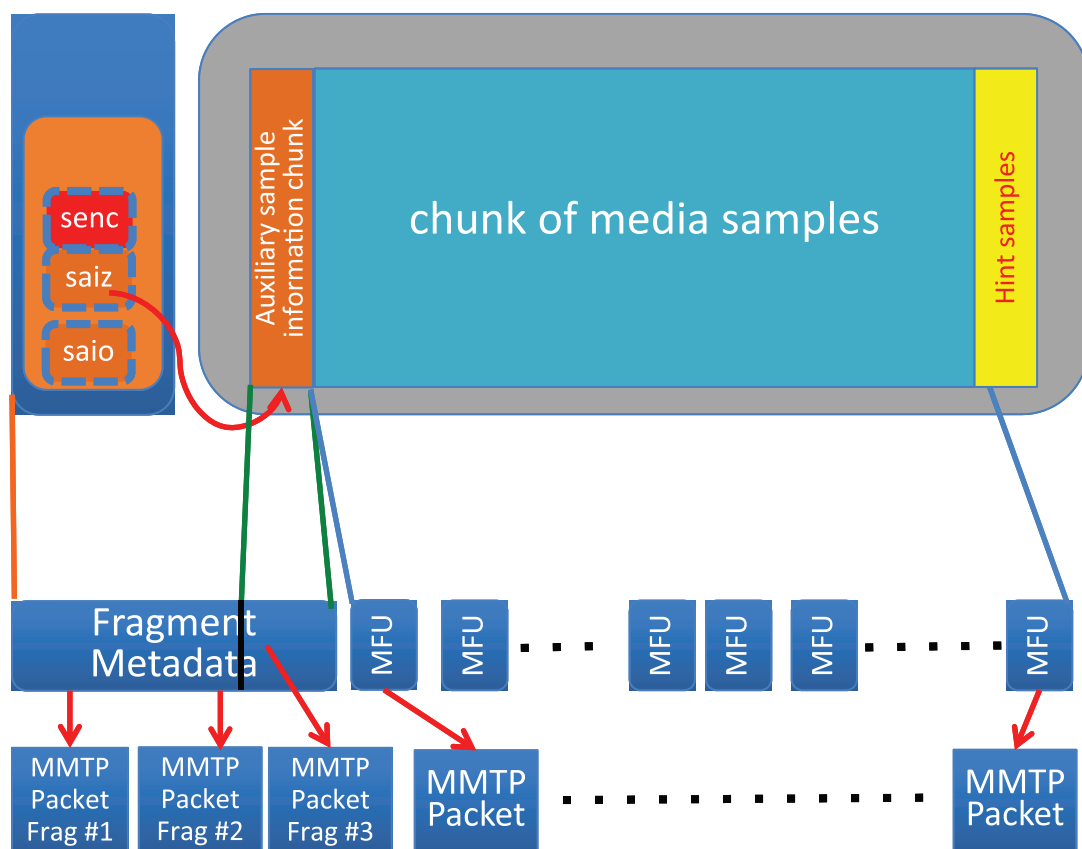


Figure 29 — Packetization considering encryption

The fragment metadata including the part in the “*mdat*” box is treated as a single payload. The receiver uses the offset and size information in the “*moof*” box to reposition the MPU metadata again in the “*mdat*” box when reconstructing the movie fragment.

5.16 HRBM message updating

5.16.1 General

The hypothetical receiver buffer model (HRBM) guarantees that each MMTP packet has a fixed end-to-end delay although it was delivered through a network with jitter. An MMT sending entity determines the required buffer size and the fixed end-to-end delay for each MMTP sub-flow and signals this

information to an MMT receiving entity using the HRBM messages. Then MMT sending entity transmits MMTP packets with the schedule verified by the HRBM to ensure no buffer overflow in the MMT receiving entity.

The required buffer size is calculated by using the maximum duration of the buffer, which can be obtained by subtracting the minimum transmission delay from the fixed end-to-end delay, and the maximum bitrate of the MMTP packet stream. In some applications, the MMT sending entity can temporarily change the maximum bitrate of the MMTP packet stream. In that case, the HRBM message should be updated based on the new maximum bitrate of the MMTP packet stream.

5.16.2 HRBM message sending schedule

When updating the HRBM message, the MMT sending entity should assume that an MMT receiving entity will prepare another de-jitter buffer which operates according to the updated HRBM message and passes the MMTP packets transmitted after the MMTP packets containing the updated HRBM message to the new de-jitter buffer (it can be done by checking the value timestamp field in the MMTP packet header). It is recommended that the updated HRBM message should be transmitted between consecutive MPUs in order to prevent unexpected delay in the MMTP packet decapsulation buffer.

5.16.3 Use case

Consider a service supporting a synchronized playout at multiple devices where each device is connected to the server using individual unicast connection. It can be assumed that the server has the same number of logical MMT sending entities as the number of devices connected to the server. In order to support a synchronized playout, the MMT sending entities can simultaneously transmit the same MPU to each MMT receiving entities using the same transmission schedule based on the same fixed end-to-end delay and the buffer size. When a new device has been joined to the service, the corresponding MMT sending entity is instantiated. If the device is joined in the middle of an MPU sending duration, the MPU cannot be delivered to the device completely. In order to reduce the service access time, the MMT sending entity can quickly transmit the whole MPU to the newly joined device if possible (burst mode) and then it transmits the next MPU using the same schedule with the other MMT sending entities in the server (normal mode).

For more detailed analysis, the following parameters are defined for normal mode:

- MPU duration: D seconds
- The maximum bitrate of the MMTP packet stream: M bytes/s
- The maximum number of bytes required to deliver a MPU: $M \cdot D$ bytes
- Fixed end-to-end delay: F seconds (the minimum transmission delay = d)
- The required buffer size: $B = (F - d) \cdot M$ bytes

Assume that the device has been joined after J ($< D$) seconds from the start of the delivery of an MPU at other MMT sending entities in the server. Then the maximum bitrate of the MMTP packet stream in the burst mode, M' , is given by:

$$M' = (M \cdot D) / (D - J).$$

As a result, the required buffer size for burst mode, B' , is increased compared to required buffer size for normal mode as follows:

$$B' = (F - d) \cdot M' = (F - d) \cdot M \cdot D / (D - J) = B \cdot D / (D - J)$$

$$= B + B * J / (D - J).$$

Then the MMT sending entity can determine whether it can operate in burst mode or not considering the available bandwidth and the above parameters. If it is possible to operate in burst mode, the MMT sending entity signals the values of M' and B' by HRBM message and sends the first MPU in burst mode. After the MMT sending entity has finished the delivery of the first MPU in burst mode, it operates in normal mode. Before it transmits the MMTP packet containing the second MPU, the values of M and B should be signalled to the MMT receiving by updating the HRBM message. If the HRBM message is not updated properly, then $B * J / (D - J)$ bytes of the memory in the MMT receiving entity is wasted.

5.16.4 HRBM buffer operation in unicast environment

An HRBM mechanism is used to ensure stable buffer management at the receiving entity during media packet transmission. Through the HRBM, it is ensured that every packet undergoes the same delay when it is delivered to the media player inside the client, i.e., *fixed_end_to_end_delay* (Δ). In a broadcast environment, that value is usually determined considering the receiving entity which has the longest delay to ensure safe reception at every receiving entity. However, as media transmission is moving toward IP-based unicast environments more and more, it is probable that real-time-ness can be better improved more by setting a smaller *fixed_end_to_end_delay* value. For example, transmission environment characteristics such as network jitter and delay between sender and receiver are able to be measured. Then, *fixed_end_to_end_delay* (Δ) can be as small as possible while also guaranting smooth packet reception.

Figure 30 shows the current HRBM buffer model defined in ISO/IEC 23008-1. It is comprised of an MMTP pacing buffer (optional), FEC decoding buffer (optional), de-jitter buffer and de-capsulation buffer. And HRBM signalling message is sent to set the operation parameter of buffer in MMT receiving entity. In an IP-based environment, every receiving entity may experience more varied network channel environments than in the broadcast case.

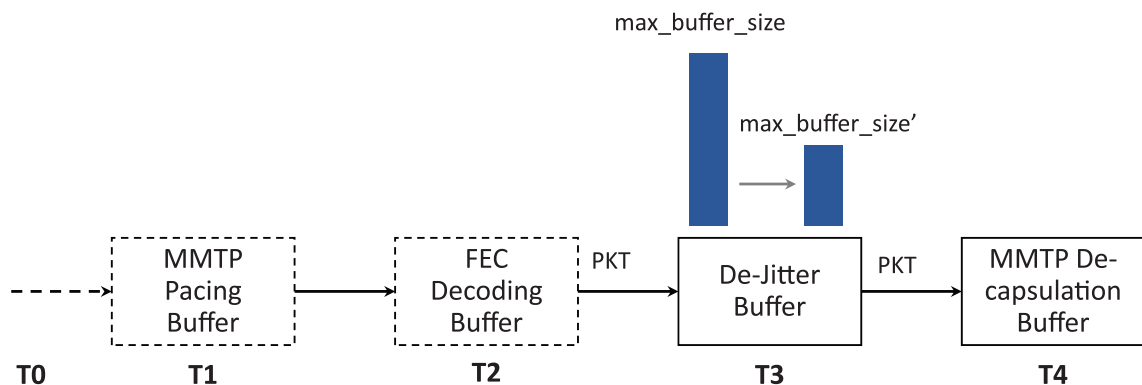


Figure 30 — HRBM buffer model in MMT

Figure 31 shows the timing model diagram of HRBM in MMT. If the time when the MMT packet is out of the de-jitter buffer (de-jitter buffer out time, $t_s + \Delta$) is decreased, the presentation time of media also can be reduced. It means media can be rendered faster if *fixed_end_to_end_delay* is configured, namely Δ , as small as possible, which directly results in improvement of real-time performance.

In Figure 31, T_0 means network delay time that the MMTP packet suffers, and that information value can be acquired from the *propagation_delay*, *min_transmission_delay* or *max_transmission_delay* parameters, which is in a RQF (receiving quality feedback) message that the MMT receiving entity sends.

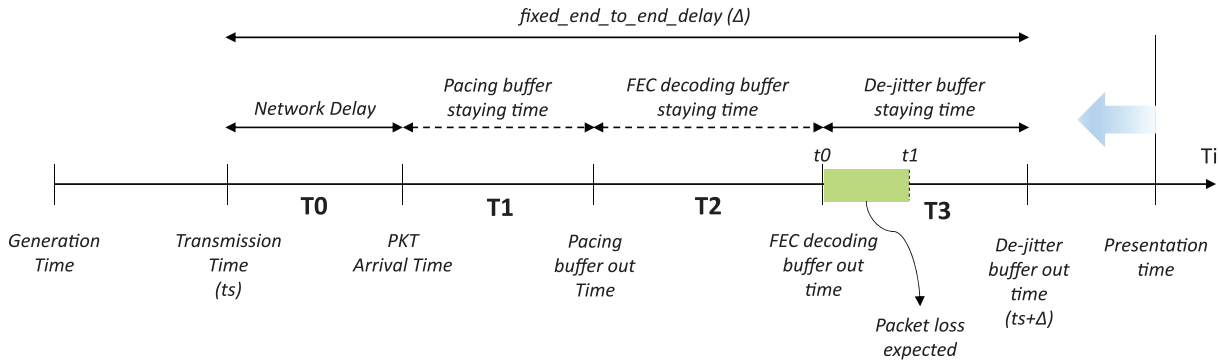


Figure 31 — Timing model diagram of HRBM in MMT

$T1$ means staying time of MMT packets in the pacing buffer. The MMT sending entity can know that value from the two parameters *target_pacing_buffer_level* (which is from pacing buffer status feedback, PSF message) and *pacing_buffer_removal_rate* (which is from pacing buffer removal rate, PRR message) by calculating $T1 = \text{target_pacing_buffer_level} / \text{pacing_buffer_removal_rate}$.

$T2$ means the time that the MMT packet is stored at the FEC decoding buffer. The MMT sending entity can guess that time from the *protection_window_time* delivered in the AL-FEC message if used. If AL-FEC is not used, $T2$ equals zero.

At last, $T3$ value is the time that the packet stays at de-jittering buffer. And $T3$ can be calculated as $T3 = \Delta - (T0 + T1 + T2)$.

Based on the information acquired above, the MMT sending entity can issue a new HRBM message with new updated values in average as;

$$\text{fixed_end_to_end_delay}' = T0 + T1 + T2 + T3$$

$$\text{max_buffer_size}' = \{ \text{max_transmission_delay} - \text{min_transmission_delay} \}$$

$$* \text{max_bitrate} * \text{margin}$$

where,

max_bitrate is the maximum bitrate of the MMTP packet stream that the server sends;

margin is the ratio that might receiving entity should secure to prevent buffer overflow considering the cases such as packet retransmission, etc.

If the new *max_buffer_size'* value is smaller than previous *max_buffer_size*, then MMT sending entity can assume the possibilities of packet loss in De-Jitter buffer due to shrinking buffer size secured. If the packet loss is expected, those packets should be retransmitted to compensate loss at MMT receiving entity.

The MMT sending entity can guess the lost packets in the MMT receiving entity side by estimating transmission time of those packets. After sending the updated HRBM message (HRBM'), the MMT sending entity can expect that new HRBM parameter value will be applied at the MMT receiving entity. In this case, the range of packets expected to be lost at de-jitter buffer will be from the newest packet put at de-jitter buffer at t_0 , to packets stayed until t_1 . Here t_0 and t_1 can be calculated as,

$$t_0 = ts + T1 + T2$$

$$t_1 = t_0 + (max_buffer_size' - max_buffer_size) / max_bitrate * margin$$

where,

max_bitrate is the maximum bitrate of the MMTP packet stream that the server sends;

margin is the ratio that might receiving entity should secure to prevent buffer overflow considering the cases such as packet retransmission, etc.

The MMT sending entity assumes the packets that currently have a life time of t ($t_0 \leq t \leq t_1$) are highly likely to be lost at the receiving side by buffer size shrinking and it can send those identified packets again to the receiving entity without a request by its own decision. In this case, the MMT sending entity should tag the original MMTP packet timestamp ts as a re-transmitted packet header not with current time ts' . This is to ensure re-transmitted packets are transacted fast and not to undergo another new *fixed_end_to_end_delay* which make re-transmitted packets useless due to late arrival at the de-jitter buffer.

Lost packet compensation can also be done through an ARQ feedback operation. Before shrinking the buffer size, the MMT receiving entity identifies the packets that will be out of buffer, and the MMT receiving entity counts their packet sequence number. Then, the MMT receiving entity requests designated packets with an AF message. When it receives the MMT packets, it identifies the timestamp value, ts , in the MMTP packet header and order the packets to be queued in the de-jitter buffer according to the order of ts . The packets with larger ts values will be stacked in FIFO (first in first out) order.

After sending the MPU that is managed with the previous HRBM parameter, the MMT sending entity sends the newly updated HRBM' message to the MMT receiving entity and begins to send the next MPU that will be managed with new HRBM parameters. After the MMT receiving entity receives the new HRBM' message, it recognizes ts at the packet header that carries the new HRBM message, HRBM', and compares it with that of the packet carrying MPU to decide from which packets it will apply the new policy with HRBM'.

5.17 MMTP packet with padded data

An MMTP packet could be delivered over TCP/IP and UDP/IP flows. When the MMT sending entity send the MMTP packet with padded data, it could be identified and calculated to remove the padded data at the MMT receiving entity. To identify the padded data to remove, the size of padded data is needed in the entire delivered IP packet. The way to calculate the size of padded data in an MMTP packet over UDP packet is shown in [Figure 32](#).