TECHNICAL REPORT

ISO/IEC TR 23008-13

Third edition
2020-12

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

Part 13: MMT implementation guidance

Reference number
ISO/IEC TR 23008-13:2020(E)

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

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.

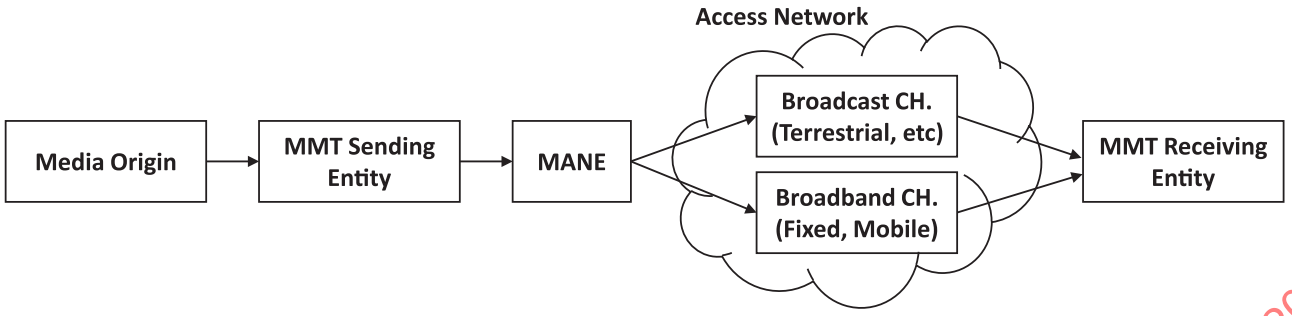


Figure 1 — Example of MMT-based media distribution chain

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.

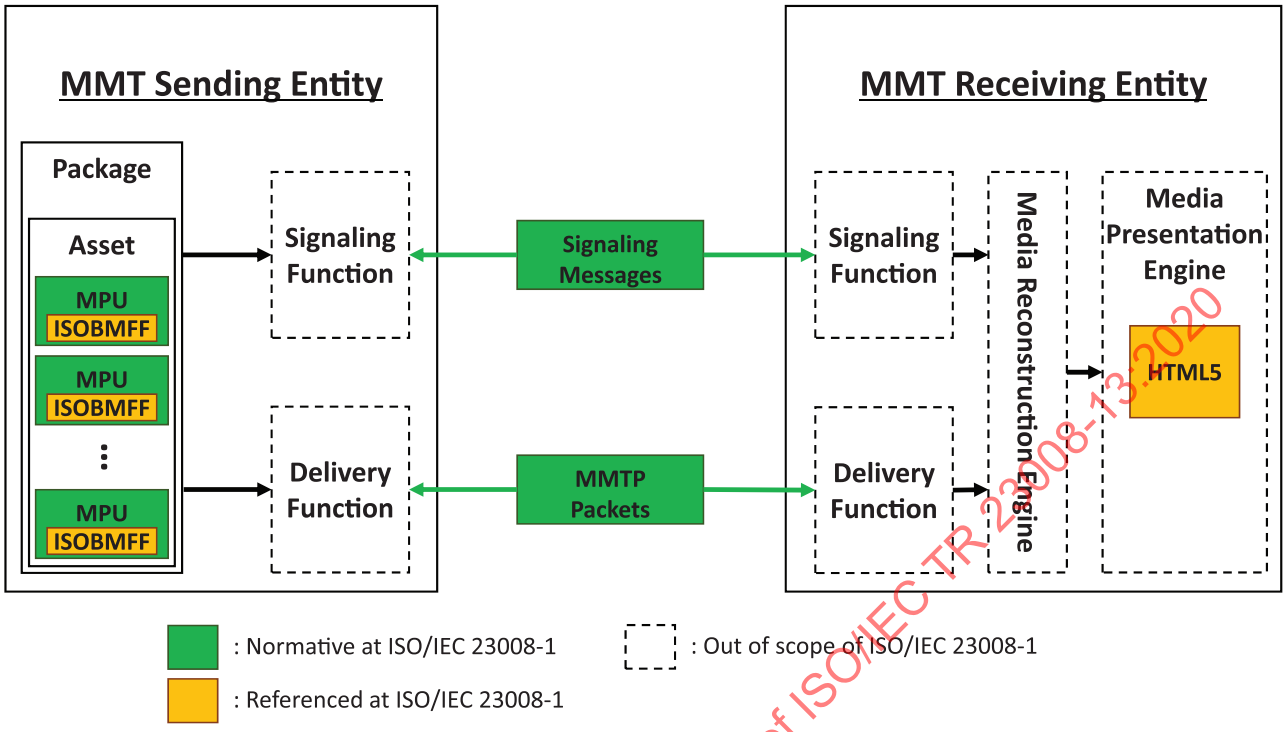


Figure 2 — MMT functions deployment

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.

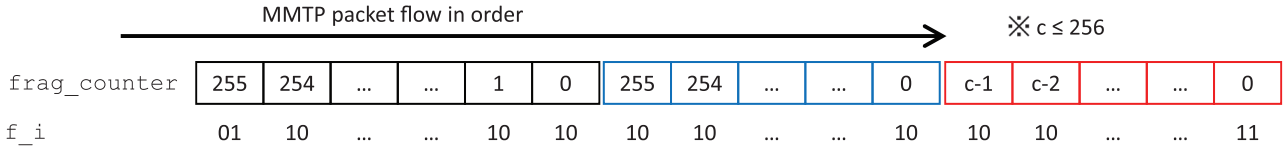


Figure 3 — Fragment counter and fragmentation indicator changes for 513~768 fragments

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 (ISOBMFF) with certain restrictions. However, the usage of the ISOBMFF may give the impression of high end-to-end delay, which may seem not suitable for live broadcast. This 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).

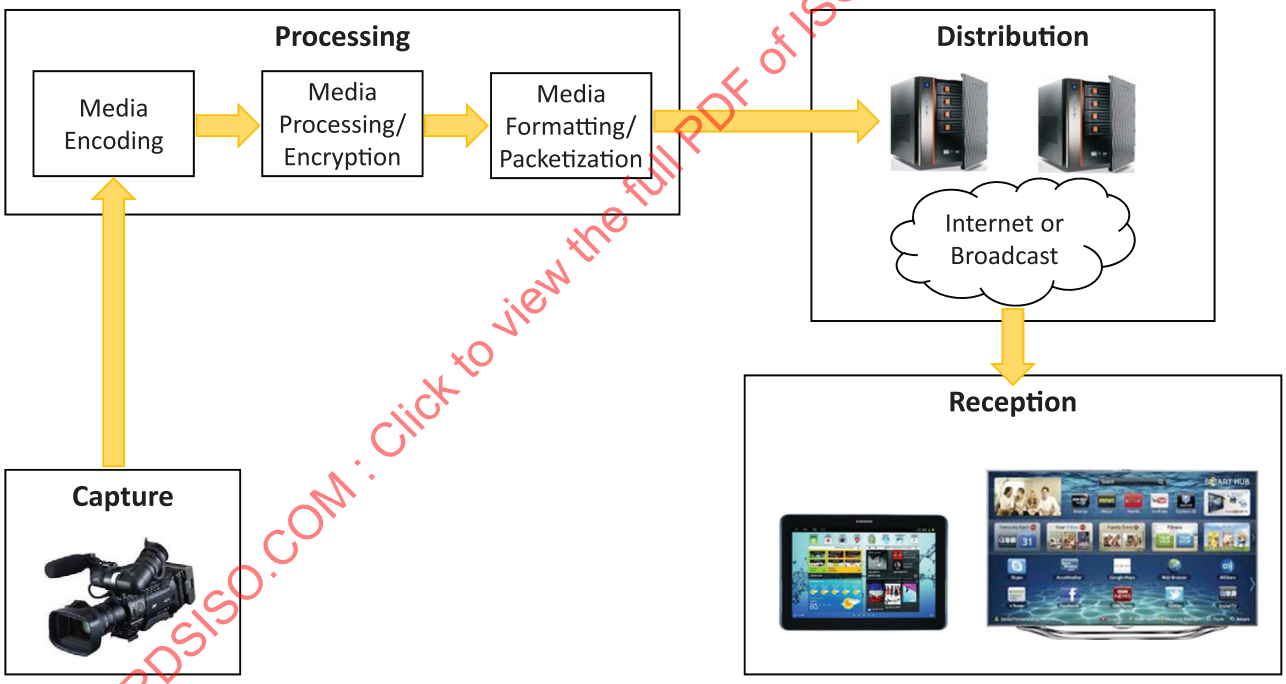


Figure 4 — Example of a broadcast scenario

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 ISOBMFF 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.

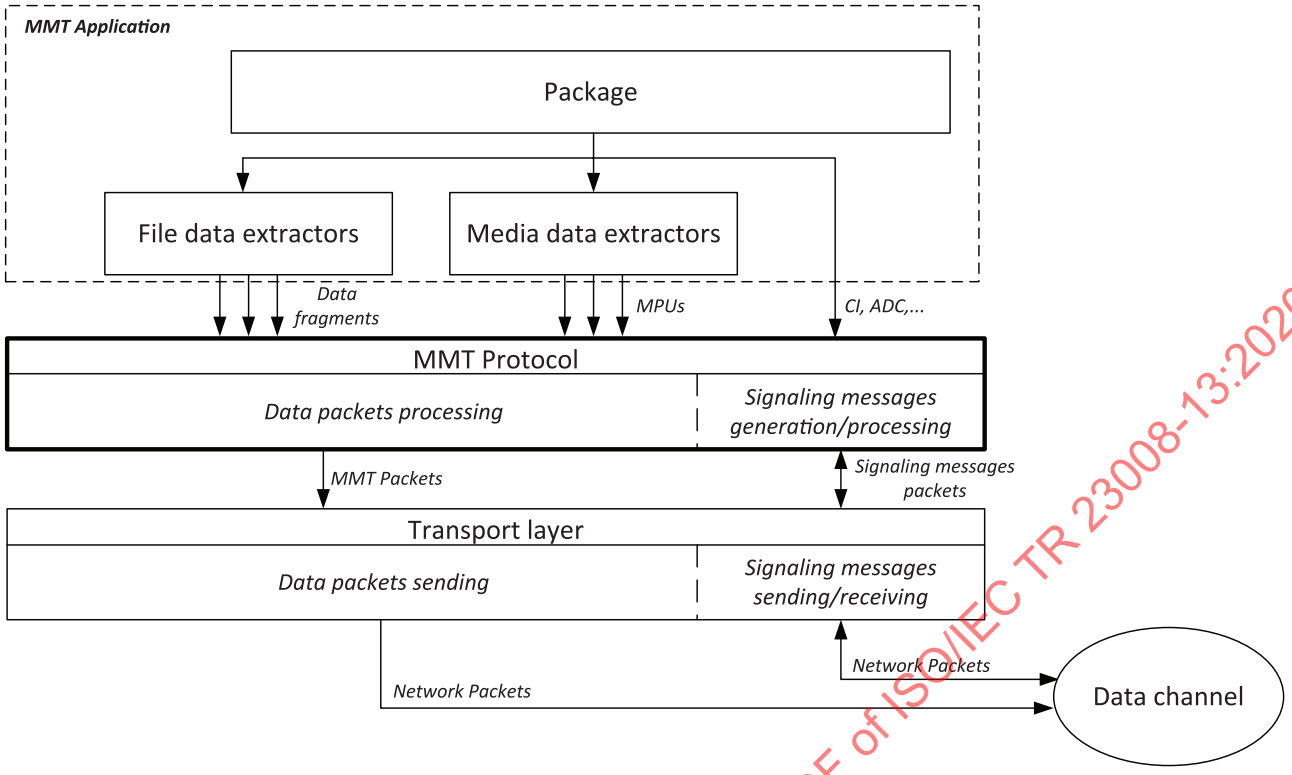


Figure 6 — MMT sending entity structure

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.

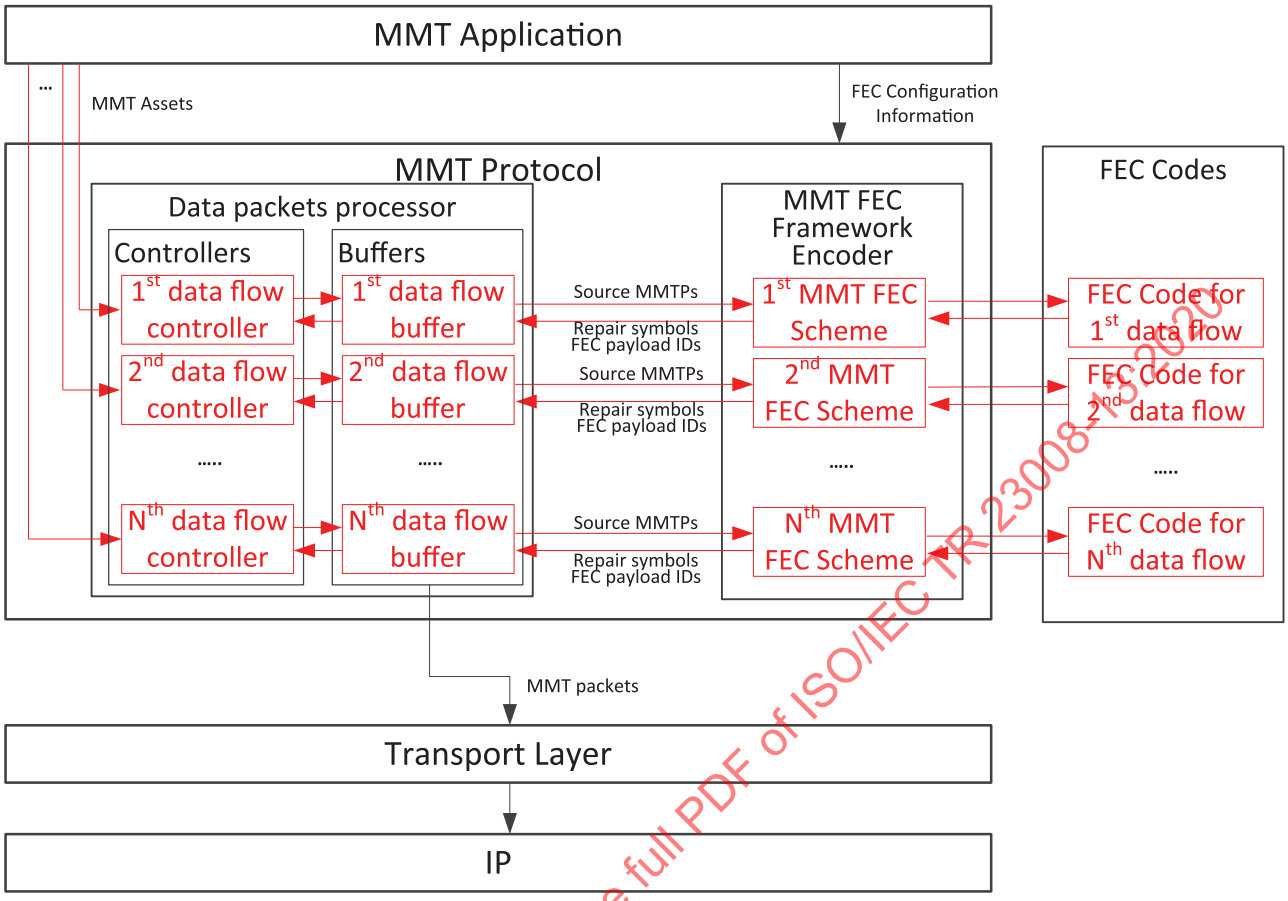


Figure 7 — Architecture for AL-FEC (MMT sending entity)

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.

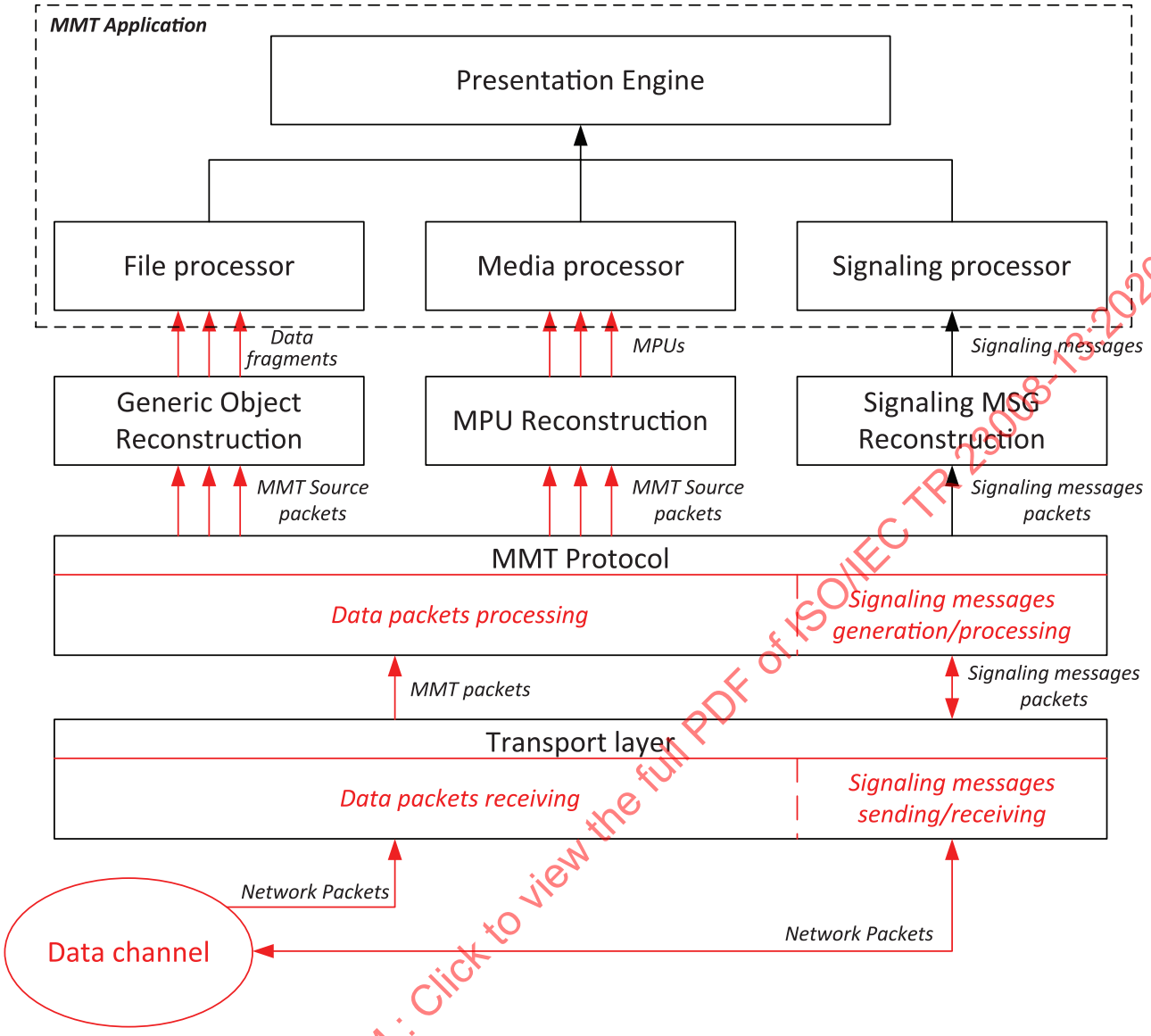


Figure 8 — MMT receiving entity structure

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.

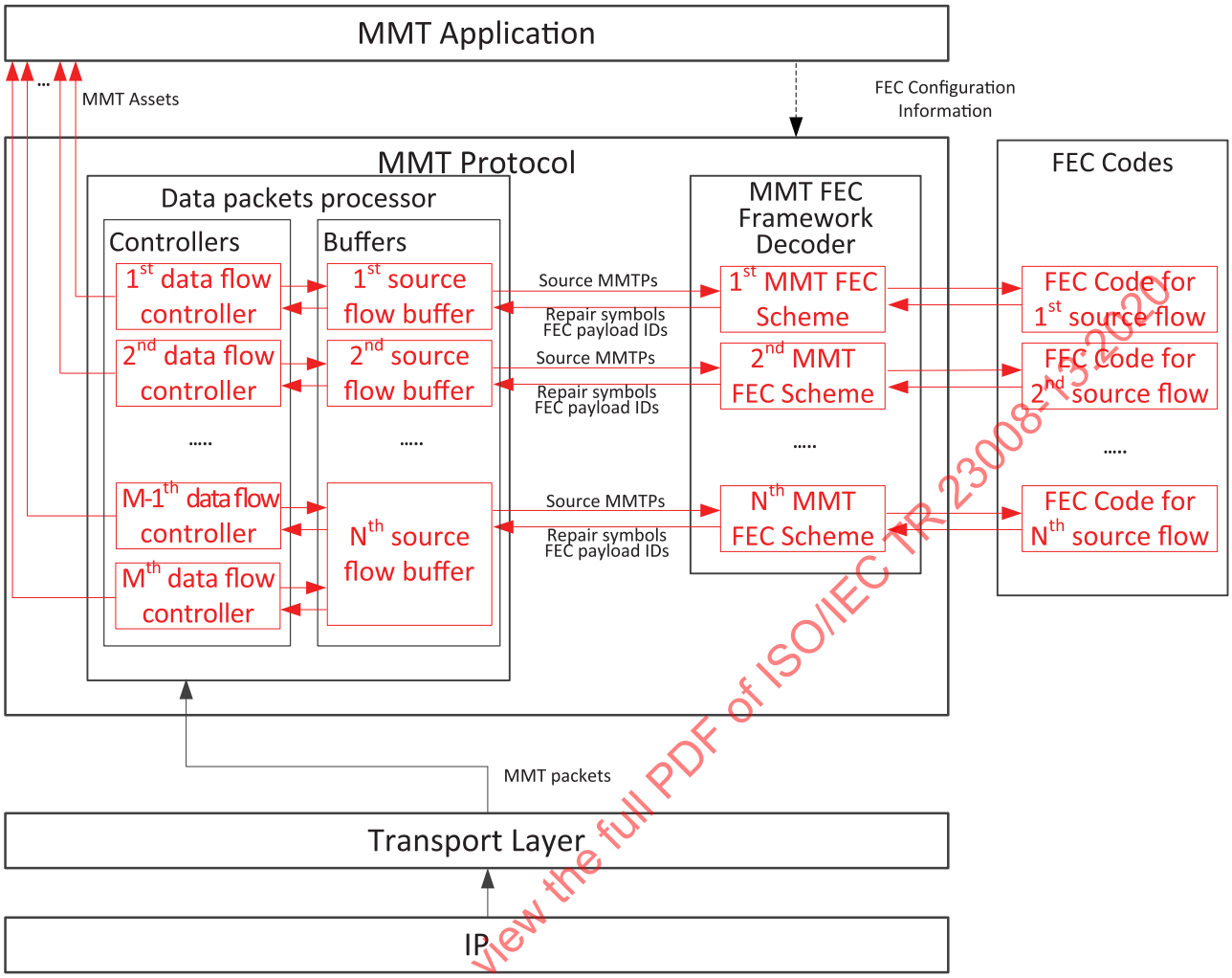


Figure 9 — Architecture for AL-FEC (MMT receiving entity)

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.

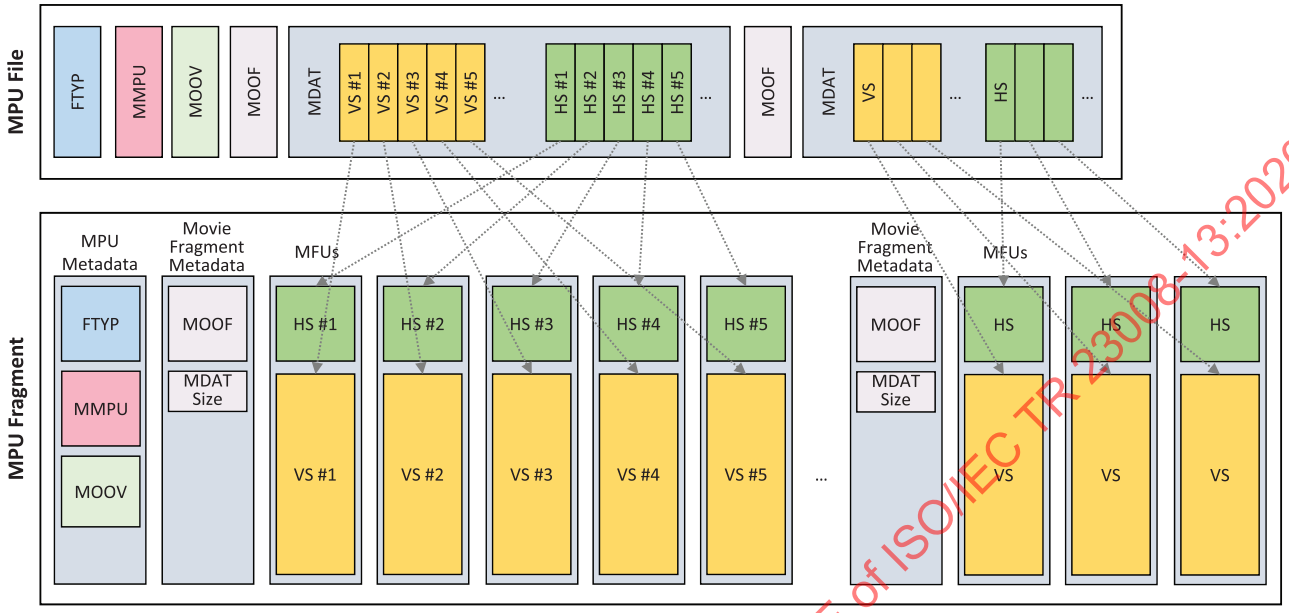


Figure 10 — MPU fragment building block

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.

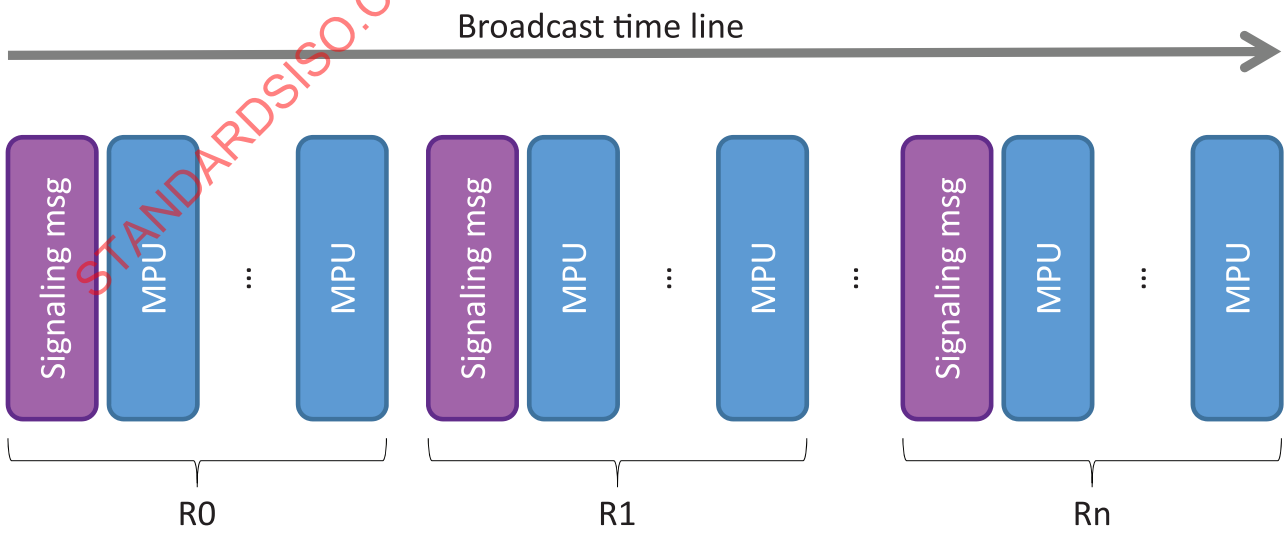


Figure 11 — Periodical signalling information transmission

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.

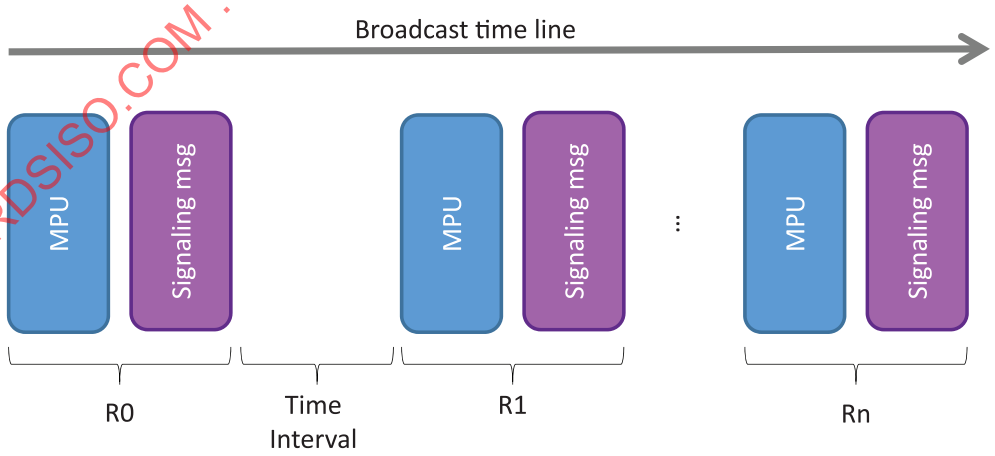


Figure 12 — Signalling message behind the MPU

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.

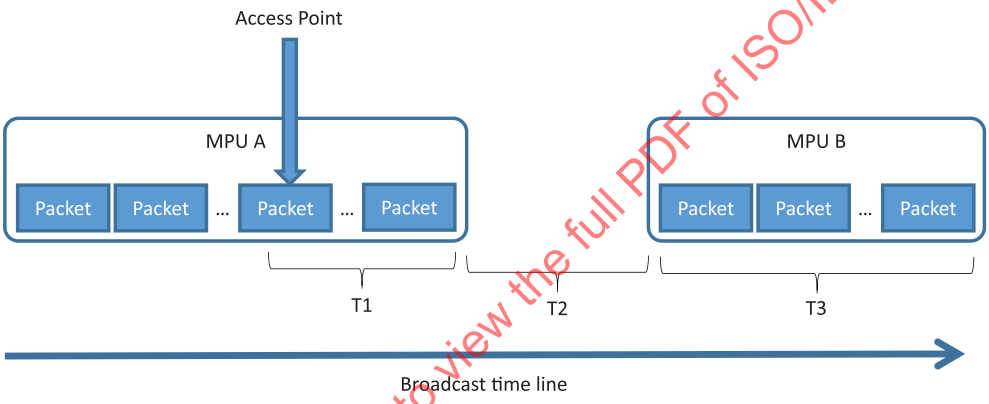
When the MMT receiving entity randomly accesses the broadcast channel, there are two cases: the MMT receiving entity accesses the channel while a certain MPU transmission is ongoing; or the MMT receiving entity accesses the channel during the time interval between consecutive MPUs.

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.

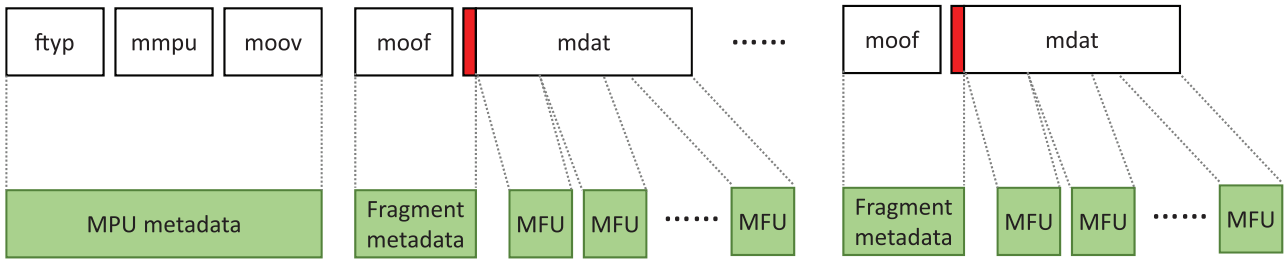


Figure 14 — MMTP payload generation

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.

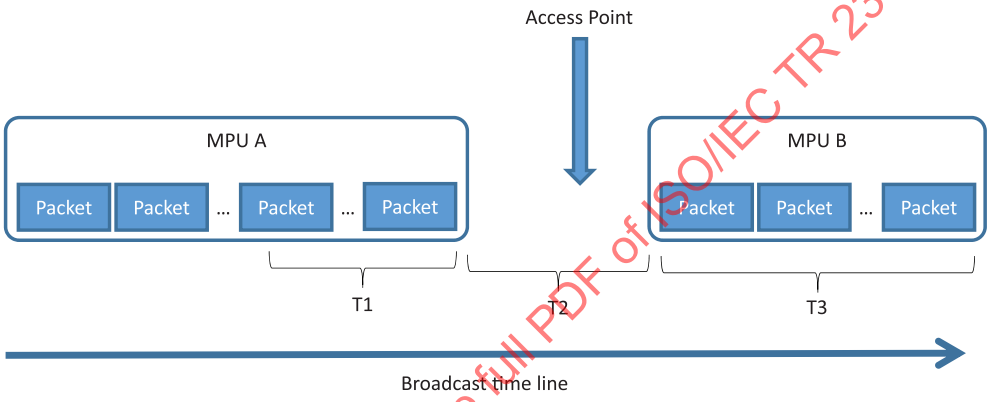


Figure 15 — Random access in the middle of interval (case 2)

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, samplenumber

      The values of '*sequence_number*', '*movie_fragment_sequence_number*' and '*samplenumber*' 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 scalibility, 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.

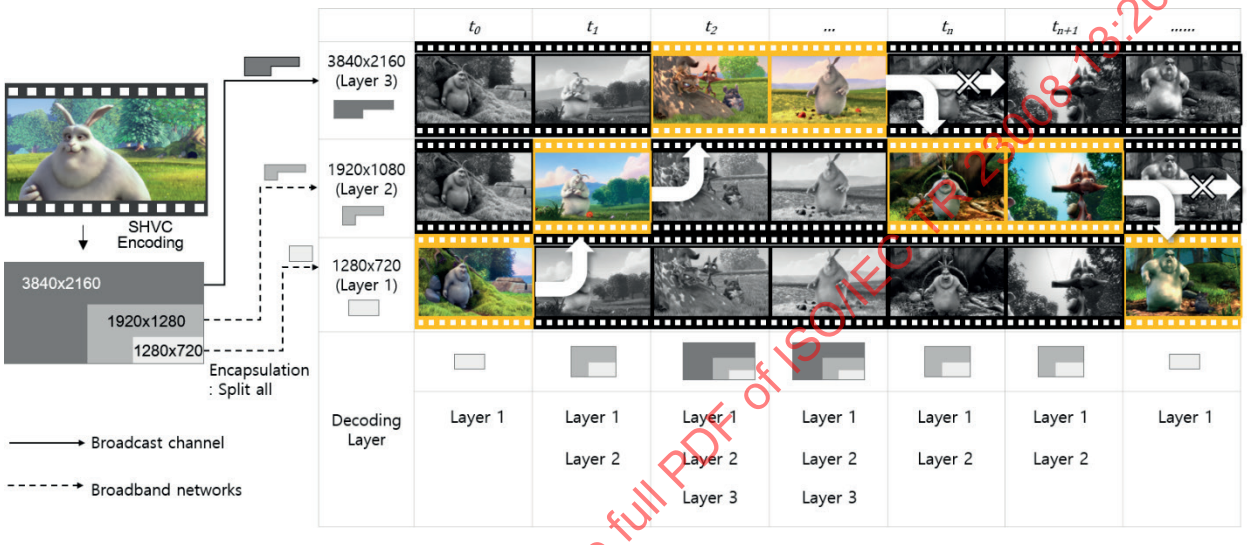


Figure 16 — Test scenario

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 (ISOBMFF). 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 component[1)] such as MPEG-2 TS.

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

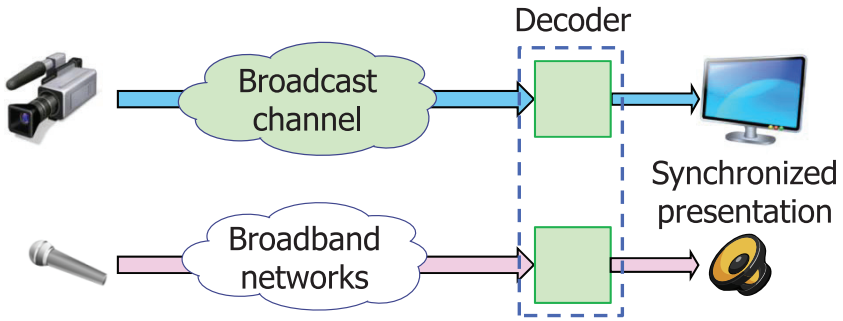


Figure 18 — Combination of streaming components for presentation

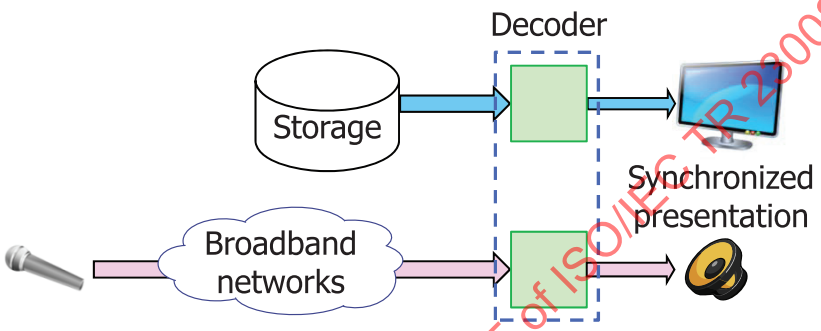


Figure 19 — Combination of streaming component with pre-stored component for presentation

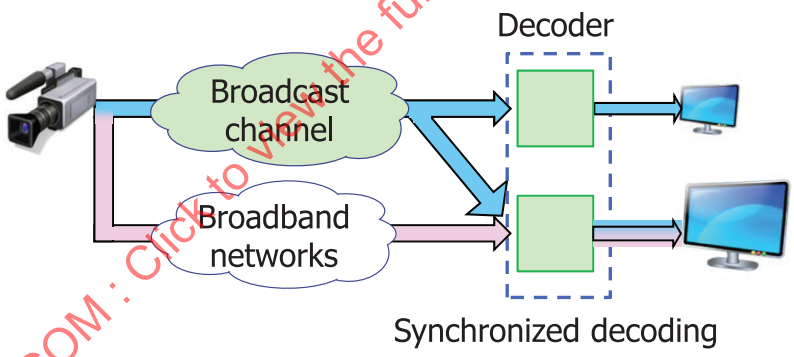


Figure 20 — Combination of components for decoding

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 information[2].

— 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.

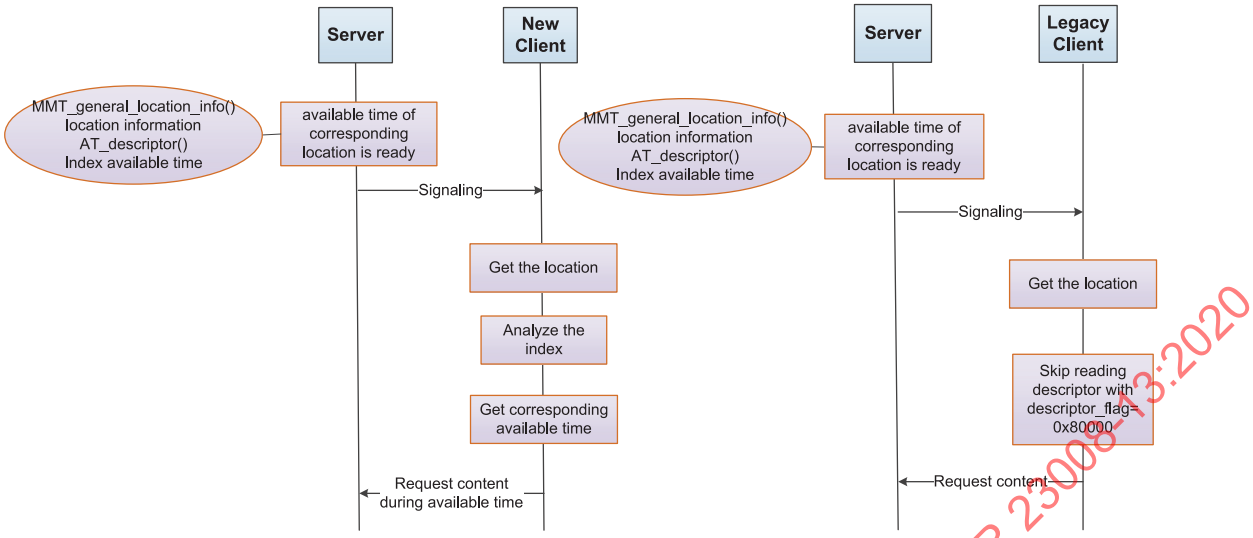


Figure 21 — Content request procedure

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.

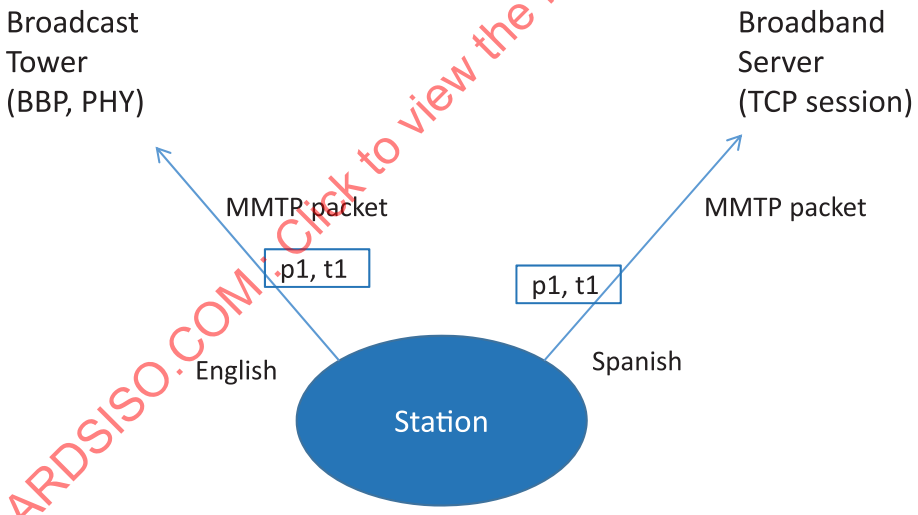


Figure 22 — Single station for hybrid delivery

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.

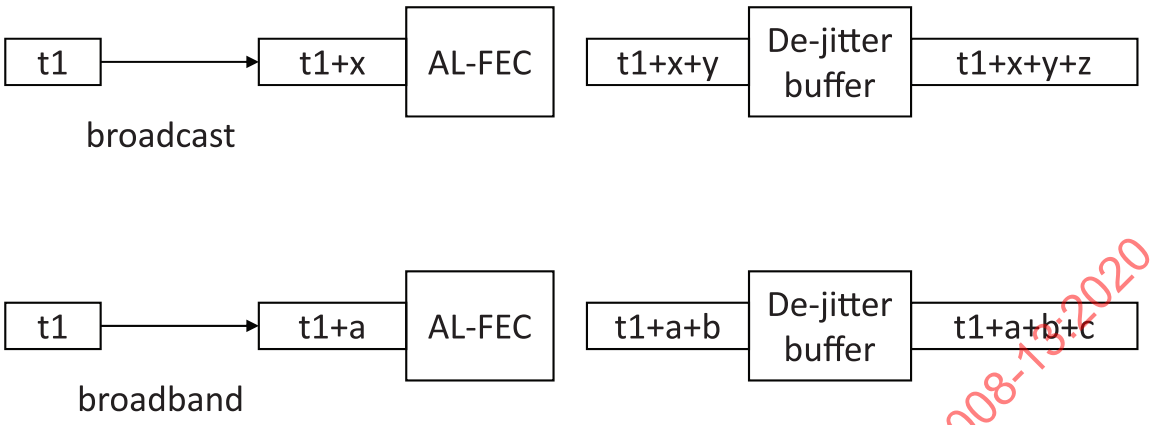


Figure 23 — HRBM buffers

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:

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

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.

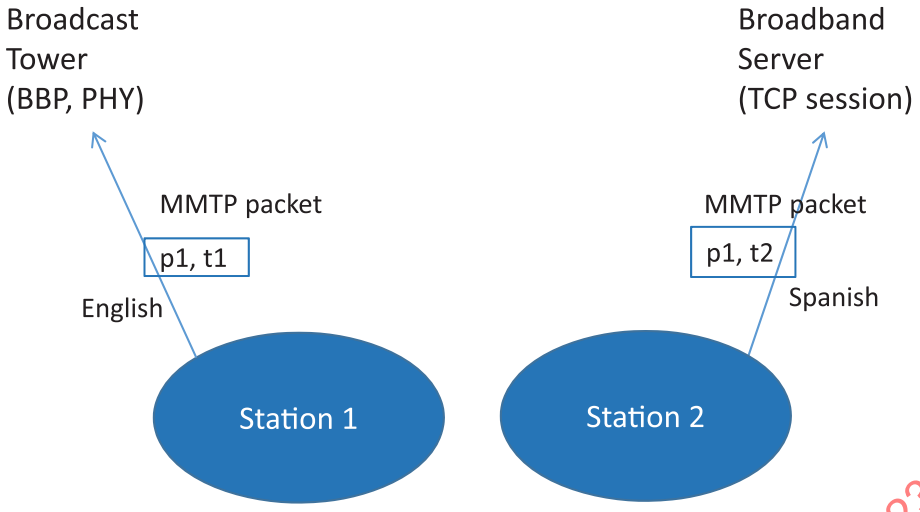


Figure 24 — Multiple stations for hybrid delivery

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).

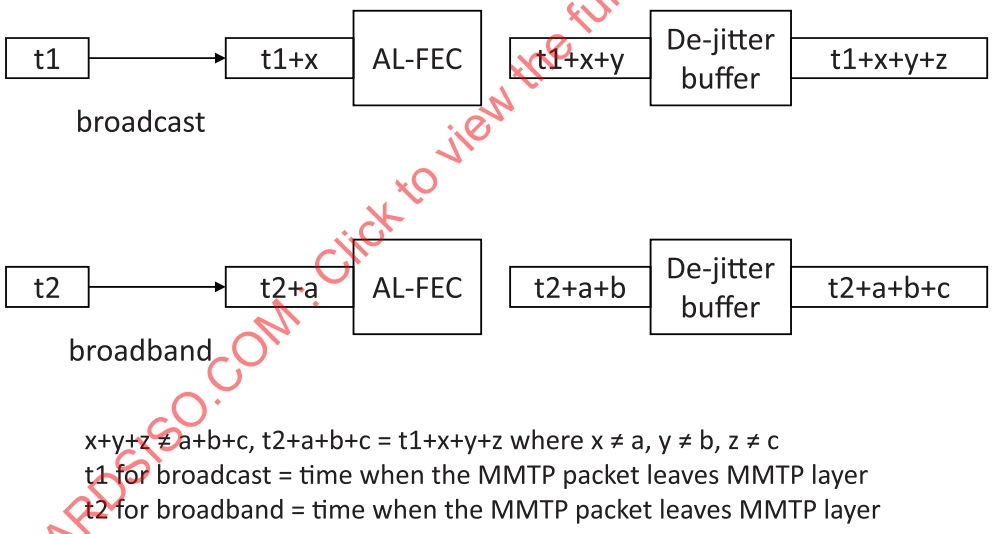


Figure 25 — Multiple HRBM buffers

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 *alpha* which are calculated by max_transmission_delay corresponding to network B, in this case, *alpha* 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.

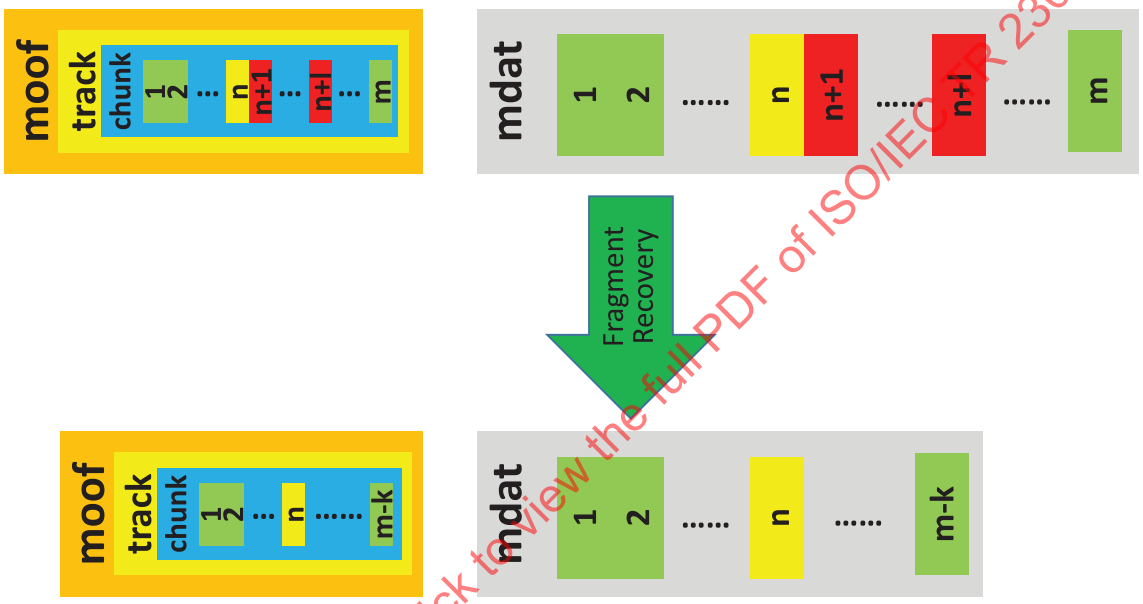


Figure 26 — Fragment recovery after packet loss

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.

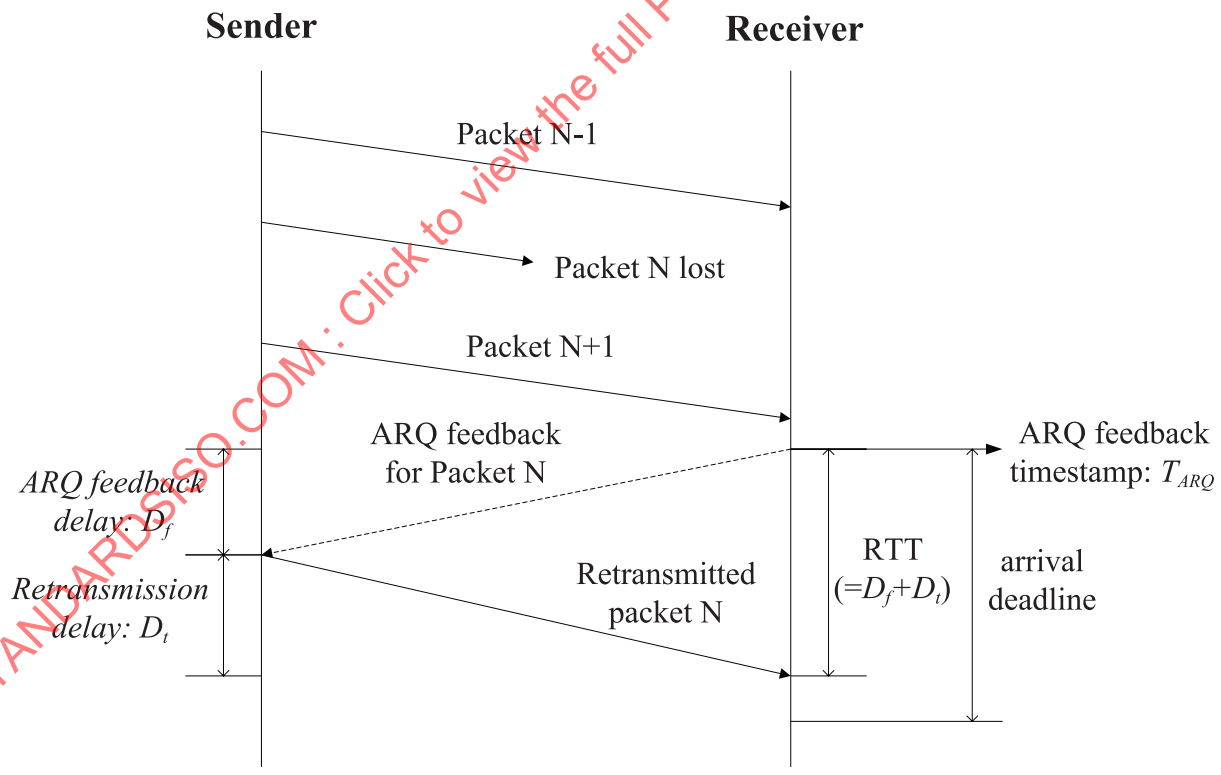


Figure 27 — Operation of the arrival-deadline constrained ARQ

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 re-transmission 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.

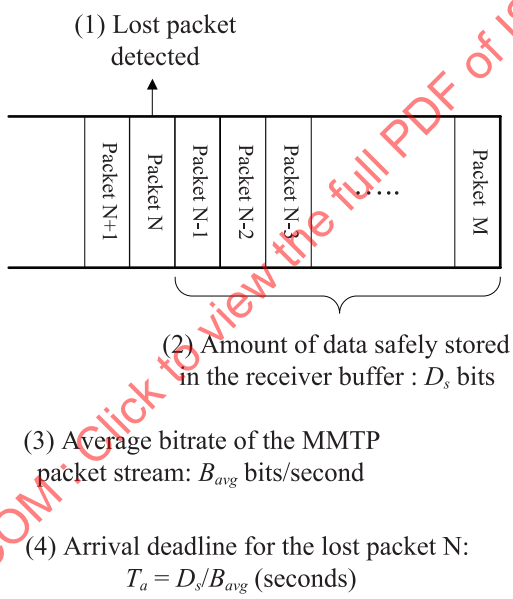


Figure 28 — Example of calculating *arrival_deadline* at the receiver

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.

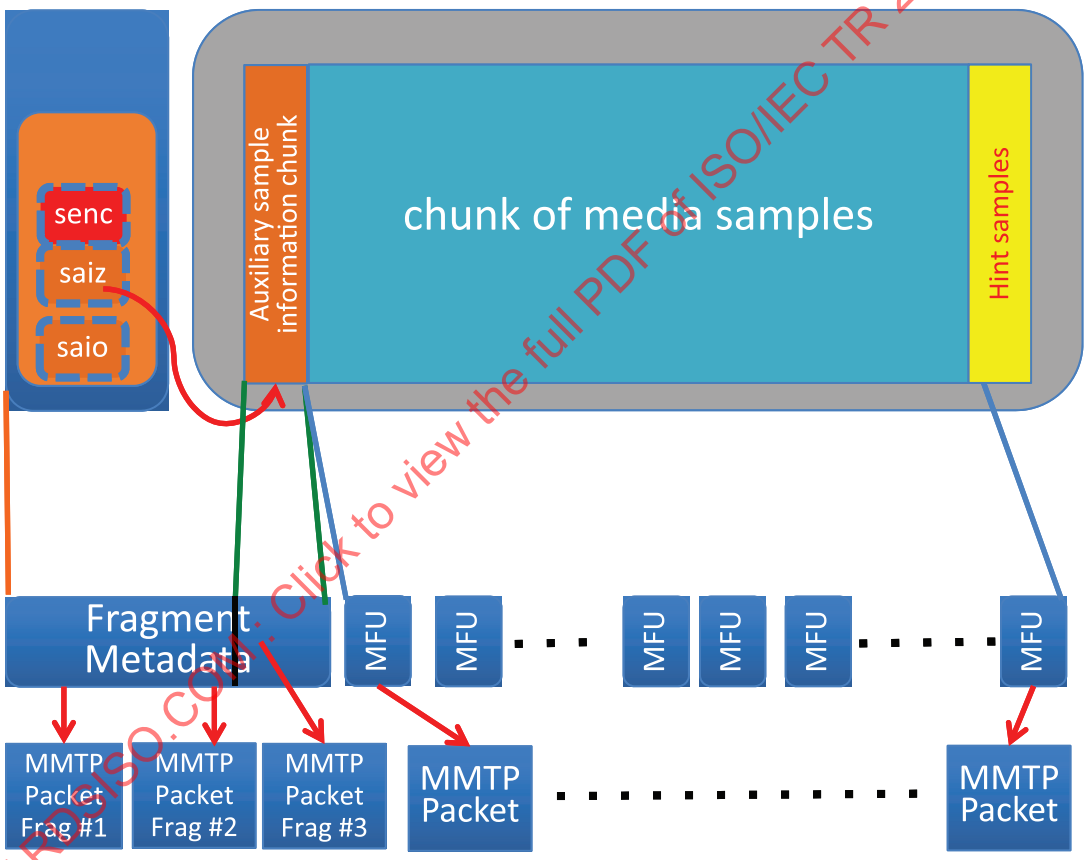


Figure 29 — Packetization considering encryption

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: $M*D$ bytes

— Fixed end-to-end delay: F seconds (the minimum transmission delay = d)

— The required buffer size: $B = (F - d) * M$ 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:

$$M' = (M * D) / (D - J).$$

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

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

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

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 enviroment 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.

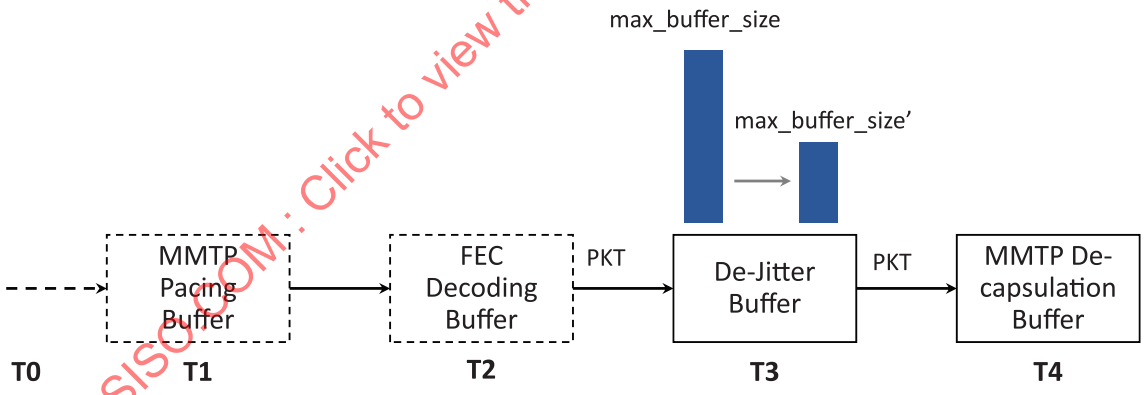


Figure 30 — HRBM buffer model in MMT

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, *ts* + Δ) 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, *T0* 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.

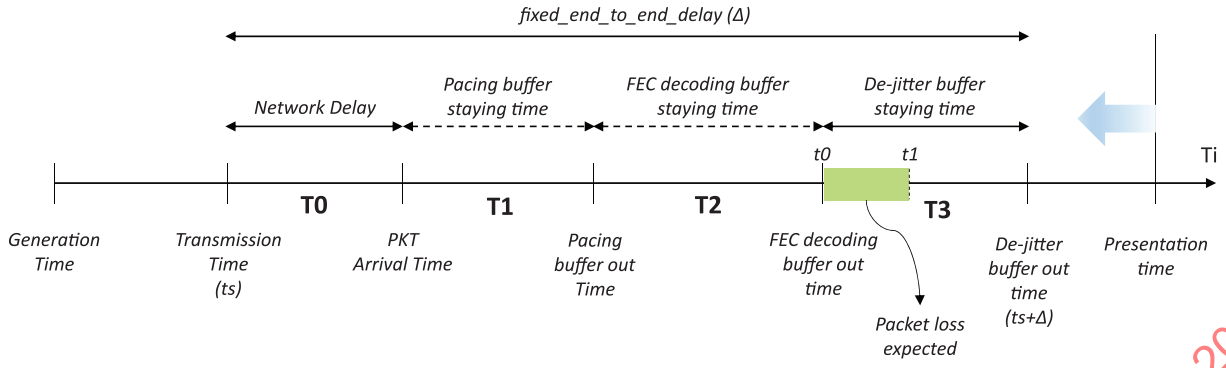


Figure 31 — Timing model diagram of HRBM in MMT

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 = target_pacing_buffer_level / 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 = Δ − (*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;

fixed_end_to_end_delay' = T0 + T1 + T2 + T3

max_buffer_size' = { max_transmission_delay − min_transmission_delay }

** 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.

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 HBRM 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.

Figure 32 — MMTP packet over UDP

As shown in Figure 32, for an MMTP packet over UDP, the length of the UDP packet is specified as the length in bytes of the entire datagram which is consisted with the UDP header and data in the UDP header. The field size sets a theoretical limit of 65,536 bytes (e.g., the minimum length of 8 bytes header + 65,527 bytes of data) for a UDP datagram. In IPv4, the practical limit for data length is 65,507 bytes (65,536 bytes – 8 bytes of UDP header – 20 bytes of IP header). The MMTP packet header indicates the length of MMTP packet header (V=1), which contains the minimum length of MMTP packet header (which is 18 bytes) and the size of extension header (see Figure 33). The length of padded data could be calculated by provided information in the MMTP packet header with extension header fields.

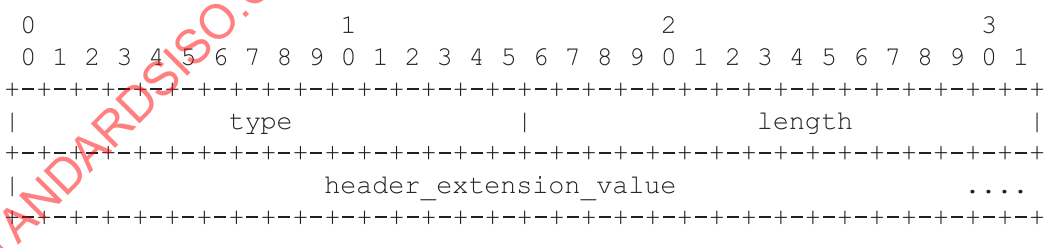


Figure 33 — MMTP packet header with extension header field

For example, the “type” indicates this payload contains the padding data, the “length” presents the size of value and the “header_extension_value” provide the size of padding data from the end of UDP packet tail.

5.18 Constraints on signalling splicing points

5.18.1 General

Generally, a primary MMT sending entity which initiates the MMT session for MMT applications sends the ACR message. When a MANE or a MMT middle box receives the ACR message which includes the parameter FEC_clean_type, after the FEC processing signalled by FEC_clean_type, it locates the changing point in the target asset via the random access information, RAP_flag, in the MMTP packet and performs the requested change with the designated asset in terms of the specified settings in the ACR message. There are some constraints on MMT assets that are specified for changing points or splicing points in order to have a good splicing results including ordinary splicing and seamless splicing.

5.18.2 Constraints on Case 1 – Asset change at the start of MPU

For a single MPU asset, the asset change at the start of MPU is the same as at the start of the asset. It usually generates a smaller size of asset. Generically an MMT asset may contain multiple MPUs and an MPU starts with a SAP.

— MMT asset change points occur at the MPU boundaries.

  This subclause defines the asset change-in point as the point right before the first target MPU in the first target assets, and the asset change-out point as the point immediate after the last target MPU in the last target assets. The asset changing point is either the asset change-in point or the asset change-out point. This subclause also uses the asset splicing points and the asset changing points interchangeably.

— The first target MPU and/or the first designated MPU should start with a SAP of Type 1 or Type 2[3] in terms of the change type in the ACR message.

  NOTE 1 It ensures a clean processing starting from the asset change-in point till the asset change-out point. The asset change-in point and the asset change-out point can be at the same place, e.g., for the change type of 'insertion'.

  NOTE 2 The MPU of the target asset or the designated asset starts with other SAP types, e.g., Type 3, may cause complicated situations in decoding/rendering process, such as non-smooth decoding (sometime erroneous decoding, depending on a MANE device's capability) and timing mismatch.

— For the change type of 'replacement', the first designated MPU should start with a SAP of Type 1 or Type 2 and the first target MPU after the change-out point should start with a SAP of Type 1 or Type 2.

— For the change type of 'insertion', the first target MPU after the change-in point and the first designated MPU should start with a SAP of Type 1 or Type 2.

— For the change type of 'overlay', the first designated MPU should start with a SAP of Type 1 or Type 2.

5.18.3 Constraints on Case 2 – Asset change at a point in MPU

For a single SAP MPU, the asset change is always at the start of MPU. It usually generates a smaller size of MPU. Generically an MPU may contains multiple SAPs with Type 1, Type 2 or Type 3 etc.

— MMT asset change points occur at the start of a SAP.

  This subclause defines the first target SAP as the first SAP point in the target assets as the changing point for the asset change and defines the first designated SAP as the first SAP point in the designated assets for the asset change.

— The first target SAP and/or the first designated SAP should be of Type 1 or Type 2[3] in terms of the change type in the ACR message.

— For the change type of 'replacement', the first designated SAP should be of Type 1 or Type 2 and the first target SAP after the change-out point should be of Type 1 or Type 2. The changing point is the start of the fragment that contains the SAP.

— For the change type of 'insertion', the first target SAP after the change-in point and the first designated SAP should be of Type 1 or Type 2. The changing point is the start of the fragment that contains the SAP.

— For the change type of 'overlay', the first designated SAP should be of Type 1 or Type 2. The changing point is the start of the fragment that contains the SAP.

5.18.4 Signal the splicing point for target assets in Case 1 and Case 2

In the MMTP packet header, the RAP_flag may mark for (timed media):

— **Signalling message**

— **MPU fragment type 0**: MPU metadata – contains the ftyp, mmpu, moov, and meta boxes as well as any other boxes that appear in between.

— **MPU fragment type 1**: Movie fragment metadata – contains the moof box and the mdat box, excluding all media data inside the mdat box but including any chunks of auxiliary sample information

— **IRAP picture**: I picture as a RAP.

NOTE An IRAP picture can, as an MFU, appear in the middle of a MPU.

For Case 1 with the constraints, the RAP_flag may mark only the first SAP of the MPU. If MPUs are of a big size, the number of splicing points is limited.

For Case 2 with the constraints, the RAP_flag may mark each of the SAPs in MPUs. It generates a good numbers of splicing points, though all SAPs starting with Type 1 of Type 2 may reduce video coding efficiency.

6 Use cases for MMT deployment

6.1 General

This clause gives implementation guidance on specific deployment use cases and examples utilizing ISO/IEC 23008-1 for specific purposes. In particular it guides implementers on how MMT specification can be utilized or extended for the specific topics such as, but not limited to:

— DASH over MMT;

— network caching/MANE utilizing MMT;

— usage and extensions of ISO/IEC 23008-1 for Japanese broadcasting services.

6.2 Delivery of DASH presentations using MMT

6.2.1 General

When streaming a DASH presentation, e.g., over a broadcast channel, the HTTP protocol can no longer be used. MMTP provides the necessary tools to support the delivery of a DASH presentation. A DASH presentation consists of the presentation information (PI) (which is the MPD) and the data segments (initialization and media segments).

6.2.2 Delivery of the MPD

The MPD makes the core part of the presentation information of the presentation. The MPD is assigned its own MIME type: “application/dash+xml”, which is used to identify the type of the presentation. The MPD is embedded in the MPI table, which in turn is delivered using the MPI message. The format of the message may either be binary or XML. In case of XML format, the MPD is embedded using the <![CDATA[ ]]> encapsulation. The MPD updates are delivered using the same mechanism. The updates are discovered by checking for the table identifier in signalling messages of that particular package.

The MPD is then used by the client (DASH client) to drive the presentation. It is required that all segments that are addressed by the MPD should be readily available at the receiver side, i.e. delivered and recovered by their segment availability start time. This may be ensured by setting delivery schedule and the HRBM parameters appropriately.

6.2.3 Delivery of the data segments

6.2.3.1 Delivery using the MPU mode

When delivering DASH content using the MPU mode, each Representation is considered as an MPU. It is important to note that the concatenation of all segments of a Representation results in a fragmented ISOBMFF compatible file, which is equivalent to an MPU in MMT terms.

The initialization segment of the Representation is the MPU metadata and is marked appropriately during the delivery to ensure that it is recovered by the receiver. The MPU metadata is delivered repetitively to ensure that all clients, independent of the time they join the session will ultimately be able to recover it. The MPU mode provides the means for marking these important packets by setting the FT field to MPU metadata. Similarly, fragment metadata is important for every fragment and should also be delivered repetitively to increase the probability of recovery.

All segments of a Representation are then marked as belonging to the same MPU by using the same packet_id and MPU sequence number to mark them. The mapping between the Representation and the packet_id and MPU sequence number is provided by the identifier_mapping syntax element that is provided in the MP table. The simplest usage of the identifier_mapping in this case is to provide the representation_id as provided in the MPD. Other approaches, e.g., using the URL template for all segments of that Representation may be used instead.

At the sender side, the data is delivered on movie fragment basis. However, at the receiver side, each media segment needs to be recovered appropriately and timely. It is then necessary for the receiver to be able to identify the segment boundaries and the segment information. This may be done in one of the following ways:

— a segment is recovered by identifying its boundaries using the presence of the segment index box (“sidx”) from the current and the following segment;

— a segment is recovered by checking the timestamp of the first sample in the movie fragment (for example, extracted from the “tfdt” box or the “sidx” box) and using the MPD to identify the media segment that has the same starting time;

— the segment boundary may be deduced from the segment duration by dividing the fragment start time by the duration of the media segment for that Representation (see Figure 34).

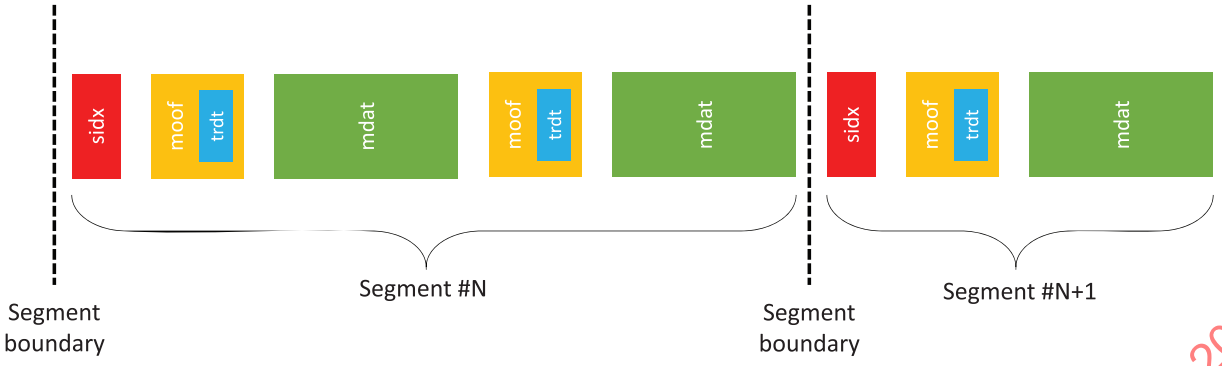


Figure 34 — Segment boundary detection

6.2.3.2 Delivery using the GFD mode

When using the GFD mode, each segment of the DASH Representation is delivered as a regular file. The connection between those segments may be established using the packet_id. The connection between the file (transport object) and the segment information is established using the identifier_mapping information in the MP table. The most appropriate mapping in this case could be the URL template or the URL list. Based on that mapping the TOI can be used to directly recover the segment number and the segment URL.

6.3 Client operation for DASH service delivered through MMT protocol

6.3.1 Delivery of MPD with MMTP

For the application using MMTP for the delivery of DASH service, MPD can be received as an MPI_table() with MIME type set to application/dash+xml.

6.3.2 Delivery and consumption of DASH segments with MMTP

6.3.2.1 Reconstruction of DASH segments

6.3.2.1.1 Use of GFD mode

MMTP provides GFD mode for the delivery of any type of generic file. Any DASH segment can be delivered as a generic file using the GFD mode of MMTP. The reconstruction procedure described in subclause 5.2.3 is applied to reconstruct DASH segments in this mode.

6.3.2.1.2 Use of MPU mode

MPU mode of MMTP is used to deliver MPUs. Therefore, MPU mode can be used to deliver a DASH segment when it also has 'mpuf' as a compatible brand. The reconstruction procedure described in subclause 5.2.2 is applied to reconstruct DASH segments in this mode.

6.3.2.2 Consumption of DASH segments

6.3.2.2.1 Client with HTTP cache

If the client receiving DASH segments over MMTP has an HTTP/1.1 cache, reconstructed DASH segments are consumed by the DASH client through the HTTP cache as shown in Figure 35. As soon as complete DASH segments are reconstructed, they are moved to the HTTP/1.1 cache, then the DASH client receives DASH segments through normal HTTP/1.1 request/response operations.

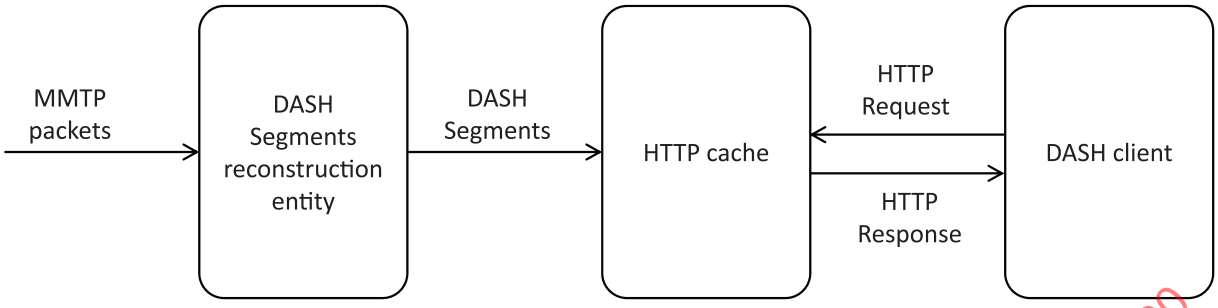


Figure 35 — Client with HTTP cache

6.3.2.2.2 Client without HTTP cache

If the client receiving DASH segments over MMTP does not have an HTTP/1.1 cache, reconstructed DASH segments are consumed by the DASH client through special DASH segments buffered as shown in Figure 36. As soon as complete DASH segments are reconstructed, they are moved to a DASH segments buffer which stores DASH segments with associated URLs as shown in Figure 37. Associated URLs of DASH segments can be found from a GFDT when GFD mode is used or from an mmpu box when MPU mode is used. The DASH client in this case looks in the DASH segments buffer to find an appropriate segment by using a URL as an index for the search. There may be more than one representation delivered into the DASH segments buffer, each comprising different data rates, levels of video quality and robustness. If a DASH segment at the desired quality level is not found, a DASH segment at the next lower quality level can be found, if available.

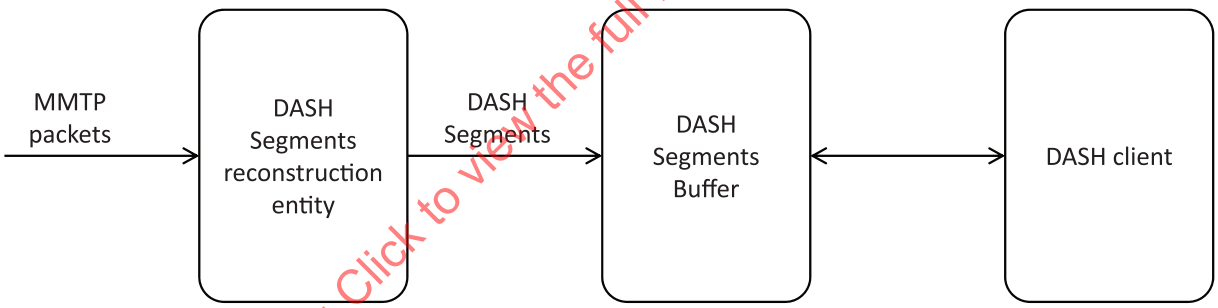


Figure 36 — Client without HTTP cache

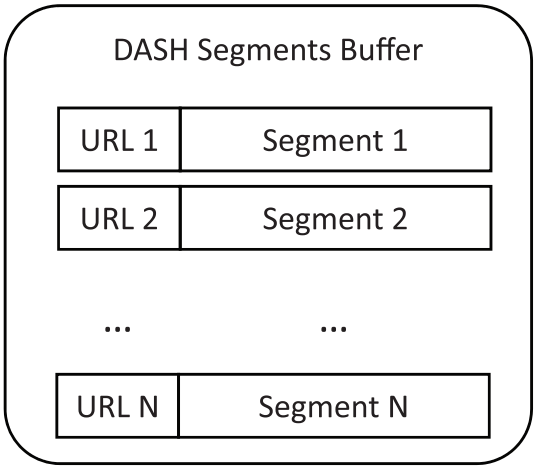


Figure 37 — Storage of DASH segments

6.4 Hybrid of MMT and DASH over heterogeneous network

Figure 38 shows a simplified architecture of the client implementing both MMT and DASH. The figure shows the most important components of MMT and DASH only.

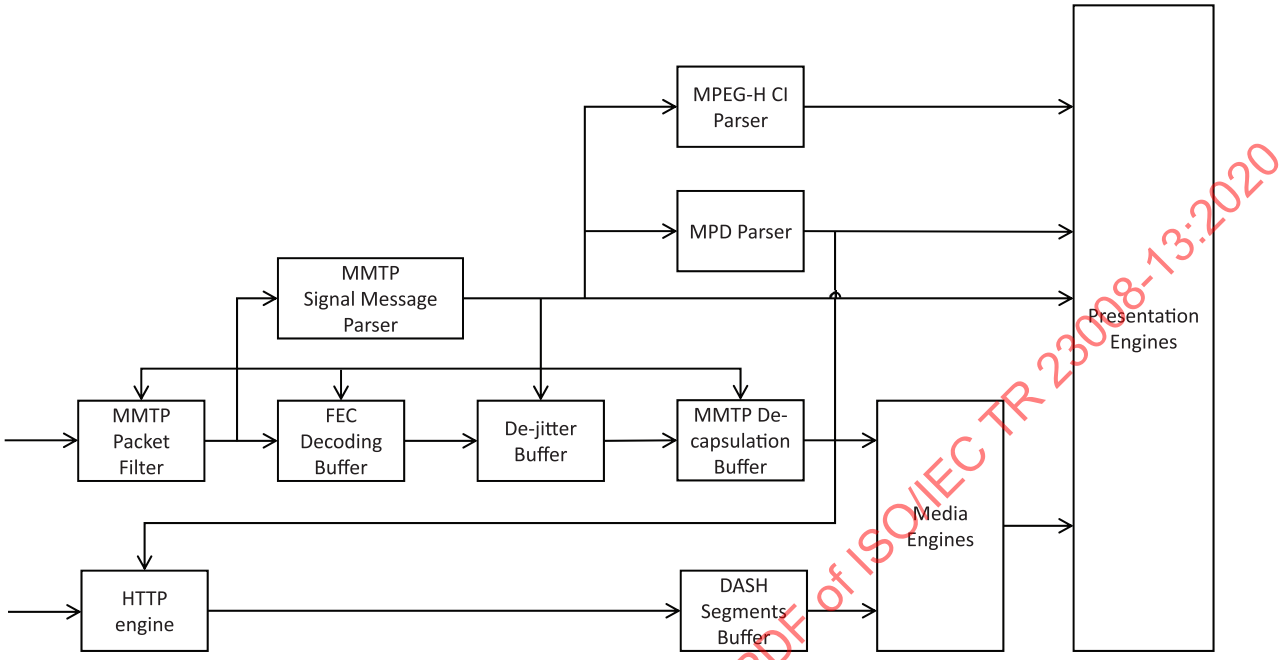


Figure 38 — Conceptual architecture of hybrid service client

The function of each block is as follows:

1. The MMTP packet filter receives MMTP packets from the broadcast network and delivers them to the appropriate processing block, either the MMTP signalling message parser or the FEC decoding buffer. The MMTP packet filter performs the following checks depending on the packet_id of each MMTP packets.

    a. When the client is tuned into the specific service, then MMTP packet filter is only given the packet_id of MMTP signalling messages by the processing block, which is not shown in Figure 38. After the MPT_message is processed by the MMTP signalling message parser, the MMTP packet filter is given the packet_id of the media component and starts to filter MMTP packets with the media component.

2. The MMTP signaling message parser receives MMTP packets from the MMTP packet filter and parses the MMTP signalling message carried in the packet. Information retrieved from the MMTP signalling message is delivered to the appropriate processing blocks.

    a. When the client is tuned into the specific service, the MMTP signalling message parser firstly looks for the MPT_message and processes it to get the list of MMT assets and their packet_ids.

    b. Whenever an MPI_message is received, it processes the PI_content_type_byte field of the MPI_table() and then delivers the data carried as PI_content_byte of the MPI_table() to either the MPEG-H CI parser, MPD parser or presentation engine. If the value of the PI_content_type_byte is HTML5 then the contained data is delivered to the presentation engine. If the value of the PI_content_type_byte is MPEG-H CI then the contained data is delivered to the MPEG-H CI parser. If the value of the PI_content_type_byte is DASH MPD then the contained data is delivered to the DASH MPD parser.

    c. Whenever an AL_FEC message is received, the information in the message is signalled to the FEC decoding buffer.

d. Whenever an HRBM message is received, the information in the message is signalled to the de-jitter buffer.

3. The MPEG-H CI parser receives MPEG-H CI from the MMTP signalling message parser and processes it. The presentation time of the first access unit of the MPU is known from the 'MediaSync' element referencing such MPU and is delivered to the presentation engine. The presentation time of the first access unit of the first segment described by the DASH MPD is also known from the 'MediaSync' elements referencing the relevant DASH MPD file and is delivered to the presentation engine.

4. The MPD parser receives the DASH MPD through the MMTP signalling message parser and processes it. The MPD parser provides relative presentation time of each DASH segment to the presentation engine.

5. The FEC decoding buffer receives MMTP packets with dedicated packet_id and immediately copies packets to the de-jitter buffer. If there are any packets which are not received until the protection window time of the AL_FEC message signalled by the MMTP signal message parser, a repair operation is applied to the received packets. If packets are recovered, they are also immediately copied to the de-jitter buffer.

6. De-jitter buffer receives MMTP packets from FEC decoding buffer. The de-jitter buffer stores those packets until the fixed_end_to_end_delay provided by the HRBM message has passed since the time specified in the time_stamp field of each of the MMTP packets and copies them to the MMTP de-capsulation buffer.

7. The MMTP decapsulation buffer depacketizes MMTP packets received from the de-jitter buffer and reconstructs the MPUs. Reconstructed MPUs are delivered to media engines.

8. The HTTP engine makes GET requests with the URL received from the MPD parser through the broadband network and receives DASH segments as responses. Received DASH segments are delivered to the DASH segment buffer.

9. The DASH segment buffer receives DASH segments from the HTTP engine and delivers them to the media engine as soon as the processing of previously delivered DASH segments is finished.

10. The media engine receives MPUs and DASH segments and decode them with appropriate decoders. It delivers decoded results to the presentation engine.

11. The presentation engine receives the HTML 5 document from the MMTP signalling message parser. It renders media data from the media engine in the location specified by the HTML 5 document and at the time provided by the MPEG-H CI parser and the MPD parser.

6.5 MMT caching for effective bandwidth utilization

6.5.1 Overview of MMT caching middlebox architecture

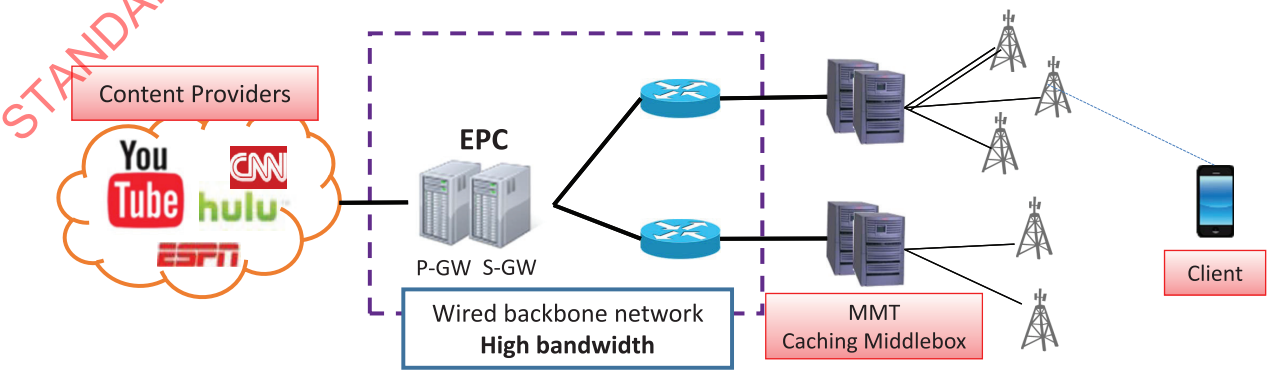


Figure 39 — Overview of an MMT caching middlebox

The network is composed of three entities: a server (or a content provider), a caching middlebox, and a client (see Figure 39). From the perspective of a client, the caching middlebox works transparently such that the client cannot tell a middlebox from a server. We assume that the server knows the existence of the caching middlebox.

The goal of the caching middlebox is (a) to reduce the bandwidth usage at the path between the server and the middlebox and (b) to adapt the video rate to available bandwidth per each client. For the former, the middlebox focuses on network redundancy elimination on the MPU delivery. It attempts to fetch all MPUs that belong to a requested video from the server for example by using HTTP, but actually downloads only those MPUs that do not exist in the middlebox cache (cache-missed MPUs). To identify an MPU, it uses an MPU ID which is a hash value of the content of the MPU.

For serving the media to clients, the middlebox adapts the MPUs to the available bandwidth per client, and fairly distributes the bandwidth to each client. That is, the middlebox uses MMTP to deliver the MPUs to the clients.

Moreover, deploying MMT caching middleboxes along with an HTTP CDN is costly since the CDN has to support MMT instead of HTTP. For easy deployment, it is desirable that the MMT caching middlebox cooperates with HTTP (caching) servers so that they communicate in HTTP.

6.5.2 Content-based caching of MMT media

6.5.2.1 Content-based caching

Content-based caching caches a content by its chunks instead of by the whole content. It divides the content into smaller chunks, and names each chunk using its content hash. Each chunk is a unit of caching, but its chunk name (or ID) is essentially a tightly-bound summary of the chunk content. So, using the chunk names, one can easily identify the duplicates across contents with different names (e.g., aliases) or even the set of partially-redundant chunks among different contents. Content-based caching is used for MMT media delivery.

6.5.2.2 Caching unit

The MMT caching middlebox uses an MPU as a caching unit (chunk), so the chunking process is simple and fast.

6.5.2.3 MPU ID (chunk ID) generation

6.5.2.3.1 General

An MPU ID is used for determining a cache hit or a miss. An MPU ID should be calculated by hashing over the content binary by carefully excluding the metadata such as an asset ID or an MPU sequence number that are not bound to the video or audio content itself. In addition, the MPU ID should be calculated differently according to the type of an MPU.

6.5.2.3.2 MPU with timed media

Multiple mdat atoms could exist in a timed media. In that case, the MPU ID is conceptually a hash value over all mdat binaries stitched together in an MPU (see Figure 40).

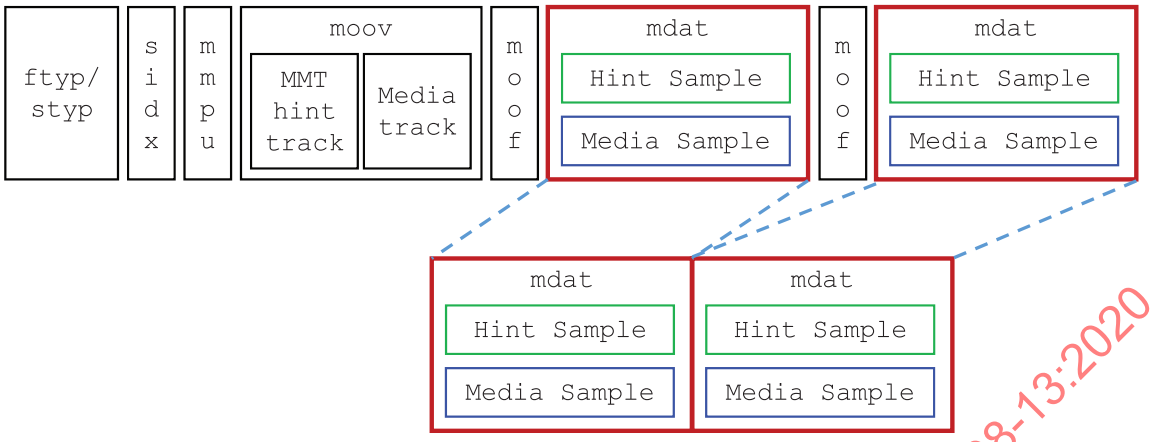


Figure 40 — MPU id generation for timed media

The actual MPU ID is calculated over payloadized mdat sequences. There are two reasons behind it. First, the middlebox can reduce undesirable delay if it fetches the MPU in the payloadized format since it can deliver each MFU right away without waiting for the full MPU. Second, payloadized mdat sequences would fix the location of the hint samples, which would produce the same ID (hash) even if those hint samples are stored in different locations at a storage (see Figure 41).

MPU ID = One-way hashing (payloadized mdat sequence)

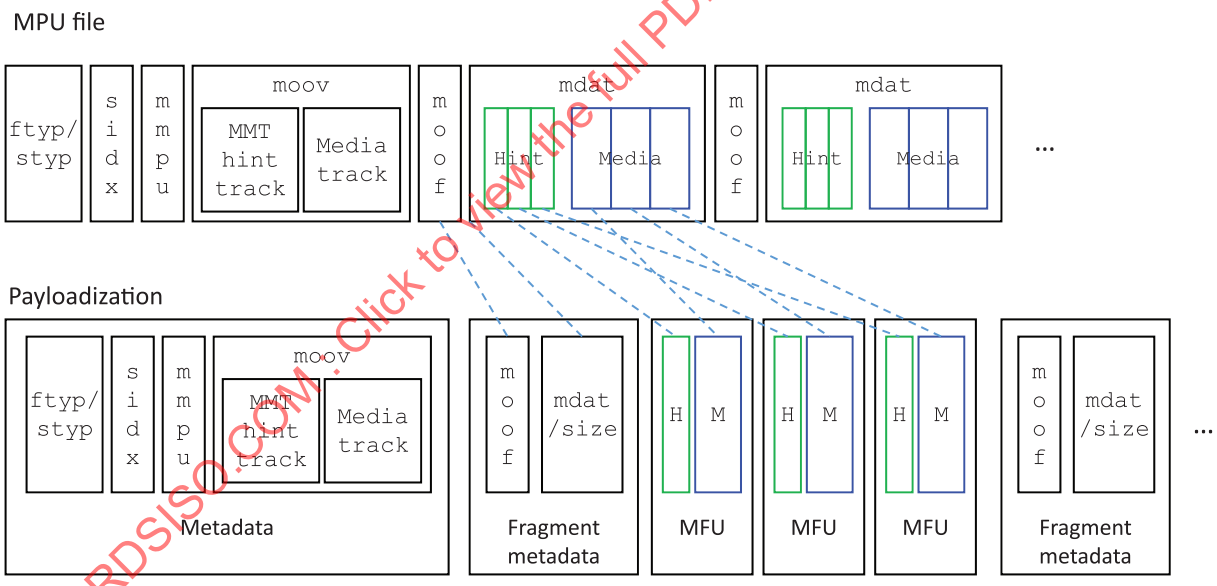


Figure 41 — Payloadized format

6.5.2.3.3 MPU with non-timed media

For non-timed media, an MPU ID refers to a hash over all items stitched together in an MPU (see Figure 42).

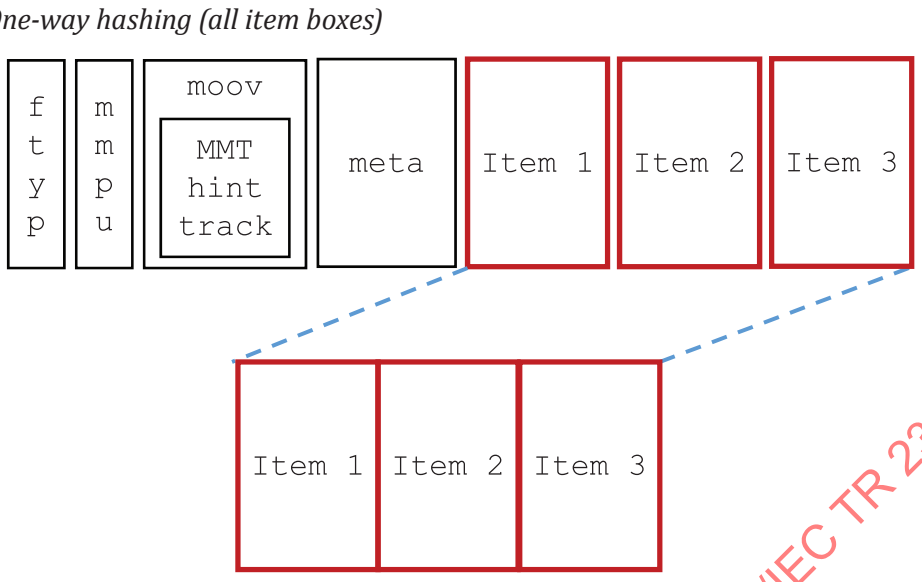


Figure 42 — MPU id generation for non-timed media

6.5.3 MPU sync protocol between server and caching middlebox

6.5.3.1 General

This subclause proposes a protocol based on a redundancy elimination technique for delivering MMT media from a server to a caching middlebox. It augments HTTP with a few custom headers for exchanging MPU requests, MPU IDs (chunk IDs), and metadata for delivering MMT packages.

6.5.3.2 Protocol overview

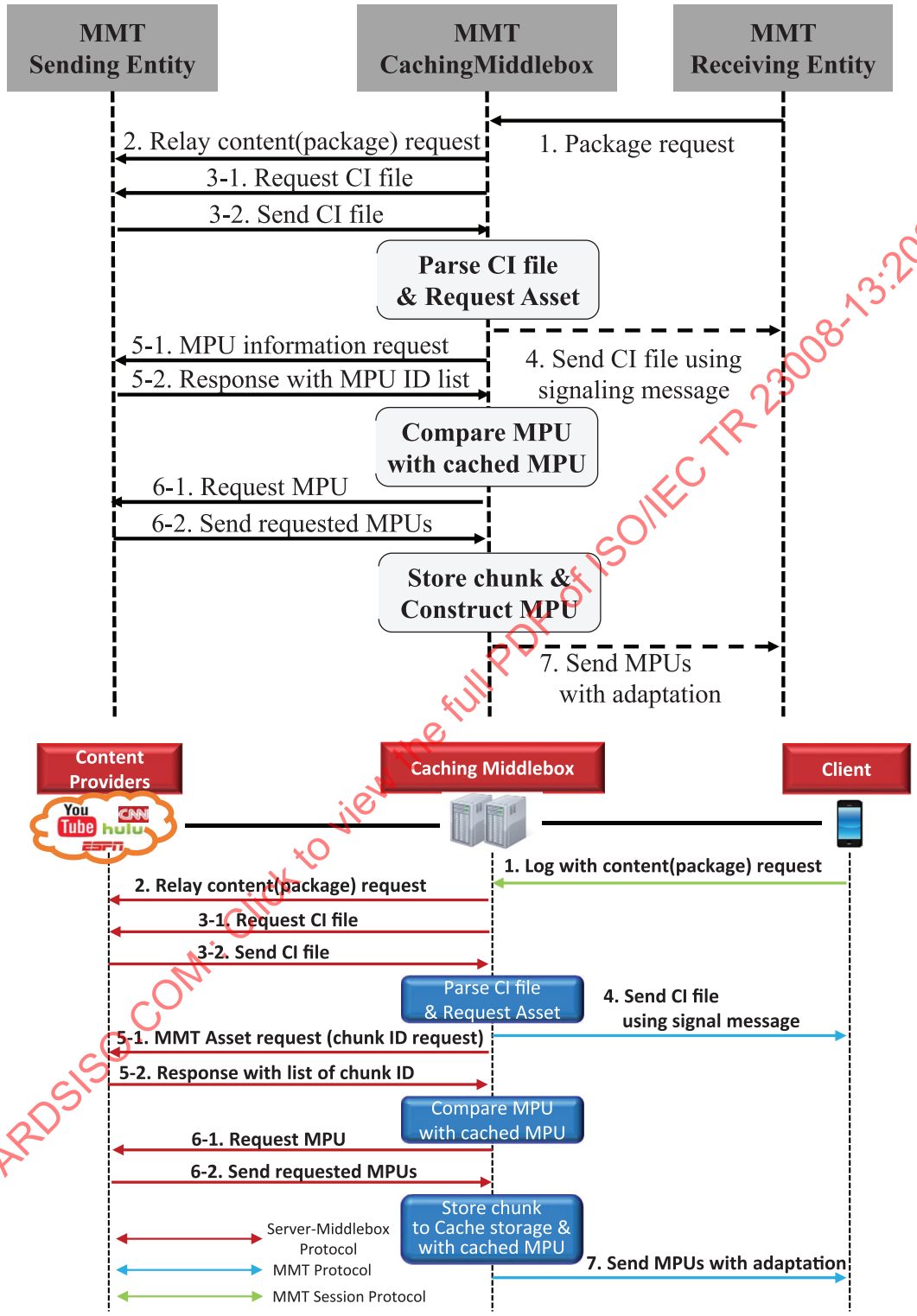


Figure 43 — Operation sequence

Figure 43 shows an overview of how the protocol works. Note that the following shows one implementation of the protocol using custom HTTP headers, but the same mechanism can be implemented using custom web services based on XML-RPC, JSON-RPC, RESTful API, etc.

First, in order to work as server, the middlebox asks for a CI file and an HTML file at the server using "*X-CIRequest*" and "*X-HTMLRequest*" HTTP custom headers. After parsing the CI and HTML files, the

middlebox retrieves the name of the asset, and requests MPU IDs for the MPUs in the asset, using the "X-MPUIDRange" header. On receiving the MPU IDs from the server, the middlebox determines whether the chunk content is cached for each ID. In case of a cache hit, the middlebox requests the header of the MPU that consists of non-mdat boxes (e.g,, moof and moov boxes) by using the "*X-MPUHeaderRequest*" header, and reconstructs the MPU in real time. On a cache miss, the middlebox asks for the entire MPU by using the "*X-MPURequest*" header and store the MPU content (e.g., mdat part) in cache storage along with its MPU ID.

6.5.3.3 Request messages

6.5.3.3.1 General

In this subclause, custom HTTP headers are defined as one example for requesting a CI file, a MPU ID list, an MPU, and an MPU header (see Table 1). The request and response messages follow the HTTP message format defined in RFC 2616:1997, Section 4[3]. These are an example of a possible implementation, and other web service mechanisms could also be used.

Table 1 — Request header fields

Request header =	X-CIRequest	; subclause 6.5.3.2.2
	| X-HTMLRequest	; subclause 6.5.3.2.3
	| X-MPUIDRange	; subclause 6.5.3.2.4
	| X-MPURequest	; subclause 6.5.3.2.5
	| X-MPUHeaderRequest	; subclause 6.5.3.2.6

6.5.3.3.2 X-CIRequest

X-CIRequest is used with the HTTP GET method to request for the CI file of an MMT package. The path of this request represents the name of the package. When the "*X-CIRequest*" field is present in the request, the server should provide a CI file of requested package.

*Device_id = 1*DIGIT*

X-CIRequest = "X-CIRequest" ":" device_id

Example:

```
GET /Package1 HTTP/1.1
Host: test.mmt.com
X-CIRequest: 1
```

6.5.3.3.3 X-HTMLRequest

X-HTMLRequest is used with the HTTP GET method to request the HTML file of an MMT package. The path of this request represents the name of the package. When the "*X-HTMLRequest*" field is present in a request, the server should provide an HTML file of a requested package.

*Device_id = 1*DIGIT*

X-HTMLRequest = "X-HTMLRequest" ":" device_id

Example:

```
GET /Package1 HTTP/1.1
Host: test.mmt.com
X-HTMLRequest: 1
```

6.5.3.3.4 X-MPUIDRange

X-MPUIDRange is used with the GET method to request the list of MPU IDs. The *X-MPUIDRange* field in a GET request specifies two numbers that represent the start and end sequence numbers of MPUs, and

the response should contain the MPU IDs that correspond to those MPUs in the range. The path in the first line represents the asset name.

MPURange-specifier = MPU_SeqNumStart "-" MPU_SeqNumEnd

*MPU_SeqNumStart = 1*DIGIT*

*MPU_SeqNumEnd = 1*DIGIT*

X-MPUIDRange = "X-MPUIDRange" ":" MPURage-specifier

Example:

```
GET /Package1/Asset1 HTTP/1.1
Host: test.mmt.com
X-MPUIDRange: 1-4
```

6.5.3.3.5 X-MPURequest

X-MPURequest is used with the GET method to request a whole payloadized MPU from a server. The *X-MPURequest* field specifies the sequence number of an MPU that is being requested, and the path in the first line represent the asset name.

*MPU_SeqNumber = 1*DIGIT*

X-MPURequest = "X-MPURequest" ":" MPU_SeqNumber

Example:

```
GET /Package1/Asset1 HTTP/1.1
Host: test.mmt.com
X-MPURequest: 3
```

6.5.3.3.6 X-MPUHeaderRequest

X-MPUHeaderRequest is used with the GET method to request the MPU header which consists of non-mdat boxes from a server. *X-MPUHeaderRequest* specifies the sequence number of an MPU whose header is being requested, and the path in the first line represents the asset name.

*MPU_SeqNumber = 1*DIGIT*

X-MPUHeaderRequest = "X-MPUHeaderRequest" ":" MPU_SeqNumber

Example:

```
GET /Package1/Asset1 HTTP/1.1
Host: test.mmt.com
X-MPUHeaderRequest: 3
```

6.5.3.4 Response messages (see Table 2)

6.5.3.4.1 General

Table 2 — Response header fields

Response header =	X-MPUIDRange	; subclause 6.5.3.3.2
	| X-MPUHeader	; subclause 6.5.3.3.3
	| X-MPUContent	; subclause 6.5.3.3.4

6.5.3.4.2 X-MPUIDRange

If X-MPUIDRange is used in a response, it specifies the range of the requested MPU IDs in the response. Moreover, *X-MPUIDRange* could be used without the request as well. The length represents the total number of chunks in the requested asset. If the total number of chunks is unknown (e.g., live media),

the length is set to -1. Each line in the response body consists of a sequence number of an MPU, name of the hashing algorithm, and its MPU ID. The number of lines in the response body should match the range in the *X-MPUIDRange* header.

Header format:

MPURange-specifier = MPU_SeqNumStart "-" MPU_SeqNumEnd "/" Length

*MPU_SeqNumStart = 1*DIGIT*

*MPU_SeqNumEnd = 1*DIGIT*

*Length = 1*DIGIT*

X-MPUIDRange: "X-MPUIDRange" ":" MPURange-specifier

Body Format:

```
MPU_SeqNumber Hash_scheme Hash_value(MPU ID)
MPU_SeqNumber Hash_scheme Hash_value(MPU ID)
```

Example:

```
HTTP/1.1 200 OK
X-MPUIDRange: 1-4/10
Content-Length: 48

1 SHA-1 2341242
2 SHA-1 2421245
3 SHA-1 2241244
4 SHA-1 1234124
```

6.5.3.4.3 X-MPUHeader

X-MPUHeader is a response header that specifies the MPU sequence number, and payloadized MPU metadata is carried in the response body. This is used as a response to X-MPUHeaderRequest in the request.

Header format:

*MPU_SeqNumber = 1*DIGIT*

X-MPUHeader = "X-MPUHeader" ":" MPU_SeqNumber

Body format:

```
Payloadized MPU metadata
```

Example:

```
HTTP/1.1 200 OK
X-MPUHeader: 3
Content-Length: 2879097

…ftyp…mp41…mmpu…moov…
```

6.5.3.4.4 X-MPUContent

X-MPUContent is a response header that specifies the MPU sequence number whose payloadized full content is carried in the response body. This is used in response to X-MPURequest in the request.

Header format:

*MPU_SeqNumber = 1*DIGIT*

X-MPUContent = "X-MPUHeader" ":" MPU_SeqNumber

Body format:

```
Payloadized full MPU content in the response
```

Example:

```
HTTP/1.1 200 OK
X-MPUContent: 2
Content-Length: 2879097

…ftyp…mp41…mmpu…
```

6.5.4 MMT cache manifest

The data path for cache in video broadcasting over the internet is illustrated in Figure 44. Directly serving each end user from the originating CDN server is not feasible for several reasons:

— Computing power: the server computing power is limited, can only support a limited number of concurrent connections,

— Out-going bandwidth: the original content server has n x B out-going bandwidth, to support n users consuming video at bit rate B. For emerging high bandwidth applications like UHD, this can easily eat up the bandwidth, e.g., B=16 Mbps, to serve 1000 user will require 16 G out-going bandwidth.

— End-to-end uncertainty: once the packet left the originating server, the current state of the internet cannot guarantee throughput nor delay, in fact, this is the uncertainty that will stay with us for a long time

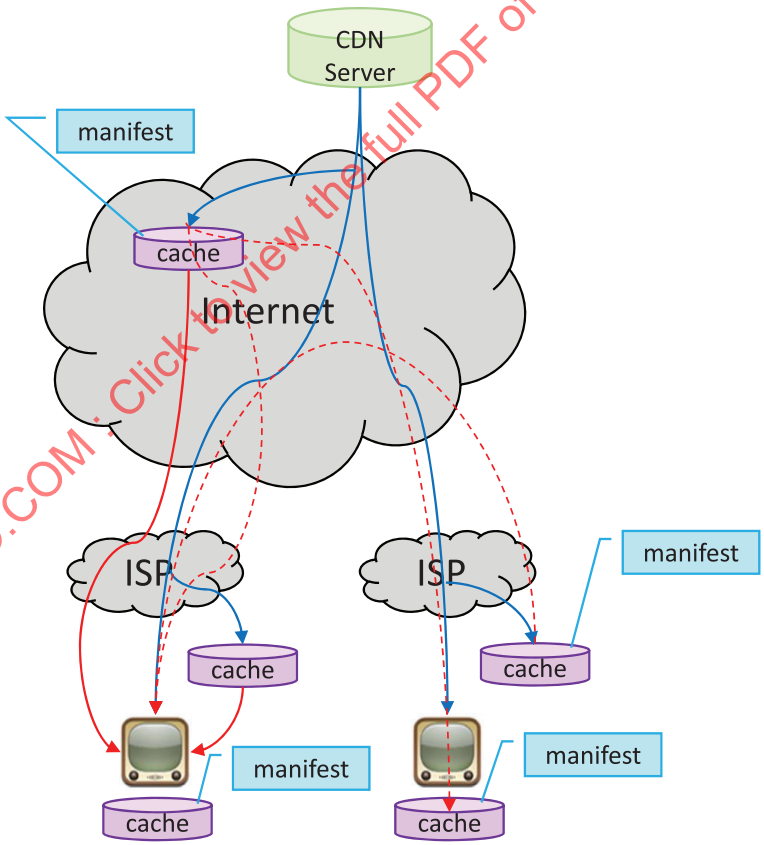


Figure 44 — Cache in Internet Video Delivery

By utilizing caches at various places in the network, instead of serving the end user from the originating server, cache hosts can do the job, with usually much shorter RTT and alternative delivery path that may have less congestion.

Content at the originating server is well behaving a predictable, as it is becoming available linearly over time. But for content at caches, such characteristics are not the case. Due to user behaviour, and the

different start time of streaming, different cache may contain different segments of content, and with different sub-representations (in DASH terminology) with different quality and QoS requirement. A hypothetical simple case without fragmentation is illustrated in Figure 45.

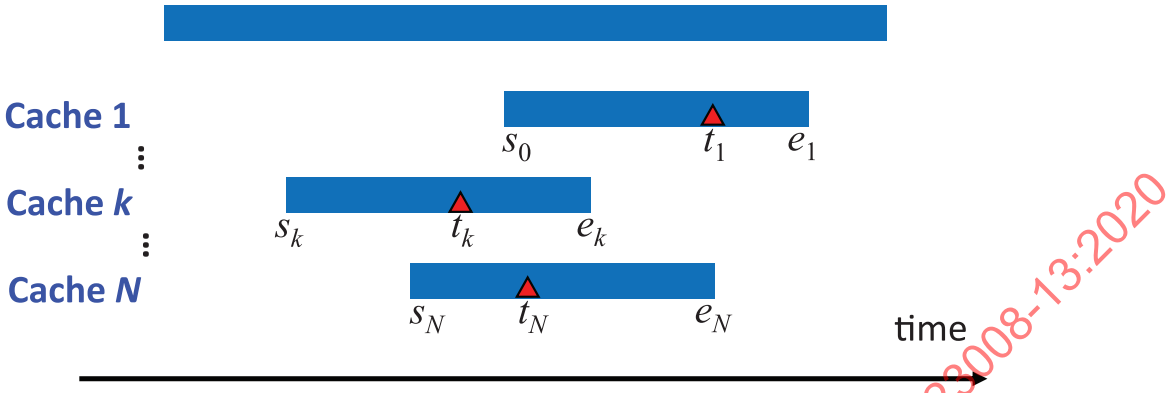


Figure 45 — Cache status

To facilitate cache assisted streaming, at least the content start and end timestamp, {sk, ek} current playback timestamp, *tk*, need to be signalled. For the same segment, when different sub-representations are available, their associated MFU index, resulting quality, and QoS info need to be signalled, similar to the multiple operating points ADC case in streaming. The MMT cache manifest is organized around asset ids, and then for each asset, those segments are present in the cache, listed by timestamp/fingerprint. For each segment, a new data structure, segment manifest or SegMf, can be introduced, which in turn consists of multiple rate-distortion operating points, each is represented by MFU index, which references a flat mdat byte file, and associated QoS parameters like average rate, peak rate; and QoE parameters signalling the spatio-temporal quality of the segment conforming to the ISOBMFF quality metrics. Notice that the MFU index is internal to the cache operation, and an API should be provided to retrieve associated MFUs from the cache host (see Figure 46).

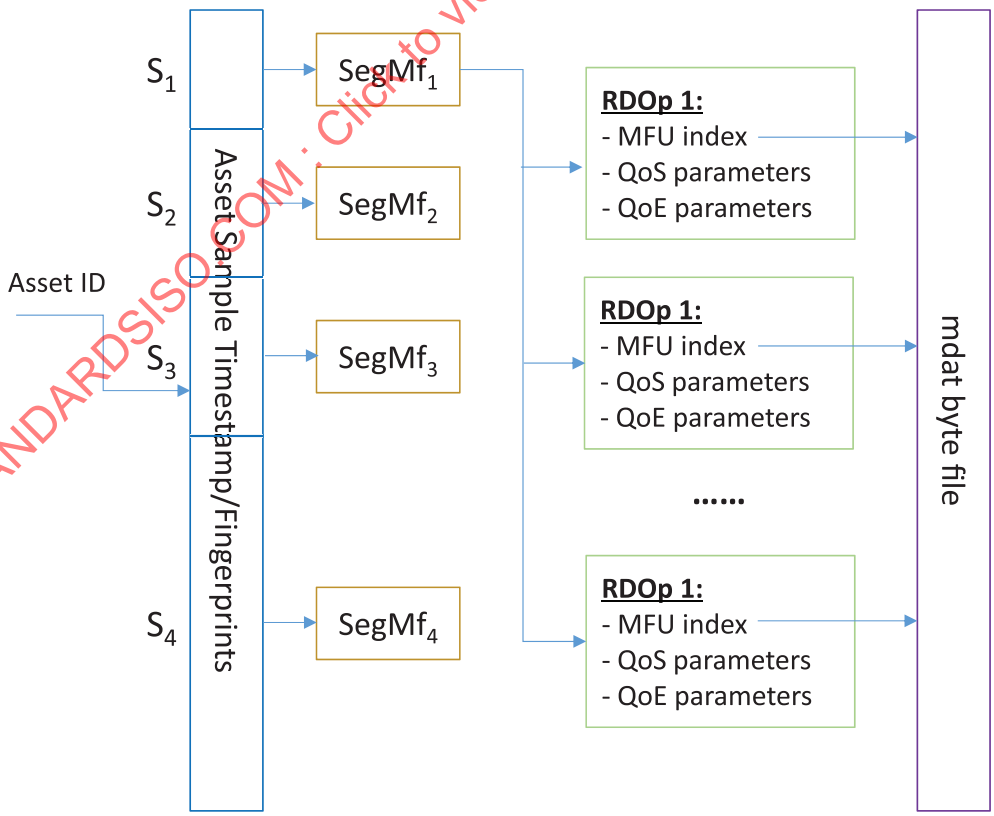


Figure 46 — MMT cache illustration

6.6 Usage of ADC signalling message

6.6.1 General

This subclause provides MMT implementation information for what an MMT sending entity should do and what a MANE can do to gain a benefit from ADC message.

6.6.2 Operation in MMT sending entity

An MMT sending entity can send the ADC information of an asset to MANEs and receiving entities by ADC signalling messages. The ADC message is made from the ADC parameters by MMT sending entity for assets within an MMT packet flow.

To provide more accurate information, an MMT sending entity can update the ADC message with parameter values that describes current asset characteristics best and transmit it as tagged with newer *version* value. The update of values in an ADC message can be done periodically or aperiodically and will be sent by the MMT sending entity. Then, the valid period of the updated version of the ADC should also be provided to help the MANE router estimate how long the ADC message is valid and meaningful.

6.6.3 Operation in MANE router

The MANE router can make more efficient utilization of network resources through ADC signalling messages. The MANE router identifies which ADC message is describing which asset by comparing *packet_id* within the MMTP packet header and the ADC message. By checking the change of *version* field in an ADC message, the MANE can acknowledge there is an update of delivery characteristics of a corresponding asset which will be expected to be valid until *validity_duration*.

The ADC describes QoS requirements and statistics of assets for delivery. Even though the MMTP packet header has some QoS related fields, it provides only packet level information on how to handle it. However, the ADC message can provide much more additional information than described in the packet header, such as peak rate and required buffer sizes, etc. to MANE routers helping them expect delivery characteristics in media level, so it can provide a big picture of delivery characteristics about an asset. By inspecting ADC messages, the MANE router can guess the burstiness characteristics of each asset from the *peak_rate* and *sustainable_rate* fields in advance, which is hardly provided by the packet header. Then MANE router can estimate how much buffer the MANE should secure to absorb the burstiness of that asset during *validity_duration*.

Moreover, the MANE router can characterize required network resources based on the corresponding updated ADC messages, for an MMT flow which is comprised of the flow of multiple assets. In case of the delivery of assets having delay constraints, network resources between the media source and destination can be reserved based on its peak-rate. However, it may result in inefficient bandwidth usage, because fixed bandwidth resource is dedicated to certain traffic while it is time-varying. The MANE can estimate actually what portion of those reserved resources will not be or is not being used from the ADC message. Then, the MANE can make use of those tentatively idle resources for other traffics if available, by its own decision.

6.6.4 Example operation in MMT-receiving entities

The received ADC can help MMT-receiving entities to estimate how much buffer they should actually secure for each asset (MMT sub-flow). As ADC information is provided with the update, the MMT receiving entity can also be updated with new buffer sizes to secure.

6.6.5 QoE multiplexing gain and bottleneck coordination

When multiple video sessions are sharing a congested link, the situation is called a bottleneck. Dealing with a bottleneck is the central theme of the multimedia transport as, when a network is not congested, all traffic QoS requests can be fulfilled and there is no need for traffic engineering.

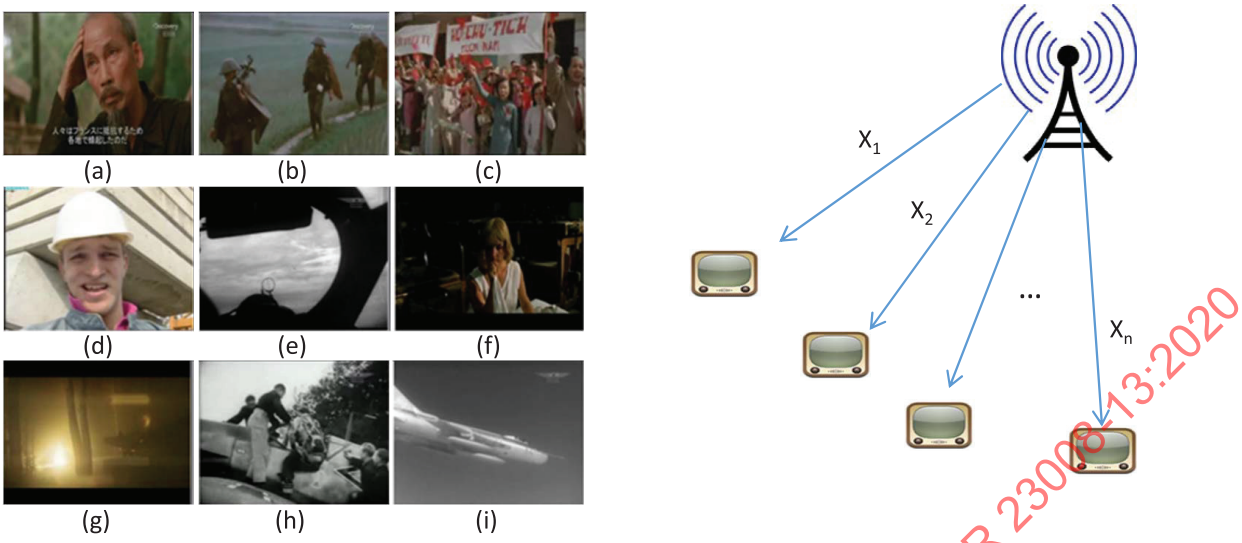


Figure 47 — Bottleneck QoE multiplexing gain problem

The problem is illustrated in Figure 47, multiple video traffics are sharing a capacity constrained link, $\sum_k x_k \leq C$, which is smaller than the total throughput required by the traffic, $\sum_k x_k > C$. The total resource constraint C can be estimated and provided from some sort of network nodes. For example, it can be an entity in the middle of the network such as a router in a wired network environment case and a cellular base station in a wireless case, etc. How to adapt each individual video stream, to meet this capacity constraint, while achieving the best QoE possible for all users, is the objective (see Figure 48). This problemis known as a QoE multiplexing gain problem.

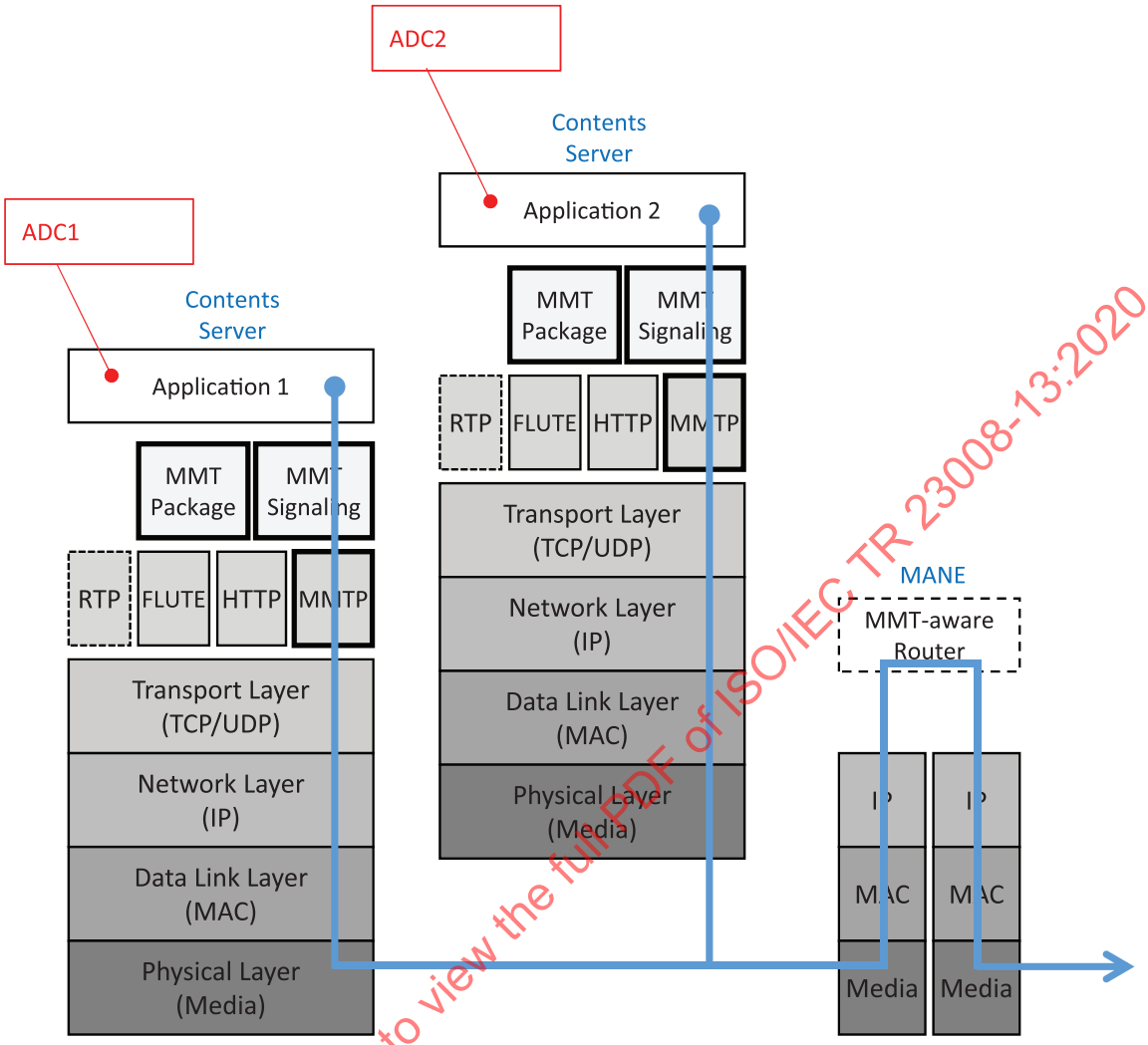


Figure 48 — MANE-based bottleneck coordination

To coordinate the transmission of video streams sharing the bottleneck link, a new MMT[2] function of bottleneck traffic orchestrator (BTO) is introduced. The BTO reads the ADC information of each video bitstream at the bottleneck, and construct the rate reduction-distortion incurred table, and compute the pruning index for each video session. This is supported by the multiple operating point ADC supported in MMT. An example of the rate reduction-distortion (RD) table computed for the 4 sec segments from the 9 video sequences illustrated in Figure 47 is shown in Figure 49.

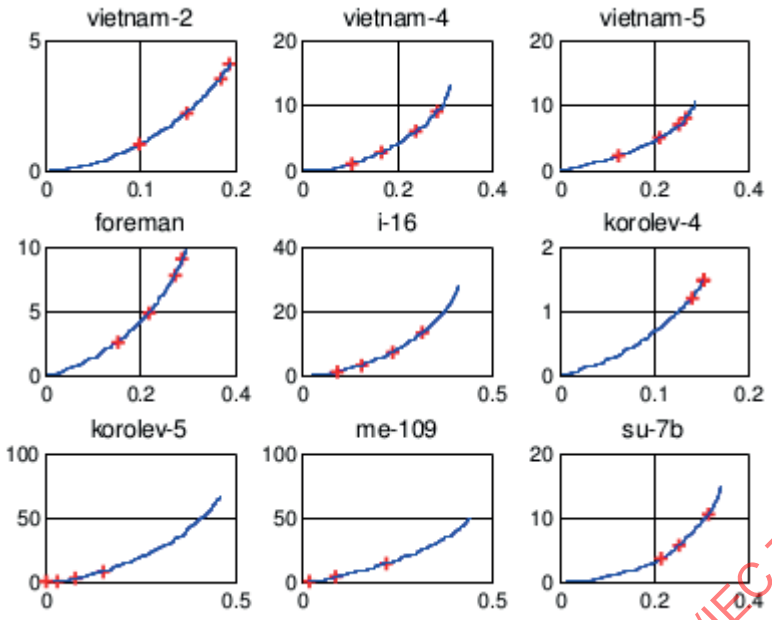


Figure 49 — Temporal distortion from frame drops and rate reduction plots

The syntax of signalling the rate-distortion trade-offs is specified in the MMT ADC, as in in ISO/IEC 23008-1.

The MMT aware routers, that can detect congestions and interpret the QoS/QoE operating points in the ADC messages, will act as the BTO. This is illustrated in Figure 49. The MANE node with MMT aware router, first detects the congestion from the transport layer, give an estimation of the approximate throughput reduction rate necessary to maintain QoS requirement, and requests QoS operating points from applications via ADC messages. Upon receiving these operating points, an utility maximization via gradient search is conducted to compute the optimal MFU drop for each application at the bottleneck, and conduct the thinning at MANE node.

There are a number of strategies the BTO can apply to achieve QoE multiplexing gains at the bottleneck. It can minimize the average distortion from pruning streams:

$$\min_{x_1, x_2, \ldots x_n} \sum_k D_k(x_k), s.t., \sum_k R_k(x_k) \leq C$$

where, x_k is the ADC operating point available for stream k, which is associated with a resulting frame loss induced distortion $D_k(x_k)$, and aggregated reduced bit rate of $R_k(x_k)$. The rate and distortion function for each stream, $\{D_k(), R_k()\}$ are carried in the ADC of each stream. The optimal solution can be easily found by a search on the ADCs of the bottlenecked streams. Upon computing the optimal operating points $\{x_1^*, x_2^*, \ldots, x_n^*\}$,they are communicated to the bottleneck MANE node, and the streams buffered are pruned to avoid congestion. Simulations on bottleneck coordination are performed on the 9 sequences as shown in Figure 47, the resulting playback demonstrated the graceful degradation of the quality when the bottleneck deficit increases, and in particular, the "easy" sequences helping out "busy" sequences in sharing the bottleneck resources.

6.7 MMT deployment in Japanese broadcasting systems

6.7.1 General

ARIB STD-B60 describes the media transport and multiplexing layers in broadcasting systems using MMT. It specifies the usage and extensions of ISO/IEC 23008-1 for broadcasting services. The extensions include the MMTP packet header and additional signalling information. Some of the

signalling information used in the MPEG-2 TS based broadcasting systems has been transplanted to the MMT-based broadcasting system.

This subclause gives implementation examples of Japanese broadcasting system based on MMT.

6.7.2 Broadcasting systems using MMT

6.7.2.1 System structure

This subclause describes the general structure of MMT-based broadcasting systems. Figure 50 shows the protocol stack of MMT-based broadcasting systems.

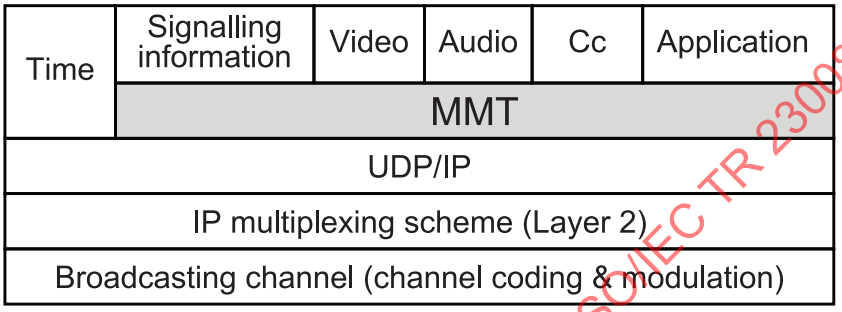


Figure 50 — Protocol stack of MMT-based broadcasting systems

In these systems, media components, such as video, audio, and closed captions (cc) constituting a TV programme are encapsulated into media fragment units (MFUs)/media processing units (MPUs). They are carried as MMT protocol (MMTP) payloads of MMTP packets and delivered in IP packets. Data applications that are related to a TV programme are also encapsulated into MFUs/MPUs, carried in MMTP packets, and delivered in IP packets.

IP packets generated like this are multiplexed over broadcasting channels with an IP multiplexing scheme, also referred to as a layer 2 (L2) protocol, e.g., the TLV multiplexing scheme described in Recommendation ITU-R BT.1869.

The systems also have MMT signalling information (MMT-SI). MMT-SI is signalling information on the structure of a TV programme and associated information on TV services like the electronic programme guide (EPG). MMT-SI is carried in MMTP packets and delivered in IP packets.

In order to provide coordinated universal time (UTC) in broadcasting systems, time information is also delivered in IP packets.

6.7.2.2 Service configuration in a broadcasting channel

ISO/IEC 23008-1 specifies the MMT package as a logical structure of content. The MMT package includes presentation information and associated assets that constitute content.

A broadcasting service is generally a series of TV programmes. In MMT-based broadcasting systems, one MMT package corresponds to one broadcasting service. The relationship between the broadcasting service and the MMT package is shown in Figure 51. As shown in Figure 51, one TV programme is distinguished from the rest of the service by its start and end times and corresponds to one event.

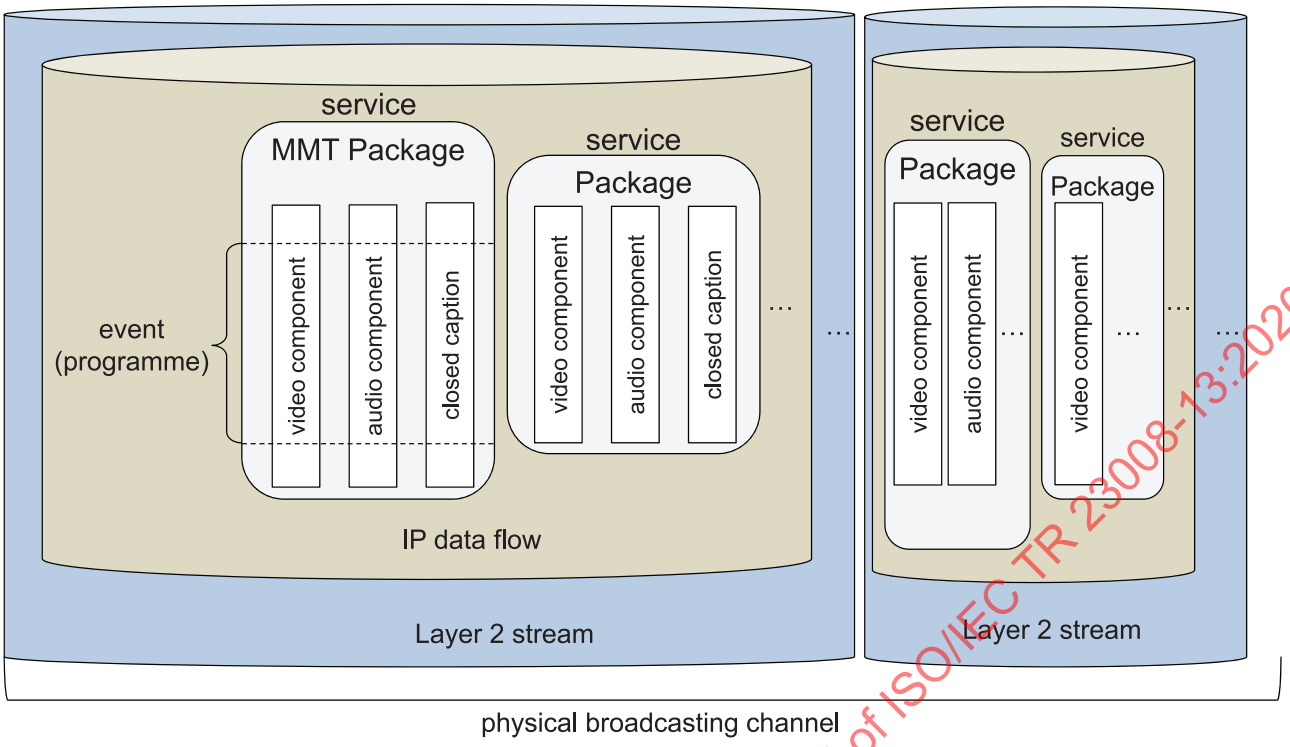


Figure 51 — Relationship between a broadcasting service and MMT package

In ISO/IEC 23008-1, an asset is defined as a media component. An asset is equivalent to a series of MPUs. In MMT-based broadcasting systems, one TV programme is an MMT package including one or more assets and signalling information. A package access (PA) message is an MMT-SI, and the MMT package table (MPT) carried in the PA message identifies assets constituting the TV programme.

Multiple MMT packages can be delivered in one IP data flow, as shown in Figure 51. Here, an IP data flow is defined as a sequence of IP packets of which the source IP address, destination IP address, protocol, source port number, and destination port number are the same combination. There may be other IP data flows carrying content for download services or extended services in addition to IP data flows carrying MMT packages.

Multiple IP data flows might be multiplexed into one layer 2 stream. The layer 2 stream includes signalling information for demultiplexing IP packets from broadcasting signals.

6.7.2.3 Service configuration in broadcasting channels and broadband networks

ISO/IEC 23008-1 has been developed to support delivery of media data over heterogeneous networks including broadcasting channels and broadband networks. In the MMT specifications, broadcasting channels and broadband networks can be treated in the same way for delivery of content. Figure 52 shows a service configuration using both broadcasting channels and broadband networks.

In Figure 52, video component 1, audio component 1, and closed caption 1 are delivered on broadcasting channels. In addition to these components, video component 2, audio component 2, and closed caption 2 are delivered on broadband networks.

In the broadcasting channels, the three components are multiplexed into one IP data flow and delivered in one layer 2 stream, since all transmitted information is delivered to all receiver terminals. On the other hand, in the broadband networks, components are delivered as a separate IP data flow, since each component is delivered to the receiver terminal requesting it.

In MMT-based broadcasting systems, media components delivered in different channels can easily be included in one MMT package. MMT-based broadcasting systems support hybrid delivery of multimedia content.

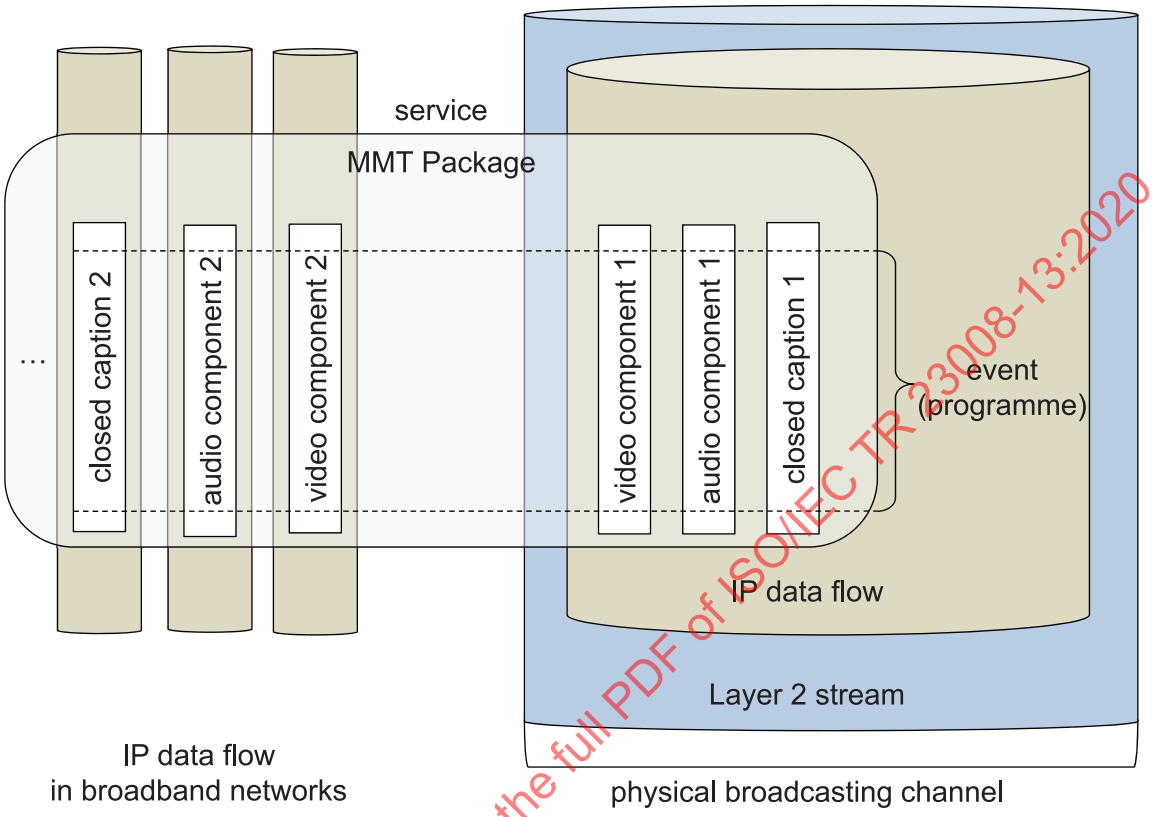


Figure 52 — Service configuration over both broadcasting channels and broadband networks

6.7.3 Media transport protocol

6.7.3.1 General

MMT-based broadcasting systems use the syntax and semantics of the MMTP payload and MMTP packet specified in ISO/IEC 23008-1. The extensions described below are applied.

6.7.3.2 Header extension of MMTP packets

ISO/IEC 23008-1 specifies a header extension in the MMTP packet. The header extension has three fields: extension_type, extension_length, and header_extension_value. Although the header extension can be used for various purposes, it contains only one piece of information. The multi-type header extension described below enables it to contain multiple pieces of information.

header_extension_value – When the extension_type field is set to 0x0000, this field has the structure shown in Table 3.

Table 3 — Structure of multi-type header extension

Syntax	No.of bits	Mnemonic
Header_extension_value {		
for (i=0; i<N; i++) {		
hdr_ext_end_flag	1	bslbf
hdr_ext_type	15	uimsbf
hdr_ext_length	16	uimsbf
for (j=0; j<M; j++) {		
hdr_ext_byte	8	bslbf
}		
}		
}		

hdr_ext_end_flag – When this flag is set to '1', this multi-type header extension is the end of the header extension. When this flag is set to '0', this multi-type header extension is not the end of the header extension.

hdr_ext_type – This field specifies the type of multi-type header extension. The value of hdr_ext_type is specified in Table 4.

Table 4 — Hdr_ext_type values

Value	Description
0x0000	reserved for future use
0x0001	reserved for ARIB STD-B61 (scrambling information)
0x0002	reserved for ARIB STD-B60 (download_id)
0x0003 – 0x7FFF	reserved for future use

hdr_ext_length – This field specifies the number of bytes of the following hdr_ext_byte field.

hdr_ext_byte – This field provides information on multi-type header extension.

6.7.3.3 Encapsulation of multimedia data

6.7.3.3.1 General

In order to improve the inter-operability of MMT-based broadcasting systems, the following constraints apply to carriage of multimedia data in MMTP packets.

6.7.3.3.2 Encapsulation of video data

6.7.3.3.2.1 MFU format for HEVC stream

When a high efficiency video coding (HEVC) stream is carried in the MMT protocol, input to the MMT process is a sequence of network abstraction layer (NAL) units. A NAL unit is encapsulated into an MFU when an HEVC stream is carried in the MMT protocol.

If an HEVC encoder generates the byte stream format specified in Rec. ITU-T H.265 | ISO/IEC 23008-2:2017 Annex B, one start code prefix (0x000001) followed by one NAL unit is replaced with 32-bit length information of the NAL unit (unsigned integer format). Namely, the NAL unit together with the length information are encapsulated into one MFU.

Figure 53 shows an overview of generating MMTP packets and MFUs from a sequence of NAL units output from an HEVC encoder.

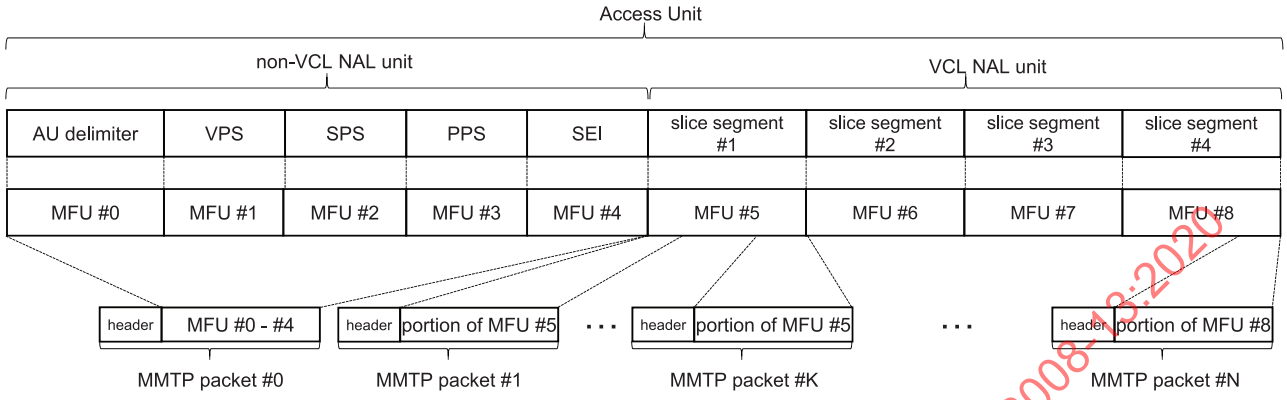


Figure 53 — Overview of packetization of NAL units of HEVC streams

The duration of video MPU greatly influences the channel change time at the receiver terminal, since the video stream is decoded and presented at the receiver terminal on a per MPU basis. In order to reduce the channel change time, the MPU of an HEVC stream is constructed in intra random access point (IRAP) intervals.

6.7.3.3.2.2 Encapsulation of HEVC bitstream subsets

HEVC supports temporal sub-layer coding. One example is that when 120-Hz video is encoded, two streams can be generated: one is a sub-bitsream for 60-Hz video; the other is a bitstream subset for 120-Hz video. At the receiver terminal, 60-Hz video can be decoded from the sub-bitstream, and 120-Hz video can be decoded from both the sub-bitstream and the bitstream subset.

Figure 54 shows an overview of encapsulation of HEVC bitstream subsets. Note that this figure shows display order frame sequence. When an MMT package is made up of various media components, the sub-bitstream and the bitstream subset are encapsulated into separate assets. In Figure 54, the sub-bitstream is encapsulated into asset 1 and the bitstream subset is encapsulated into asset 2. Since they are separate assets, the access units of asset 1 and asset 2 are carried in MMTP packets that have different packet IDs.

The sequence number of an MPU that the access units of the bitstream subset belong to is identical to the sequence number of an MPU that the access units of the sub-bitstream belong to in the same time period. Assigning the same sequence number to both MPUs enables receiver terminals to easily identify the MPUs that include corresponding access units in the same GOP.

In the example shown in Figure 54, the decoding of asset 2 depends on asset 1. A dependency descriptor stating that asset 2 depends on asset 1 is inserted in the asset_descriptors_byte field of the MP table. In addition to the dependency descriptor, an MPU timestamp descriptor and MPU extended timestamp descriptor are inserted in the asset_descriptors_byte fields of both asset 1 and asset 2.

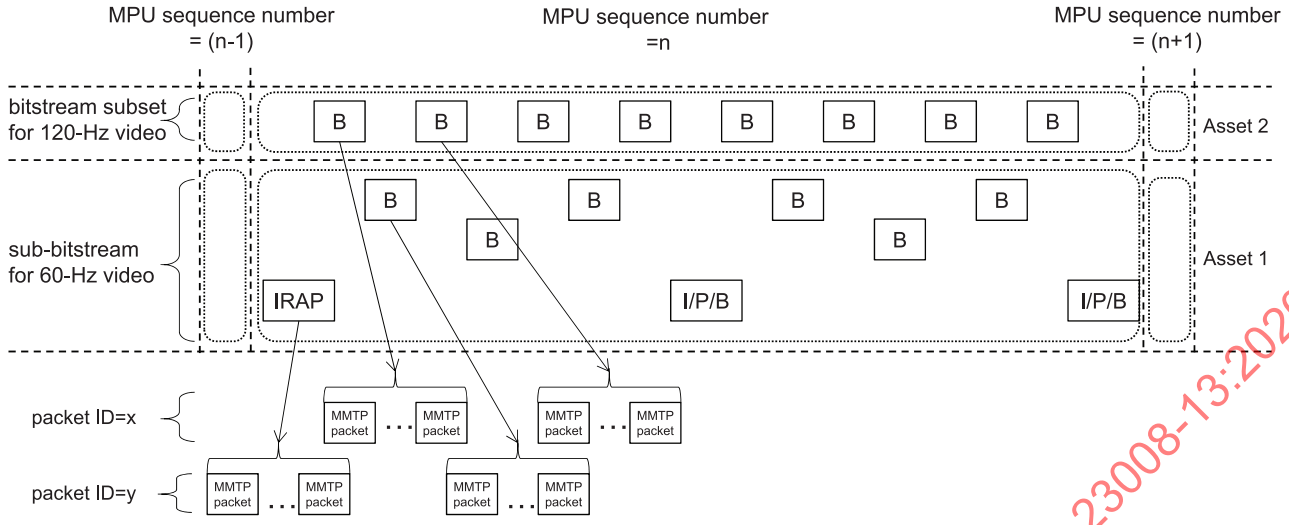


Figure 54 — Overview of encapsulation of HEVC sub-bitstream and bitstream subsets for temporal sub-layer coding

6.7.3.3.2.3 Clean random access of HEVC bitstream (for SAP Type 3)

For enabling a clean random access of SAP Type 3 in HEVC bitstream, the following operation in system-level is recommended.

When a random access is occured at the random access point with a clean random access (CRA) or broken link access (BLA) picture in an HEVC bitstream, the associated RASL pictures are discarded from the HEVC bitstream before fed into an HEVC decoder. The NAL units of the CRA/BLA and the RASL pictures can be identified by NAL unit type (NUT) which is signalled in NAL unit header (NUH) or in HEVCDecoderConfigurationRecord of HEVCConfigurationBox defined in RFC 2136[4]. When the NUTs of NAL units at the random access point are CRA, the variable HandleCraAsBlaFlag of the HEVC decoder is set equal to 1, or the NUHs of the NAL units of the CRA picture are rewritten by changing the values of NUTs from 21 to 17 in the HEVC bitstream. This is because the CRA picture should be handled as a BLA picture in the HEVC decoder, when a random access is occurred. The decoded picture buffer (DPB) of the HEVC decoder is refreshed before the CRA/BLA picture is decoded.

6.7.3.3.2.4 Fast random access of HEVC bitstream (for SAP Types 2 and 3)

For enabling a fast random access of SAP Types 2 and 3 in HEVC bitstream, the following operation in system-level is recommended.

When a random access is occurred at a random access point with an IRAP picture, the presence of leading pictures is inspected by parsing the NUTs of NAL units associated with the IRAP pictures in the HEVC bitstream. The NUT is signalled in NAL unit header (NUH) or in HEVCDecoderConfigurationRecord of HEVCConfigurationBox defined in RFC 2136[4]. If the associated leading pictures are present in the HEVC bitstream, all the associated NAL units marked as RADL (NUT 6, 7) and RASL (NUT 8, 9) are discarded from the HEVC bitstream. After the IRAP picture is decoded, the decoded picture of the IRAP is immediately displayed by ignoring its original presentation time, and repeatedly displayed until the following trail picture is presented. The decoded picture buffer (DPB) of the HEVC decoder is refreshed before the IRAP picture is decoded.

According to Rec. ITU-T H.265 | ISO/IEC 23008-2:2017, 6.6.2.3.2.4, to support use of HEVC bitstream with SAP type 2 or 3 for MPU following operations should be done by the MMT server to create MMTP bitstream.

— The value of the R field of MMTP packets carrying MFUs of the pictures preceding the first IRAP picture of a movie fragment in decoding order, the MMTP packets whose value of the *movie_fragment_sequence_number* field is same with the value of the same field of the MMTP packets carrying MFUs

of the IRAP picture but the value of the *sample_number* field is smaller than the value of the same field of the MMTP packets carrying MFUs of the IRAP picture, is set to 0.

— The value of the R field of MMTP packets carrying MFUs of the IRAP picture of a movie fragment is set to 1.

Following operation should be done by MMTP decapsulation buffer for each MMTP packets carrying MFUs of HEVC bitstream, the value of the FT field is equal to 2, to support use of HEVC bitstream with SAP type 2 or 3.

— Step 1: When the MMTP packets whose value of the *movie_fragment_sequence_number* field is different from the value of the same field of the preceding MMTP packets is firstly received, such MMTP packets and all succeeding MMTP packets whose value of the R field is equal to 0 are immediately deleted from the MMTP decapsulation buffer. They are not delivered to the decoder buffer until the MMTP packets whose value of the *movie_fragment_sequence_number* field is the same as the value of the same field of the previously received MMTP packet but the value of the R field is equal to 1 are received.

— Step 2: When the MMTP packets whose value of the R field is equal to 1 is firstly received as described in step 1, such MMTP packet and all succeeding MMTP packets carrying MFUs belong to the same AU are processed by the MMTP decapsulation buffer and delivered to the decoder buffer.

— Step 3: After receiving MMTP packets whose value of the R field is equal to 1 is processed as described in step 2, all succeeding MMTP packets carrying MFUs whose value of the *movie_fragment_sequence_number* field is the same as the value of the same field of the MMTP packets described in step 2 are immediately deleted from the MMTP decapsulation buffer. They are not delivered to the decoder buffer until the MMTP packet whose value of the *movie_fragment_sequence_number* field is the same as the value of the same field of the MMTP packets described in step 2 but the value of the *dep_counter* field is 0 is received. Such an MMTP packet is also immediately deleted from the MMTP decapsulation buffer and it is not delivered to the decoder buffer.

— Step 4: After receiving the MMTP packet whose value of the *dep_counter* field is 0 as described in step 3, the following operation to all succeeding MMTP packets carrying MFUs whose value of the *movie_fragment_sequence_number* field is the same as the value of the same field of the MMTP packets described in step 3 is applied:

  — If the value of the *f_i* field of the MMTP packet is equal to 00, all data units of such an MMTP packet whose value of the *dep_counter* field is 0 are immediately deleted from the MMTP decapsulation buffer and they are not delivered to the decoder buffer until any data unit whose value of the *dep_counter* field is not equal to 0 is found.

  — If the value of the *f_i* field of the MMTP packet is not equal to 00, then this MMTP packet and all succeeding MMTP packets are stored in the MMTP decapsulation buffer and not processed until the MMTP packet whose value of the *f_i* field is equal to 11 is received. If MMTP packets whose value of the *f_i* field is equal to 11 are received and whose value of the *dep_counter* field is equal to 0, then all packets stored in the MMTP decapsulation buffer whose value of *movie_fragment_sequence_number* field and the value of the *sample_number* field are the same as such a packet are immediately deleted from the MMTP decapsulation buffer and they are not delivered to the decoder buffer.

  — If the value of the *f_i* field of the MMTP packet is not equal to 00, then this MMTP packet and all succeeding MMTP packets are stored in the MMTP decapsulation buffer and not processed until the MMTP packet whose value of the *f_i* field is equal to 11 is received. If MMTP packets whose value of the *f_i* field is equal to 11 are received and whose value of the *dep_counter* field is not equal to 0, then all packets stored in the MMTP decapsulation buffer whose value of *movie_fragment_sequence_number* field and the value of the *sample_number* field are the same as such a packet are processed by the MMTP decapsulation buffer and delivered to the decoder buffer

6.7.3.3.3 Encapsulation of audio data: MFU format for MPEG-4 AAC and MPEG-4 ALS

When an MPEG-4 advanced audio coding (AAC) stream or MPEG-4 audio lossless coding (ALS) stream is carried in the MMT protocol, input to the MMT process is in the form of either LATM/LOAS stream or a data stream.

The low overhead audio transport multiplex (LATM) includes an audio channel configuration and provides multiplexing functions for audio data. The low overhead audio stream (LOAS) provides synchronization for audio data. When an audio encoder generates a LATM/LOAS stream, one AudioMuxElement () specified in ISO/IEC 14496-3 is encapsulated into one MFU.

When an audio encoder generates a data stream, a raw data stream is encapsulated into one MFU.

6.7.4 Signalling information

6.7.4.1 General

There are three kinds of MMT-signalling information: message, table, and descriptor. Some of the signalling information specified in ISO/IEC 23008-1 is not used in broadcasting systems. This subclause summarizes the signalling information essential to broadcasting systems. Additional signalling information may be used in broadcasting systems.

6.7.4.2 MMT signalling information messages

6.7.4.2.1 List of MMT-signalling information messages

Table 5 shows the list of messages.

Table 5 — List of messages

Message name	Message_id assignment	Description	Specified in ISO/IEC 23008-1	Use in broad-casting systems
Package access (PA) message	0x0000	Is the entry point of MMT-signalling information. Conveys one or more tables.	X	X
Media presentation information (MPI) message	0x0001 – 0x000F	Conveys a presentation information document.	X	
MMT package table (MPT) message	0x0010 – 0x001F	Conveys a whole or a subset of an MP table.	X	
Clock relation information (CRI) message	0x0200	Conveys clock related information to be used for mapping between the NTP timestamp and MPEG-2 STC.	X	
Device capability information (DCI) message	0x0201	Conveys information on required device capabilities for the package consumption.	X	
Application layer-forward error correction (AL-FEC) message	0x0202	Conveys configuration information of an AL-FEC scheme to be used to protect asset.	X	
Hypothetical receiver buffer model (HRBM) message	0x0203	Conveys information on end-to-end transmission delay and memory requirement to a receiving terminal.	X	

Table 5 *(continued)*

Message name	Message_id assignment	Description	Specified in ISO/IEC 23008-1	Use in broad-casting systems
M2section mes-sage	0x8000	Conveys the MPEG-2 section-format table. Tables and descriptors in MPEG-2 TS based conventional broadcasting systems can be reused by this message.		X

6.7.4.2.2 Detailed specifications of messages

6.7.4.2.2.1 PA message

The syntax and semantics of the PA message are specified in ISO/IEC 23008-1.

6.7.4.2.2.2 M2section message

Table 6 shows the syntax of the M2section message.

Table 6 — Syntax of M2section message

Syntax	No.of bits	Mnemonic
M2section_Message () {		
message_id	16	uimsbf
version	8	uimsbf
length	16	uimsbf
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'1'	1	bslbf
'11'	2	bslbf
section_length	12	uimsbf
table_id_extension	16	uimsbf
'11'	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for(i=0; i<N; i++) {		
signalling_data_byte	8	bslbf
}		
CRC_32	32	rpchof
}		

The semantics of each field of the M2section message are as follows.

table_id – This field identifies the table to which the section belongs.

section_syntax_indicator – This field determines whether a normal or extension format is used. This field is always set to "1" to indicate the extension format.

section_length – This field identifies the number of data bytes following this field.

table_id_extension – This is a field extending the table identifier.

version_number – This field contains the table version number.

current_next_indicator – This field contains "1" when the table is currently used and "0" when the table cannot be used at present, but can be used next.

section_number – This field contains the number of the first section comprising the table.

last_section_number – This field contains the number of the last section comprising the table.

CRC_32 – This field complies with Recommendation ITU-T.

6.7.4.3 MMT signalling information tables

6.7.4.3.1 List of MMT-signalling information tables

Table 7 shows the list of tables.

Table 7 — List of tables

Table name	Table_id assignment	Description	Specified in ISO/IEC 23008-1	Use in broad-casting systems
Package access (PA) table	0x00	Provides information on all other signalling tables.	X	
Media presentation information (MPI) table	0x01 – 0x0F	Provides a presentation information document.	X	
MMT Package (MP) Table	0x20	Provides configuration information on the MMT package, such as lists and locations of assets.	X	X
Clock relation information (CRI) table	0x21	Provides a CRI descriptor.	X	
Device capability information (DCI) table	0x22	Provides information on the required device capabilities for consumption of the package.	X	
Package list table	0x80	Provides the IP data flow and packet id of the PA message for the MMT package as a broadcasting service. Also provides a list of IP data flows of other IP services.		X

6.7.4.3.2 Detailed specifications of tables

6.7.4.3.2.1 MMT package table

The syntax and semantics of the MMT package table are specified in ISO/IEC 23008-1.

6.7.4.3.2.2 package list table

Table 8 shows the syntax of the package list table.

Table 8 — Syntax of package list table

Syntax	No.of bits	Mnemonic
Package_List_Table () {		
table_id	8	uimsbf
version	8	uimsbf
length	16	uimsbf
num_of_package	8	uimsbf
for (i=0; i<N; i++) {		
MMT_package_id_length	8	uimsbf
for (j=0; j<M; j++) {		
MMT_package_id_byte	8	bslbf
}		
MMT_general_location_info ()		
}		
num_of_ip_delivery	8	uimsbf
for (i=0; i<N; i++) {		
transport_file_id	32	uimsbf
location_type	8	uimsbf
if (location_type == 0x01) {		
ipv4_src_addr	32	uimsbf
ipv4_dst_addr	32	uimsbf
dst_port	16	uimsbf
}		
if (location_type == 0x02) {		
ipv6_src_addr	128	uimsbf
ipv6_dst_addr	128	uimsbf
dst_port	16	uimsbf
}		
if (location_type == 0x05) {		
URL_length	8	uimsbf
for (j=0; j<M; j++) {		
URL_byte	8	char
}		
}		
descriptor_loop_length	16	uimsbf
for (j=0; j<M; j++) {		
descriptor ()		
}		
}		
}		

The semantics of each field of the package list table are as the follows.

num_of_package – This field identifies the number of packages whose locations are described in this Table.

MMT_package_id_length – This field specifies the number of bytes of the following MMT_package_id_byte field.

MMT_package_id_byte – This field identifies the MMT package ID.

MMT_general_location_info – This field indicates the location information carrying the PA message of the identified MMT package.

num_of_ip_delivery – This field specifies the number of IP flows whose locations are described in this table.

transport_file_id – This field specifies the identification of file object.

location_type – This field specifies the type of location information. When this field is set to 0x01, the location is an IPv4 data flow. When this field is set to 0x02, the location is an IPv6 data flow. When this field is set to 0x05, the location is a URL.

ipv4_src_addr – This field specifies an IPv4 source address. The IPv4 address is fragmented into four fields of 8 bits, where the first byte of this field contains the most significant byte of the IPv4 source address.

ipv4_dst_addr – This field specifies an IPv4 destination address. The IPv4 address is fragmented into four fields of 8 bits, where the first byte of this field contains the most significant byte of the IPv4 destination address.

dst_port – This field specifies the destination port number of an IP data flow.

ipv6_src_addr – This field specifies an IPv6 source address. The IPv6 address is fragmented into eight fields of 16 bits, where the first byte of this field contains the most significant byte of the IPv6 source address.

ipv6_dst_addr – This field specifies an IPv6 destination address. The IPv6 address is fragmented into eight fields of 16 bits, where the first byte of this field contains the most significant byte of the IPv6 destination address.

URL_length – This field specifies the number of bytes of the following URL_byte field.

URL_byte – This field specifies the URL.

descriptor_loop_length – This field represents the number of bytes in all descriptors immediately after this field.

6.7.4.4 MMT signalling information descriptors

6.7.4.4.1 List of MMT signalling information descriptors

Table 9 shows the list of descriptors.

Table 9 — List of descriptors

Descriptor name	Descriptor_tag value assignment	Description	Specified in ISO/IEC 23008-1	Use in broadcasting systems
Clock relation information (CRI) descriptor	0x0000	Provides the relationship between the NTP timestamp and the MPEG-2 STC for synchronization.	X	
MPU times-tamp descriptor	0x0001	Provides presentation time of MPU.	X	X
Dependency descriptor	0x0002	Provides asset identifications that depend on other assets.	X	X

Table 9 *(continued)*

Descriptor name	Descriptor_tag value assignment	Description	Specified in ISO/IEC 23008-1	Use in broadcasting systems
Generic file delivery table (GFDT) descriptor	0x0003	Provides one or more CodePoints describing the association of a specific object and object delivery properties.	X	

6.7.4.4.2 Detailed specifications of descriptors

6.7.4.4.2.1 MPU Timestamp descriptor

The syntax and semantics of the MPU timestamp descriptor are specified in ISO/IEC 23008-1.

6.7.4.4.2.2 Dependency descriptor

The syntax and semantics of the dependency descriptor are specified in ISO/IEC 23008-1.

6.7.4.5 Packet identification

Certain fixed values are used to identify MMTP packets so that a receiver terminal can easily recognize the information carried by the MMTP packet. These values are listed in Table 10.

Table 10 — Packet ID assignments

Value	Description
0x0000	PA message
0x0001	reserved for ARIB (CA message)
0x0002	AL-FEC message
0x0003 – 0x00FF	reserved for future use
0x0100 – 0x7FFF	reserved for private use
0x8000	reserved for ARIB (M2section message carrying MH-EIT)
0x8001	reserved for ARIB (M2section message carrying MH-AIT)
0x8002	reserved for ARIB (M2section message carrying MH-BIT)
0x8003	reserved for ARIB (M2section message carrying MH-SDTT)
0x8004	reserved for ARIB (M2section message carrying MH-SDT)
0x8005	reserved for ARIB (M2short section message carrying MH-TOT)
0x8006	reserved for ARIB (M2section message carrying MH-CDT)
0x8007	reserved for ARIB (Data transmission message)
0x8008 – 0xEFFF	reserved for private use
0xF000 – 0xFFFF	reserved for private use

6.7.4.6 Signalling information in ARIB systems

Additional signalling information is specified by Association of Radio Industries and Businesses (ARIB). Table 11, Table 12 and Table 13 list the messages, tables, and descriptors, respectively.

MPEG-2 TS based conventional broadcasting systems have used numerous tables and descriptors. Some of them are reused in MMT-based broadcasting systems. This signalling information has “MH-” at the beginning of its name.

Table 11 — List of messages additionally specified in ARIB systems

Message name	Message_id assignment	Description
Conditional access (CA) message	0x8001	Conveys information on conditional access.
M2short section message	0x8002	Conveys MPEG-2 short section-format table.
Data transmission message	0x8003	Conveys one or more tables related to data transmission.

Table 12 — List of Tables additionally specified in ARIB systems

Table name	Table_id assignment	Description
Layout configuration table	0x81	Assigns layout information for displaying assets.
Entitlement control message	0x82 – 0x83	Conveys common information consisting TV programme information (related to TV programmes, descrambling keys, etc.) and control information (instructions on compulsory on/off of the decoder's descramble function).
Entitlement management message	0x84 – 0x85	Conveys individual information including contract information for each subscriber and work keys to decrypt common information.
MH-conditional access table	0x86	Conveys one or more descriptors related to conditional access.
Download control message	0x87 – 0x88	Conveys information related to descrambling keys to descramble channel encryption for download.
Download management message	0x89 – 0x8A	Conveys information related to download keys to decrypt DCM.
MH-event information table	0x8B – 0x9B	Conveys information related to TV programmes such as programme name, broadcast date and time, and explanations of them.
MH-application information table	0x9C	Conveys dynamic control information and additional information for executing applications.
MH-broadcaster information table	0x9D	Presents information on broadcasters in the network.
MH-software download trigger table	0x9E	Conveys announcement information about downloads, such as the service id, schedule information, and target receiver terminals.
MH-service description table	0x9F – 0xA0	Conveys information related to the programme channel, such as the channel name and broadcaster's name.
MH-time offset table	0xA1	Indicates the current date and time and provides the time difference between the current time and indicating time for humans.
MH-common data table	0xA2	Conveys data that are commonly required for receiver terminals and stored in non-volatile memory, such as company logos.
Data directory management table	0xA3	Provides directory information on files constituting applications.
Data asset management table	0xA4	Provides the MPU configuration of the asset and the version of the MPU.
Data content configuration table	0xA5	Provides configuration information on files that are used as data content.
Event message table	0xA6	Provides information related to event messages.

Table 13 — List of Descriptors additionally specified in ARIB systems

Descriptor name	Descriptor_ tag value assignment	Description
Asset group descriptor	0x8000	Provides the group and priority within a group of assets.
Event package descriptor	0x8001	Provides a description on the relationship of events and the MMT packages.
Background colour descriptor	0x8002	Provides background colour information on the layout configuration.
MPU presentation region descriptor	0x8003	Provides information on the position of displaying the MPU.
Access control descriptor	0x8004	Identifies the conditional access method.
Scramble descriptor	0x8005	Identifies the scrambling sub-system.
Message authentication method descriptor	0x8006	Identifies the message authentication method.
MH-emergency information descriptor	0x8007	Provides information on and functions for emergency alarm signals.
MH-MPEG-4 audio descriptor	0x8008	Provides basic information for identifying the coding parameters of MPEG-4 audio streams.
MH-MPEG-4 audio extension descriptor	0x8009	Provides additional information for identifying the profile and level of MPEG-4 audio streams.
MH-HEVC video descriptor	0x800A	Provides information for identifying the coding parameters of HEVC video streams.
MH-linkage descriptor	0x800B	Provides a description of the relationship with other programme channels.
MH-event group descriptor	0x800C	Provides a description of grouping information for multiple events.
MH-service list descriptor	0x800D	Provides a description of programme channels and a list of their types.
MH-short event descriptor	0x800E	Provides the name and a brief explanation of the TV programme.
MH-extended event descriptor	0x800F	Provides detailed information about the TV programme.
Video component descriptor	0x8010	Provides parameters and explanations of video signals.
MH-stream identifier descriptor	0x8011	Identifies individual programme element signals of the TV programme.
MH-content descriptor	0x8012	Provides a description of the TV programme's genre.
MH-parental rating descriptor	0x8013	Provides information on the permitted minimum audience age.
MH-audio component descriptor	0x8014	Provides parameters and explanations of audio signals.
MH-target region descriptor	0x8015	Provides target region information.
MH-series descriptor	0x8016	Provides series information for multiple events.
MH-SI parameter descriptor	0x8017	Provides transmission parameters of signalling information, e.g., the retransmission period.
MH-broadcaster name descriptor	0x8018	Provides the broadcaster's name.
MH-service descriptor	0x8019	Provides descriptions of the programme channel and its company's name.
IP data flow descriptor	0x801A	Provides information on IP data flows in the broadcasting services.

Table 13 *(continued)*

Descriptor name	Descriptor_ tag value assignment	Description
MH-CA start-up descriptor	0x801B	Provides information on the start-up of CA programs having conditional access functions.
MH-type descriptor	0x801C	Provides type of files in data transmission.
MH-info descriptor	0x801D	Provides information related to MPU or item.
MH-expire descriptor	0x801E	Provides expiry information.
MH-compression type descriptor	0x801F	Provides the compression type and bytes of an item before compression.
MH-data component descriptor	0x8020	Identifies the coding scheme of data.
UTC-NPT reference descriptor	0x8021	Provides the relationship between NPT and UTC.
Event message descriptor	0x8022	Provides general information related to event messages.
MH-local time offset descriptor	0x8023	Provides the current local time and indicates whether daylight-savings time is observed.
MH-component group descriptor	0x8024	Provides a description of grouping information for multiple components.
MH-logo transmission descriptor	0x8025	Provides characters consisting of simple logos and references to CDT-format logos.
MPU extended timestamp descriptor	0x8026	Provides a decoding timestamp for access units in the MPU.
MPU download content descriptor	0x8027	Provides property information on the download content delivered in the MPU.
MH-network download content descriptor	0x8028	Provides property information on the download content delivered in broadband networks.
MH-application descriptor	0x8029	Provides a description of an application.
MH-transport protocol descriptor	0x802A	Provides transmission protocol and location information on applications that depend on transmission protocols.
MH-simple application location descriptor	0x802B	Provides detailed location information on applications.
MH-application permission descriptor	0x802C	Provides descriptions of the application boundary and permission information.
MH-autostart priority descriptor	0x802D	Provides priority information for launch of applications.
MH-cache control info descriptor	0x802E	Provides cache control information for caching resources constituting applications.
MH-randomized latency descriptor	0x802F	Provides latency information for application control.
Linked PU descriptor	0x8030	Provides information on linked presentation units.
Locked cache descriptor	0x8031	Provides file information that is cached and locked.
Unlocked cache descriptor	0x8032	Provides file information that is un-cached and unlocked.

6.7.5 Start-up procedure of broadcasting service

Figure 55 shows the start-up procedure of a receiver terminal from the moment a user presses a channel change button to the moment the new TV programme begins to be shown on screen. Pressing the channel change button corresponds to identifying the service_id of the desired TV programme.

The first procedure is initiated in the IP multiplexing layer. In the case of the TLV multiplexing scheme, the receiver terminal parses the address map table (AMT) to associate the service_id with the IP data flow. Then, it parses the TLV-network information table (NIT) to acquire the physical channel information, such as the channel frequency carrying the IP data flow. On the basis of the acquired information, it tunes in to the broadcasting channel and receives the desired IP data flow.

After receiving the IP data flow, the second procedure in the MMT layer is initiated. The received IP packets carry the MMTP packets. To retrieve the PA message, the receiver terminal seeks MMTP packets whose packet_id=0. It parses the received PA message and gets the MP table in the PA message.

In MMT-based broadcasting systems, multiple services might be multiplexed into one IP data flow, as shown in Figure 51. Therefore, the receiver terminal checks whether the package_id of the acquired MP Table is equal to the desired service_id or not. If the package_id of the MP table is not equal to the desired service_id, the receiver terminal acquires the package list table from the PA message. Then, from the package list table, it identifies the packet_id of the MMTP packets carrying the MP table of the desired service.

From the MP table, the receiver terminal identifies the IP data flow and packet_id of MMTP packets carrying the required MPUs in the desired TV programme. It also identifies the presentation time of the MPU by referring to the MPU timestamp descriptor included in the MP table.

Then, the receiver terminal receives the identified MMTP packets carrying media components in the form of MFUs. The MFUs are decoded and rendered at the designated time. The user watches the desired TV programme at this time.

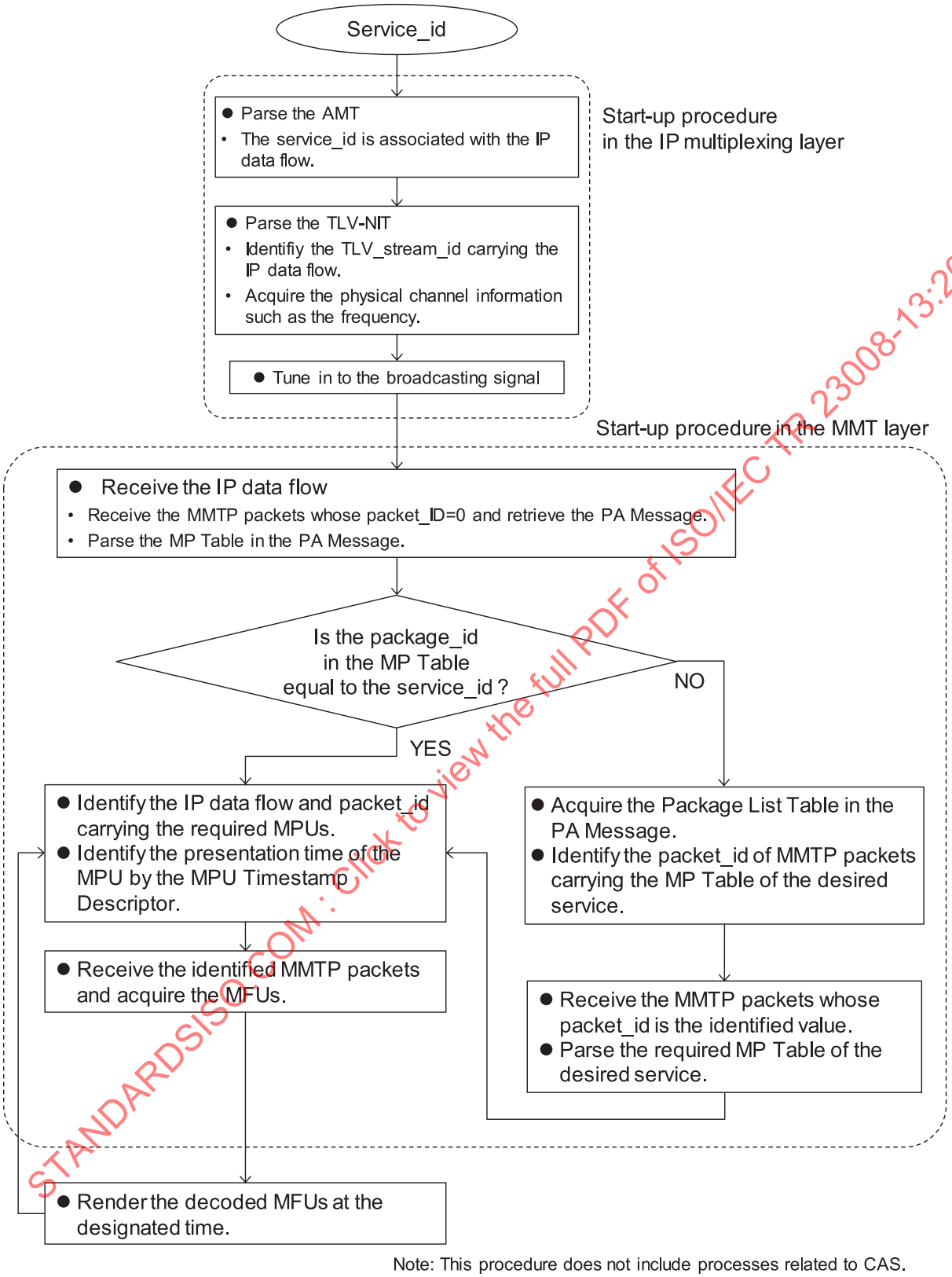


Figure 55 — Start-up procedure of broadcasting service

6.7.6 Actual packet structure

6.7.6.1 Structure of MMTP packet

Attachment 1 of this document (see subclause 6.7.6.4) is the actual packet data of MMTP/UDP/IPv6. The data is captured packets generated by a pump that outputs a sequence of MMTP packets.

Figure 56 shows an example of MMTP packet structure carrying a PA message that is one packet in the actual packet data as attachment 1.

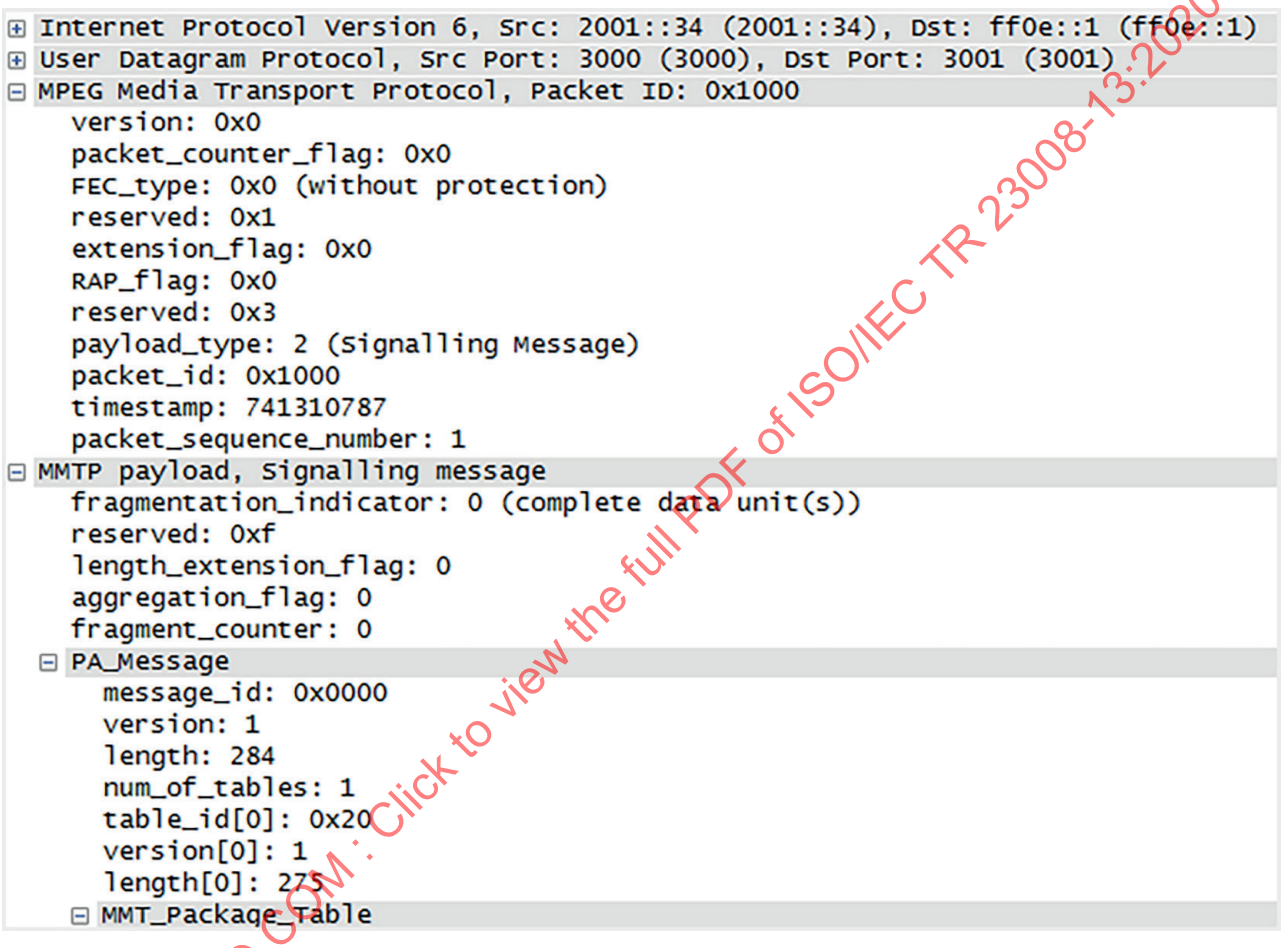


Figure 56 — MMTP packet structure

In the figure, it is shown that an MMTP packet has the following fields: version, *packet_counter_flag, FEC_type, extension_flag, RAP_flag, payload_type, packet_id, timestamp, and packet_sequence_number.*

Version 0 of an MMTP packet is used in ARIB STD-B60. A *packet_counter* field is not used, since there is no reordering of transmitted packets over broadcasting channels. At the beginning of the MMTP payload, there are *fragmentation_indicator*, *aggregation_flag*, and *fragment_counter* fields. Just after these fields, a PA message is located.

6.7.6.2 Structure of MP table

Figure 57 shows one example of an MP table that is in the PA message. This MP table is for a package that has two assets. Figure 57 shows information on the video asset.

In this MP table, an IPv6 multicast address is identified as the destination address of the *MMT_general_location_info* structure. It is shown that this MP table has an MPU timestamp descriptor and MPU extended timestamp descriptor for this asset.

```
MMT_Package_Table
  table_id: 0x20
  version: 1
  length: 275
  reserved: 0x3f
  MPT_mode: 2 (independent)
  MMT_package_id_length: 2
  MMT_package_id: 0x0100
  MPT_descriptors_length: 0
  number_of_assets: 2
  Asset
    identifier_type: 0 (AssetID)
    asset_id_scheme: 0x00000000
    asset_id_length: 1
    asset_id: 0x01
    asset_type: "hev1"
    reserved: 0x7f
    asset_clock_relation_flag: 0 (NTP)
    location_count: 1
    MMT_general_location_info
      location_type: 2
      src_addr: 2001:0000:0000:0000:0000:0000:0000:0034
      dst_addr: ff0e:0000:0000:0000:0000:0000:0000:0001
      dst_port: 3001
      packet_id: 0x0100
    asset_descriptors_length: 123
    MPU_Timestamp_Descriptor
      descriptor_tag: 0x0001
      descriptor_length: 12
      mpu_sequence_number: 1
      mpu_presentation_time: 0xda192c2f.813953de (2015-12-14 20:53:19.504)
    MPU_Extended_Timestamp_Descriptor
      descriptor_tag: 0x8026
      descriptor_length: 63
      reserved: 0x1f
      pts_offset_type: 0x0
      timescale_flag: 0x1
      timescale: 90000
      mpu_sequence_number: 1
      mpu_presentation_time_leap_indicator: 0
      reserved: 0
      mpu_decoding_time_offset: 4500
      num_of_au: 25
      dts_pts_offset: 4500
      dts_pts_offset: 15000
      dts_pts_offset: 7500
      dts_pts_offset: 3000
      dts_pts_offset: 0
      dts_pts_offset: 1500
      dts_pts_offset: 4500
      dts_pts_offset: 1500
      dts_pts_offset: 3000
```

Figure 57 — MP table structure

6.7.6.3 MMT/TLV packet structure

When an MMTP packet is carried in a TLV packet, one MMTP packet is encapsulated into one IP packet or one header compressed packet, and then that packet is encapsulated into one TLV packet.

The TLV packet is transmitted over Ethernet with the structure shown in Figure 58. This figure shows a MAC frame and IPv4 packet, which carry a TLV packet. The actual packet data of this format is provided as attachment 2. This structure is for carrying MMTP packets in TLV packets over the interface like ethernet.

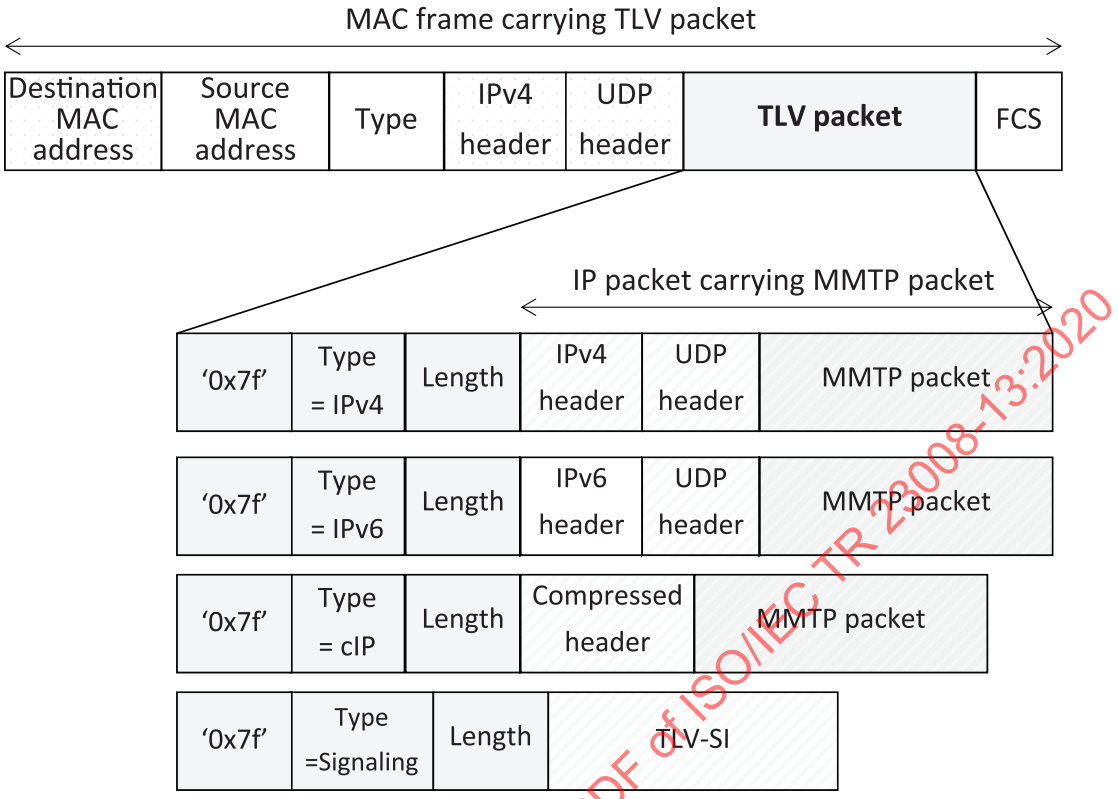


Figure 58 — TLV packet transmitted over ethernet

6.7.6.4 Captured packet

The electronic attachments are available at: https://standards.iso.org/iso-iec/tr/23008/-13/ed-3/en.

Attachment 1 is the captured packet of MMTP/UDP/IPv6 packets.

Attachment 2 is the captured packet of MMTP/UDP/IPv6 packets carried in TLV packets. In this case, TLV packets are carried by IPv4 packets and MAC frames as described in subclause 6.7.6.3.

Attachment 3 is a lua script, which is a plugin for Wireshark[3]. When this script is used with Wireshark, actual packets attached to this text can be analysed as shown in Figure 56 and Figure 57.

Attachment 1 Attachment 2 Attachment 3

6.8 MMT deployment in ATSC 3.0 systems

ATSC DoC. A/331[3] specifies the technical mechanisms and procedures pertaining to service signalling and IP-based delivery of a variety of ATSC 3.0 services and contents to ATSC 3.0-capable receivers over broadcast, broadband and hybrid broadcast/broadband networks. It specifies the usage and extensions of ISO/IEC 23008-1 for broadcasting services. The extensions include additional signalling information.

3) Wireshark, a registered trademark of the Wireshark Foundation, is an example of a suitable product available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of this product.

An *mmt_atsc3_message()* is defined by ATSC to carry information specific to ATSC 3.0 services delivery by MMT. The *message_id* for the *mmt_atsc3_message()* is set to "0x8100", which is in the "user private" range per ISO/IEC 23008-1:2017, 10.7. The syntax and semantic of this message are shown in Table 1.

Table 14 — Bitstream syntax for mmt_atsc3_message()

Syntax	No. of Bits	Format
mmt_atsc3_message() {		
message_id	16	uimsbf
version	8	uimsbf
length	32	uimsbf
message payload {		
service_id	16	uimsbf
atsc3_message_content_type	16	uimsbf
atsc3_message_content_version	8	uimsbf
atsc3_message_content_compression	8	uimsbf
URI_length	8	uimsbf
for (i=0;i< URI_length;i++) {		
URI_byte	8	uimsbf
}		
atsc3_message_content_length	32	uimsbf
for (i=0;i<atsc3_message_content_length;i++) {		
atsc3_message_content_byte	8	uimsbf
}		
for (i=0;i<length-11-URI_length-atsc3_message_content_length) {		
reserved	8	uimsbf
}		
}		
}		

message_id – A 16-bit unsigned integer field that should uniquely identify the mmt_atsc3_message(). The value of this field should be 0x8100.

version – An 8-bit unsigned integer field that should be incremented by 1 any time there is a change in the information carried in this message. When the version field reaches its maximum value of 255, its value should wrap around to 0.

length – A 32-bit unsigned integer field that should provide the length of mmt_atsc3_message() in bytes, counting from the beginning of the next field to the last byte of the mmt_atsc3_message().

service_id – A 16-bit unsigned integer field that should associate the message payload with the service identified in the serviceId attribute given in the SLT defined in [3].

atsc3_message_content_type – A 16-bit unsigned integer field that should uniquely identify the type of message content in the mmt_atsc3_message() payload, coded per ATSC A/331:2019, Table 7.8[3].

atsc3_message_content_version – An 8-bit unsigned integer field that should be incremented by 1 any time there is a change in the mmt_atsc3_message content identified by a service_id, and atsc3_message_content_type pair and URI if present. When the atsc3_message_content_version field reaches its maximum value, its value should wrap around to 0.

atsc3_message_content_compression – An 8-bit unsigned integer field coded per ATSC A/331:2019, Table 7.9[3] that should identify the type of compression applied to the data in atsc3_message_content_byte.

URI_length – An 8-bit unsigned integer field that should provide the length of the URI uniquely identifying the message payload across Services. When the URI is not present, the value of this field should be set to 0. When this mmt_atsc3_message() carries an MPD (i.e. atsc3_message_content_type = 0x0002), the URI should be present.

URI_byte – An 8-bit unsigned integer field that should contain a UTF-8 character of the URI associated with the content carried by this message excluding the terminating null character, as per RFC 3986. This field, when present, should be used to identify delivered message payloads. The URI can be used by system tables to reference tables made available by delivered message payloads.

atsc3_message_content_length – A 32-bit unsigned integer field that should provide the length of the content carried by this message.

atsc3_message_content_byte – An 8-bit unsigned integer field that should contain a byte of the content carried by this message.

6.9 Implementation of MMT based on D-TMB in China

6.9.1 Background

The D-TMB Standard, GB 20600-2006 was released in August 2006. Both fixed and mobile terminals can receive media content through single-frequency or multi-frequency network via provided transmission techniques for digital televisions. D-TMB was announced as the mandatory terrestrial broadcasting standard in China on August 1, 2007.

6.9.2 MMT over legacy DTMB infrastructure

As the terrestrial broadcasting standard in China, D-TMB possesses solid industrial implementation and achieves national-wide signal communications. Meanwhile, considering the customized advantages of MMT, it is able to provide personalized and high-quality services based on MMT protocol over legacy D-TMB infrastructure.

The D-TMB transport media content is based on MPEG-2 (TS) standard, while the MMT packets are transmitted over IP. To enable MMT transmission based on D-TMB, a protocol conversion mechanism between TS packets and MMT packets is proposed. As shown in Figure 59, the media content encapsulated in MMT protocol is converted into TS packet through the protocol conversion and followed by TS encapsulation.

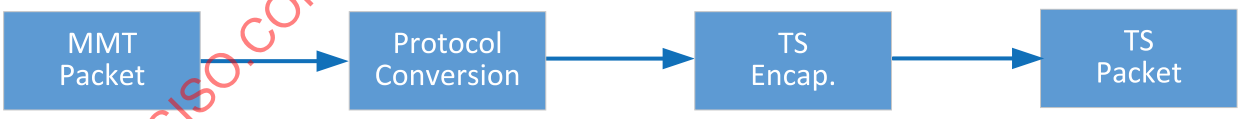


Figure 59 — Protocol conversion

6.9.3 Use cases

6.9.3.1 Synchronized and personalized presentation of multi-view

Based on the techniques above, synchronized display of main view and associated views from different networks on one or more terminals has been implemented. As shown in Figure 60, the main view of 4K, which is transport through the broadcast, and the auxiliary view of 2K, which is transport through the broadband, were played on the TV with time synchronization. The MMT packets are converted into MPEG2-TS packets in real time and transport to the D-TMB receiver by the D-TMB transmitter. Then, the TS packets, which are received by the D-TMB receiver, are converted to MMT packets. Then the MMT gateway can recognize the MMT packets.

It is realized in less than 200 milliseconds of time offset between different views. In this way, the broadcast network and the internet are efficiently integrated in the MMT system, which enables the complementarity of different networks and more flexible multimedia services.

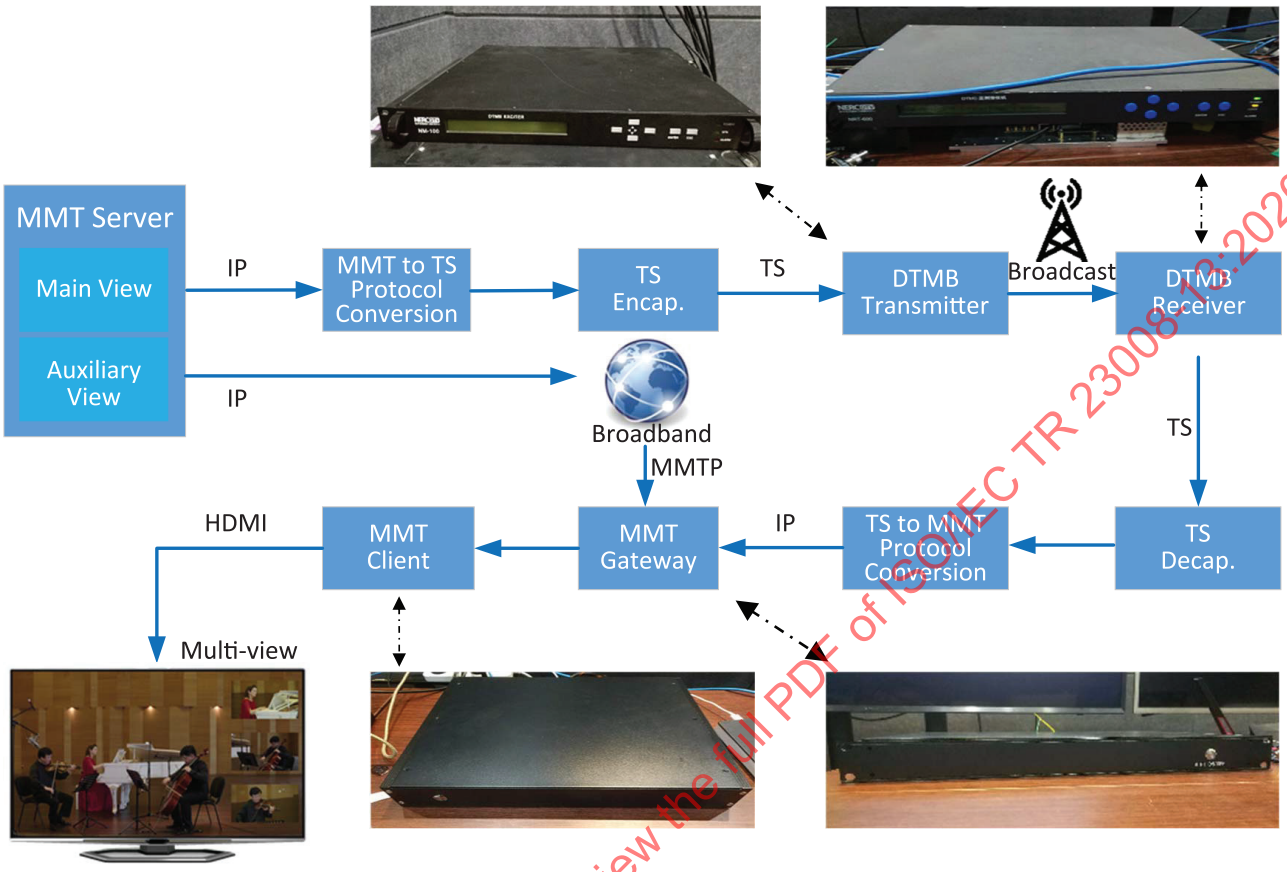


Figure 60 — Synchronized and personalized presentation of multi-view

6.9.3.2 Seamless switching between multiple networks

As shown in Figure 61, multiple networks work together to ensure the robustness of multimedia services. When the network works well, the client receives data from the broadcast network. While the client detects that the signal from the broadcast network is unstable or poor, it will automatically send a request to the server, this request tells the server to send multimedia data through broadband. Once the server gets the request, it will send multimedia data to the client through the broadband network. Since the MMT client could process the data received from multiple networks, it automatically disconnects from the broadcast network after receiving data from the broadband network. The MMT client can also adjust the bit rate adaptively to adapt to network bandwidth. The MMT system could switch between multiple networks adaptively. Furthermore, it can ensure the continuity of media services while network switching.

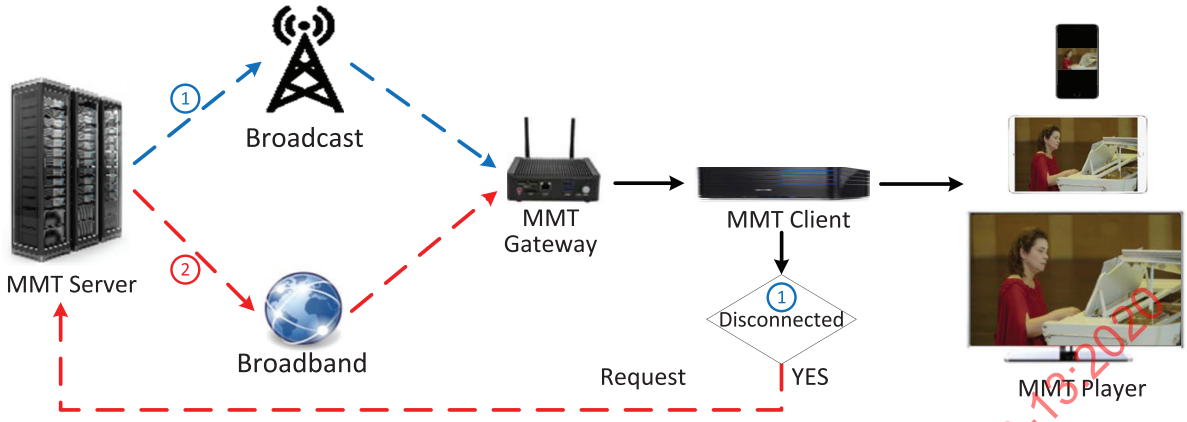


Figure 61 — Seamless switching between multiple networks

6.10 Conversion of MMTP stream to MPEG-2 TS

6.10.1 Overview of conversion operation

MMTP packets carry various data including metadata and signalling messages as depicted in Figure 62. Payload data of MMTP packets carrying metadata are processed to generate appropriate MPEG-2 section data or the values of the fields in the MPEG-2 TS packets or the PES packets. Payload data of MMTP packets carrying MFU data, i.e. the MMTP packets with the value of the FT field is '2', are converted to PES packets.

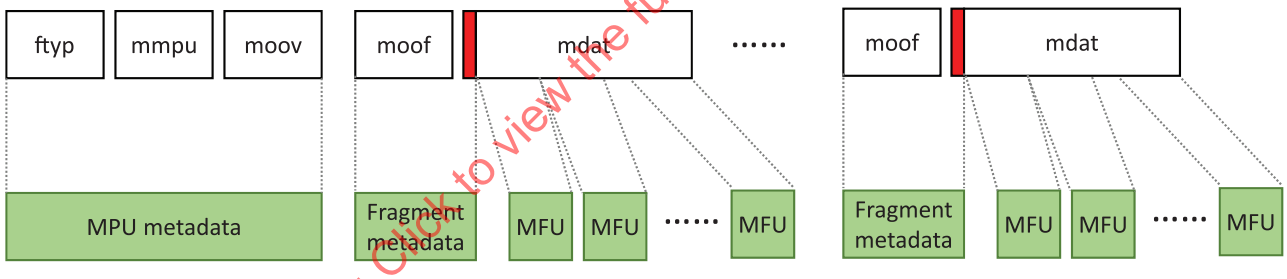


Figure 62 — MMTP packet generation

6.10.2 Restrictions to MMTP packets

MMTP allows highly flexible operation. There are many fields of MMTP packets with very wide range of values are allowed and many features configurable to various ways. As they can be beyond operation range of MPEG-2 TS, there are some restrictions to be applied to MMTP streams to ensure efficient conversion into MPEG-2 TS.

— Media track data should include codec initialization information. For example, MPUs carrying AVC video bitstream should carry all sequence and picture parameter sets (SPS and PPS) necessary for decoding the AVC video stream within that AVC video stream.

— The value of the 'packet_id' field of MMTP packets should be one between 0x0010 and 0x1FFE.

6.10.3 Calculation of PTS, DTS

In MMTP stream presentation, time and decoding time are represented as wall clock time, i.e. UTC-based absolute time. As presentation time and decoding time in MPEG-2 TS are represented by using PCR as a clock reference, presentation time and decoding time of media data carried by MMTP needs to be converted to PCR-based value. For the conversion, UTC-based presentation time and decoding time

of media data carried by MMTP are calculated by combining the value specified in *MPU_timestamp_descriptor* which provides the presentation time of the earliest sample in the MPU in presentation order and the composition time relative to the earliest sample in the MPU in presentation order known by movie fragment box data. Then, the presentation time and decoding time in UTC are converted to presentation time and decoding time referencing PCR. Considering clock of the client processing MPEG-2 TS is locked to the clock of the sender, presentation time and decoding time can be calculated by using following formula:

$$P_{PCR} = T_{PCR} + \frac{P_{UTC} - T_{UTC}}{2^{32}} \times 90{,}000$$

$$\mathrm{D_{PCR}} = \mathrm{T_{PCR}} + \frac{\mathrm{D_{UTC}} - \mathrm{T_{UTC}}}{2^{32}} \times 90{,}000$$

where

T_{PCR} is the value of the current PCR sample in 90 KHz;

T_{UTC} is the value of the current time in UTC;

P_{PCR} is the value of the presentation time of the media data in 90 KHz referencing PCR;

P_{UTC} is the value of the presentation time in UTC;

D_{PCR} is the value of the decoding time of the media data in 90 KHz referencing PCR;

D_{UTC} is the value of the decoding time in UTC.

6.10.4 Restriction related to MPEG-2 T-STD

As MPEG-2 TS should comply with the rules regarding MPEG-2 T-STD, the MMTP stream should be constructed in a way that the MPEG-2 TS converted from MMTP to comply to MPEG-2 TS rules without adding or removing data by using HRBM of MMT. As HRBM allows precise control of distance between MMTP packets, in other words when the media data of each MMTP packets available to the next entity connected to HRBM buffers, MMTP stream can be constructed to meet such a requirement by deciding timestamp of each MMTP packets appropriately by using HRBM. The relationship between HRBM and MPEG-2 T-STD can be conceptually represented as the Figure 63, where it is assumed that there is no processing delay added by MMTP to MPEG-2 TS converter. It reads reconstructed media data from MMTP decapsulation buffer with fixed rate R_C and delivers MPEG-2 TS packets to TB_n with fixed rate R_D, where R_D is the rate increased from R_C by fixed amount due to MPEG-2 TS packet header overhead introduced by the conversion operation. As TB_n should be empty in each second, at each second there should be no more media data in the MMTP decapsulation buffer available for the conversion operation before the data from the next MMTP packet is delivered to the MMTP decapsulation buffer.

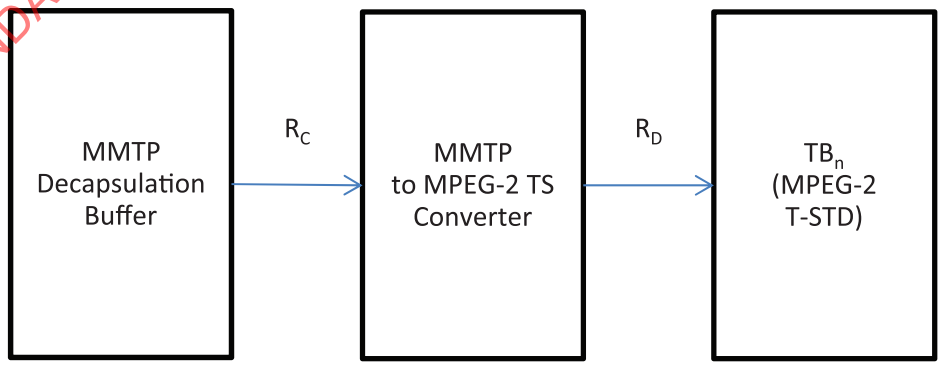


Figure 63 — Relationship between HRBM and MPEG2 T-STD

6.10.5 Packet field conversion rule

Tables 15, 16 and 17 provide basic rules to determine the value of the fields of MPEG-2 TS packet and PES packets. The value of the fields not specified in this subclause cannot be directly determined from the values of the fields of MMTP packets or signalling information but to be determined according to semantics defined in ISO/IEC 13818-1.

Table 15 — Rule for TS packet layer

MPEG-2 TS packet fields in TS packet layer	Rule for conversion from MMTP packets
payload_unit_start_indicator	The value of this field is set to '1' if the TS packet is the first packet converted from the MMTP packet with the value of the f_i field is either '00' or '01'.
transport_priority	The value of this field is set to '1' if the TS packet is converted from the MMTP packets with the priority field is smaller than a certain threshold value set by conversion system. If the threshold value is not provided by conversion system, then the TS packet generated from the MMTP packet with the value of the priority field is '0' is set to '1'.
PID	The value of this field is set to the same value of the packet_id field of the MMTP packet used to generate this TS packet.

Table 16 — Rule for adaptation field

MPEG-2 TS packet fields in adaptation field	Rules for conversion from MMTP packets
random_access_indicator	The value of this field is set to '1' if the MPEG-2 TS packet is the first packet generated from a MMTP packet with the value of the R field is '1'
elementary_stream_priority_indicator	The value of this field is set to '1' if the TS packet is converted from the MMTP packets with the priority field is smaller than a certain threshold value set by conversion system. If the threshold value is not provided by conversion system, then the TS packet generated from the MMTP packet with the value of the priority field is '0' is set to '1'.

Table 17 — Rule for PES packet

MPEG-2 TS packet field in PES packet	Rules for conversion from MMTP packets
stream_id	The value of this field is determined by the value of the asset_type field of MP table
PES_priority	The value of this field is set to '1' if the TS packet is converted from the MMTP packets with the priority field is smaller than a certain threshold value set by conversion system. If the threshold value is not provided by conversion system, then the TS packet generated from the MMTP packet with the value of the priority field is '0' is set to '1'.
data_alignment_indicator	The value of this field is set to '1'
PTS	The value of this field is calculated from composition time of media data. The value of P_{PCR} in subclause 6.10.3 is used.
DTS	The value of this field is calculated from decoding time of media data. The value of D_{PCR} in subclause 6.10.3 is used.

6.10.6 PSI Conversion rule

6.10.6.1 Comparison of content model

Both MPEG-2 systems and MMT have defined their own content models to construct a presentation with multiple media components. As shown in Figure 64, MPEG-2 system defines a program as a conceptual content model. A program is defined as a collection of program elements which share a single time base and are intended for synchronized presentation.

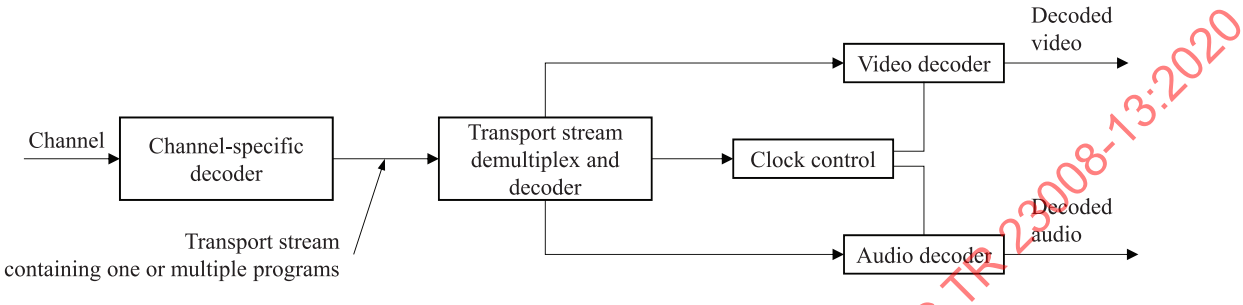


Figure 64 — Example of MPEG-2 TS demultiplexing and decoding

The content model of MMT is more complicated than that of MPEG-2 TS. An MMT package is defined as a collection of one or more MMT assets and optionally presentation information and asset delivery characteristics. As each asset will be decoded by a single independent decoder and all assets in a package are sharing same time base, UTC, and are intended for synchronized presentation, a package of MMT can be considered equivalent to a program of the MPEG-2 system and an asset of MMT can be considered equivalent to a program element of the MPEG-2 system.

6.10.6.2 Handling of program specific information

A package of MMT should be converted to a program of the MPEG-2 system and an asset of MMT should be converted to a program element of the MPEG-2 system to generate program specific information. Table 18 shows the rules to generate tables of MPEG-2 TS from MMTP stream.

Table 18 — Rule to generate tables of MPEG-2 TS from MMTP

Structure Name	Reserved PID #	Description	Conversion rule
Program association table	0x00	Associates program number and program map table PID	As MMT does not define a method to signal carriage of multiple packages in a single MMTP stream, PAT cannot be directly generated from information in MMTP stream. PID of PMT and structure of PAT should be predefined.
Program map table	Assigned in the PAT	Specifies PID values for components of one or more programs	PMT should be generated from MP table. — The value of the *packet_id* field for each asset from *MMT_general_location_info* in MP table of MMTP stream should be used for the value of the *elementary_PID* field of PMT. — The value of the *asset_type* field from MP table should be used for the value of the *stream_type* field of PMT.
Network information table	Assigned in the PAT	Physical network parameters such as FDM frequencies, transponder numbers, etc.	NIT can be generated from MP table. The value of IP address or MPEG-2 transport stream ID for each asset from *MMT_general_location_info* in MP table of MMTP stream should be used as the values to generate NIT for MPEG-2 TS.

Table 18 *(continued)*

Structure Name	Reserved PID #	Description	Conversion rule
Conditional access table	0x01	Associates one or more (private) EMM streams each with a unique PID value	There is no relevant information in MMTP stream to generate CAT.
Transport stream description table	0x02	Associates one or more descriptors from Tables 2 to 45 to an entire transport stream	There is no relevant information in MMTP stream to generate TSDT.
IPMP control information table	0x03	Contains IPMP tool list, rights container, tool container defined in ISO/IEC 13818-11	There is no relevant information in MMTP stream to generate IPMP table.

6.11 MMT service provisioning at conventional broadcast environment

MMT service provisioning is also possible for MMT-capable devices (MMT receiving entities such as TV) which are connected to STB. Currently, many people are still watching their TV through STB. However, this STB may not be able to support system protocols (e.g. MMT) or IP-based media data reception in many cases.

In this environment, TV can utilize various metadata attached in content as a service trigger point for a more enriched service. A content provider or service provider who feeds the content on TV screens through STB is able to pop up new additional content or advertisements on the TV screen to improve user service satisfaction. This content can be provided through the broadband access connected to the TV. For example, additional content such as baseball player statistics or player video in another angle view can be presented on the TV screen while the user is watching a baseball game through STB on TV.

This metadata can be in any form that can be utilized by MMT-capable TV. The metadata can be included in the form of hidden watermarks in audio/video or in the form of data in audio bitstream, etc. The metadata for service trigger can bypass STB as embedded in audio-visual media or within a bitstream. Then, it can be extracted from automatic contents recognition (ACR) engines or A/V decoders at TV.

In some cases, these data can be URLs to fetch the data related to the new additional contents/service. In another extreme end, it may be media packet, service identifier or media data itself, etc.

The metadata can be generated and embedded by content providers such as a studio. It can also be newly added, modified, removed or replaced by a service provider in the media delivery chain such as MVPD network operators under a certain contract as depicted in Figure 65. In that case, MVPD can provide its own new service, personalized advertisement or any new additional service from 3rd party provider under some contract.

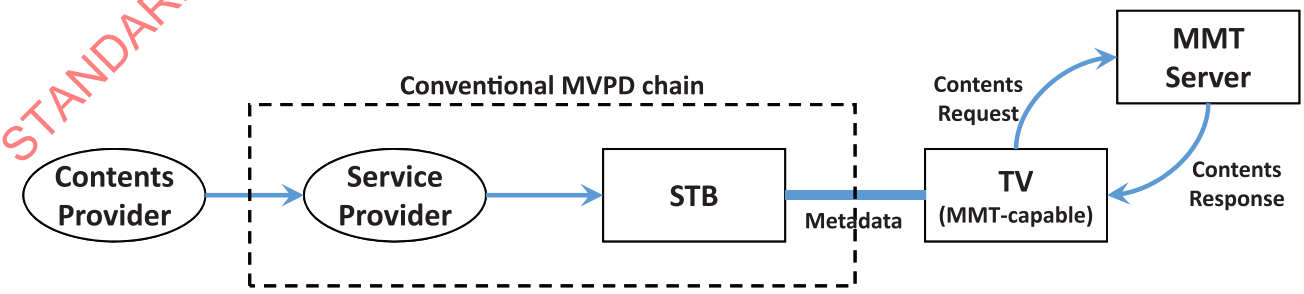


Figure 65 — MMT service in conventional broadcast environment

This metadata addition/removal/replacement operation can also be done in the STB to add or replace the service/content into another one. When the STB receives data from STB input interface, a transport processor recovers delivery units such as MPEG2-TS, IP or an MMTP packet from it. Then, a media

processor will extract actual audio/video data and each of them will be processed by an audio and video processor in STB. When metadata is delivered to TV through an STB output digital interface like HDMI, a metadata generator in STB can add/remove/replace metadata in audio or video bitstream by providing the information to audio/video processors in STB.

There may be cases where both the STB and TV are MMT-capable devices. If both of them extract metadata from the media and transact it at the same time, it could be undesirable because the same service can be triggered at both sides concurrently. In that case, STB and TV should detect their own and each other's capability whether they are supporting MMT or not. This can be done, for example, by device API, interaction between a client and a dedicated server that provides capability information or WiFi connection between STB and TV.

Additionally, an audio processor at STB should not decode any audio bitstream and bypass it through STB digital output interface such as HDMI. Then, metadata in audio bitstream is directly delivered to the TV.

Figure 66 shows the basic metadata processor architecture for a TV. The metadata processor can be an audio metadata processor or video metadata processor. If it is assumed to be an audio case, the audio metadata processor receives audio data including metadata from the audio processor inside the TV. Actual metadata is extracted from a metadata extractor and it is parsed and transformed at a metadata parser for suitable operation of the metadata processor.

The TV can decide to process and take an action for the extracted metadata based on the preference of user or whether it has the capability to process the MMT capability. If it has the MMT transaction capability, it can cope with fetching and presenting media as described in fetched media like ISO/IEC 23008-11.

If the parsed metadata includes information related to render/present the media, it is delivered to a media presentation processor. Then, the media presentation processor extracts and manipulates that information and hands it over to a media player. On the other hand, if it is the information related to media acquisition such as URLs, it is transferred to a media delivery processor. The media delivery processor fetches media from the media location information by using other interfaces and protocols such as broadband internet and MMT, etc., and hands it over to the decoder. The metadata may not include all the information to render/present the media to the TV and, in this case, the media delivery processor can query more information to do that through other interfaces and protocols.

As described above, metadata can be any type of data such as MMTP packet, contents identifier, media data itself, media locations, etc. If it is an MMT packet A/V processor, the output will be the MMTP packet that includies the MMT signalling message. Then, the metadata extractor will extract the MMT signalling message first from the extracted MMTP packet streams and it will be informed to the metadata parser.

If the metadata is a kind of custom contents identifier defined by CP or SP, the metadata processor will transact it based on a pre-defined policy of CP/SP by checking a service list and fetching media data and presentation-related data. It can be acquired by a media delivery processor and delivered to the decoder and media player.

If the metadata is a media source location to fetch media such as a URL, the metadata processor will query to those URL to acquire a PA signalling message as a root or CI page to initiate the MMT service. When a PA message is received, the MMT engine will parse it and extract all the signalling messages and assets based on it. When the CI page is received, it can render media as described in ISO/IEC 23008-11.

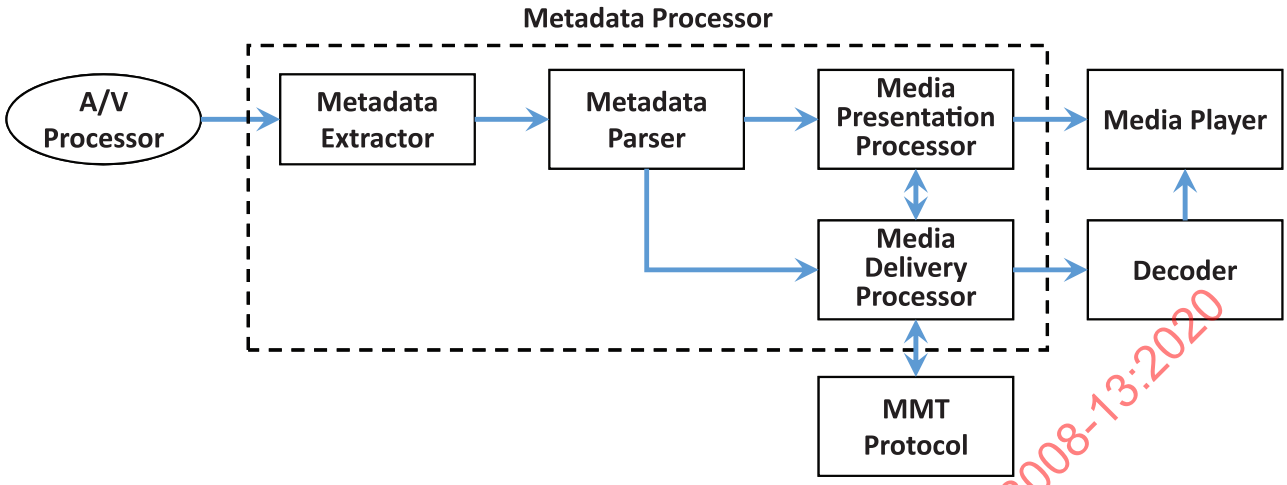


Figure 66 — Architecture of metadata processor and its interface at TV

6.12 Usage of multimedia configuration for interface switching management

Compared to an HTTP-based multimedia delivery system, an MMT-based multimedia delivery system can use the multimedia configuration (bitrate, packet loss, data bandwidth, propagation delay, buffer status, etc.) to support the QoS management during the switching between LTE unicast and WiFi to prevent the degradation of QoS.

In an MMT-based multimedia delivery system, an ADC message information can be used which defines QoS requirements and statistics of Asset for delivery, and their associated QoE quality information. This information can be used by the MMT-aware intermediate network entities for QoS-managed delivery of assets. And the mobile AP's network status can be estimated by using MMTP packet header information such as the packet counter and timestamp.

By using a mobile AP's network status and multimedia configuration, the user can accurately decide the network switch from Unicast to WiFi. If the mobile AP's network status satisfied the multimedia configuration requirement, then the user will be able to switch from LTE Unicast to WiFi. If the mobile AP's network status does not satisfy the multimedia configuration requirement, the user will continuously receive the mobile video streaming via the LTE unicast.

6.13 MMT signalling for multiple timed text assets

6.13.1 Multiple timed text assets within an MMT presentation

To provide rich experience to users there can be multiple assets displayed within a frame, not only the main video, audio and subtitle.

In a multiple timed text assets case, the assets can include various types of timed text data such as graphic effects which enhance the context of contents givev to the viewers, a program logo or more than one language of subtitle.

Figure 67 shows an example of multiple timed text data in one presentation in MMT. Area 1 can be used for the timed text asset of a program logo. The program logo can be a graphic which explains the current context of the main video. It usually changes after interstitial advertisements during a TV show. Area 2 is for the timed text asset of a subtitle. A subtitle is different to a closed caption, because a subtitle is refined in the production stage and a closed caption is a textual representation of the whole sound. Area 3 is for a director's intended timed text asset. The director's intended timed text can be an additional subtitle or graphic effect which is generated in the production stage. The director's intended timed text data is different to a conventional subtitle because it's not a textual representation of dialogue. It gives additional explanations or comments to attract more concentration from the viewers. As well as text, graphics such as an emoticon, arrow or short video clip can be utilized as director's intended timed

text. Area 4 is for the broadcaster's logo; usually the channel logo is displayed. Table 19 is a spatial assignment for multiple timed text assets.

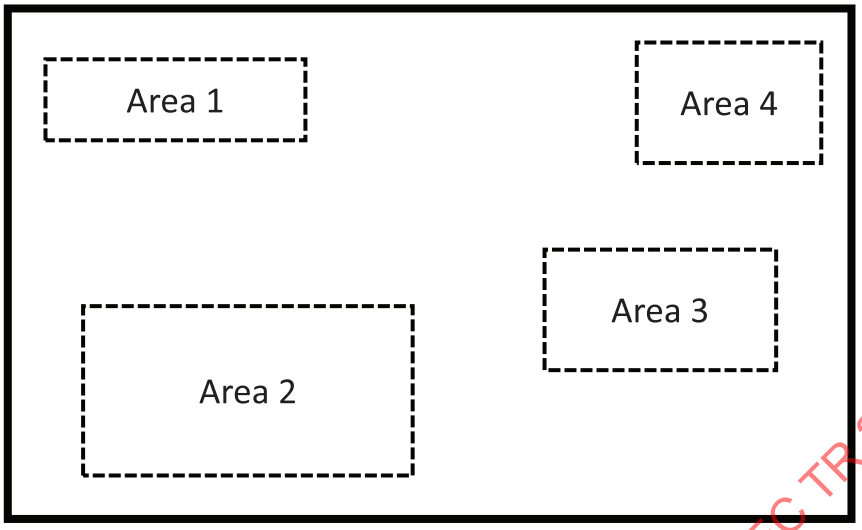


Figure 67 — Example of multiple timed text in an MMT presentation

Table 19 — Spatial assignment for multiple timed text assets

	Asset No.	Content	Language
Area 1	Asset 1	Program logo	English
Area 2	Asset 2	Subtitle	English
Area 3	Asset 3	Produced subtitle (Director intended timed text)	English
Area 4	Asset 4	Channel logo	N/A

6.13.2 Selective spatial assignment for multiple timed text assets

Using multiple timed text assets within one MMT presentation, the service provider could easily change the spatial assignment for different language users.

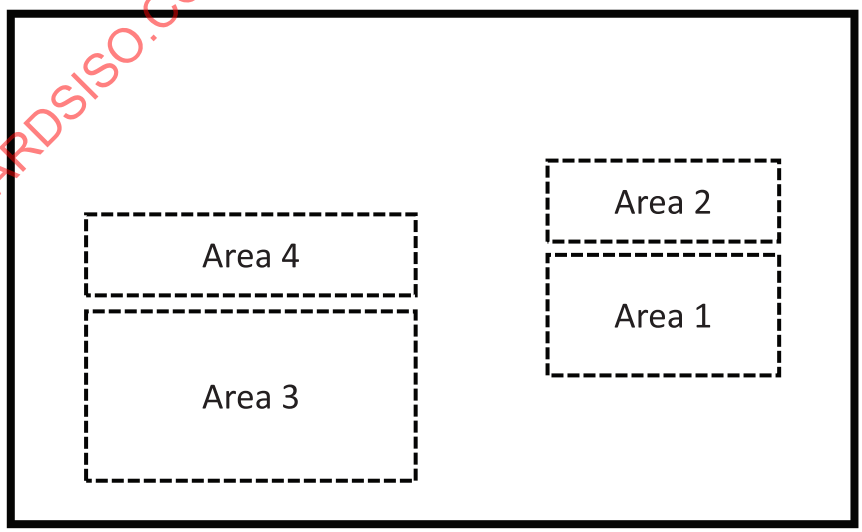


Figure 68 — Example of multiple timed text in an MMT presentation

Figure 68 shows an example of spatial assignment for selective spatial assignment. When an original content produced for English speaking viewers has one director's intended timed text asset 1 in Area 1, then the service provider can deliver the timed text asset 1 to the users who speak English. Table 20 shows the spatial assignment for the original content.

Table 20 — Spatial assignment for the original content

	Asset No.	Media type	Content	Language
Area 1	Asset 1	Timed text	Produced subtitle (Director intended timed text)	English

However, when the content is provided in many countries, the service provider can select and prepare multiple options. For example, if the service provider would like to fully localize the content for Korean viewers, then put the Korean version of the director's intended timed text in Area 1 instead of the English version. And, of course, the Korean subtitle on the main audio can be located in Area 3. Table 21 shows the spatial assignment of this example.

Table 21 — Spatial assignment for full localization

	Asset No.	Media type	Content	Language
Area 1	Asset 2	Timed text	Produced subtitle (Director intended timed text)	Korean
Area 3	Asset 4	Timed text	Subtitle	Korean

Moreoverm if a service provider would like to localize the content partially, then the related assets can be assigned as per Table 22 and Table 23.

Table 22 — Spatial assignment for partial localization

	Asset No.	Media Type	Content	Language
Area 1	Asset 1	Timed text	Produced subtitle (Director intended timed text)	English
Area 2	Asset 2	Timed text	Produced subtitle (Director intended timed text)	Korean
Area 3	Asset 4	Timed text	Subtitle	Korean

Table 22 is an example of spatial assigment for partial localized content. The director's intended timed text data can be displayed in both English and Korean. The director's intended timed text data could be a graphic effect of internet slang, for instance, then it could be produced differently in the Korean version. For example, this timed text can be a graphic effect on *'LOL'* in English, but '□□□□□' in Korean.

In a similar manner, the subtitle can be provided both in English and Korean as shown in Table 23. In Table 23, the English subtitle can be prepared for foreigners separately in closed caption.

Table 23 — Spatial assignment for partial localization with multiple subtitles

	Asset No.	Media Type	Content	Language
Area 1	Asset 1	Timed text	Produced subtitle (Director intended timed text)	English
Area 2	Asset 2	Timed text	Produced subtitle (Director intended timed text)	Korean
Area 3	Asset 3	Timed text	Subtitle	English
Area 4	Asset 4	Timed text	Subtitle	Korean

6.13.3 Example of multiple timed text assets signalling in MMT

The multiple timed text assets can be signalled using *default_asset_flag* in an MP table, layout configuration table, and asset group descriptor.

The layout for the multiple timed text assets in an MMT presentation is defined by layout configuration table and carried in the PA message. The number of areas within a presentation and the size of each area are defined. Content can use multiple layouts, and the PA message can be updated when the layout needs to be change.

For efficient and rapid management on multiple timed text assets, an asset group descriptor can group the assets by the contained content and its language. For more than one timed text, assets have the same content but are different in language; a group descriptor can assign the same value of *group_identification* on all assets, but distinguish between them with a different value of *selection_level*.

In this case, all asset groups should have the same value of the *selection_level* to enable asset selection by language. Table 24 shows asset group descriptor usage for the examples described in subclause 6.13.2.

Table 24 — Example of field values of asset group descriptor for multiple timed text assets

Asset No.	Content	Language	group identification	selection level
Asset 1	Produced subtitle (Director intended timed text)	English	0	0
Asset 2	Produced subtitle (Director intended timed text)	Korean	0	1
Asset 3	Subtitle	English	1	0
Asset 4	Subtitle	Korean	1	1

To indicate multiple timed text assets to be displayed at the same media presentation time, a *default_asset_flag* in an MP table can be used. When the timed text asset selection has changed, the MP table should be updated. Only used timed text assets can have *default_asset_flag* value as '0'. Table 25 gives an example to compose a single frame at the same media presentation time using combination of *group_identification*, *selection_level*, and *default_asset_flag*.

Table 25 — Example of field values of asset group descriptor and default asset flag in MPT

Asset No.	Content	Language	group identification	selection level	default asset flag
Asset 1	Produced subtitle (Director intended timed text)	English	0	0	0
Asset 2	Produced subtitle (Director intended timed text)	Korean	0	1	0
Asset 3	Subtitle	English	1	0	1
Asset 4	Subtitle	Korean	1	1	0

Based on the field values of Table 25, the director's intended timed text would be displayed in both English and Korean version, and subtitle would be displayed only Korean.

6.13.4 Carriage of TTML based timed text in MMT

6.13.4.1 General

Timed text markup language (TTML) based timed text can be stored in an ISOBMFF track as defined in ISO/IEC 14496-30. ISO/IEC 23008-1 specifies the 'mpuf' branded ISOBMFF, MPU, for timed media contents. Therefore, to encapsulate the TTML based timed text in MPU, both ISO/IEC 14496-30 and ISO/IEC 23008-1 should be satisfied.

6.13.4.2 MPU encapsulation of TTML based timed text

In accordance with MMT, the timed media MPU can be encapsulated based on ISOBMFF. The example of MPU encapsulation of timed media data from MMT specification is depicted in Figure 69.

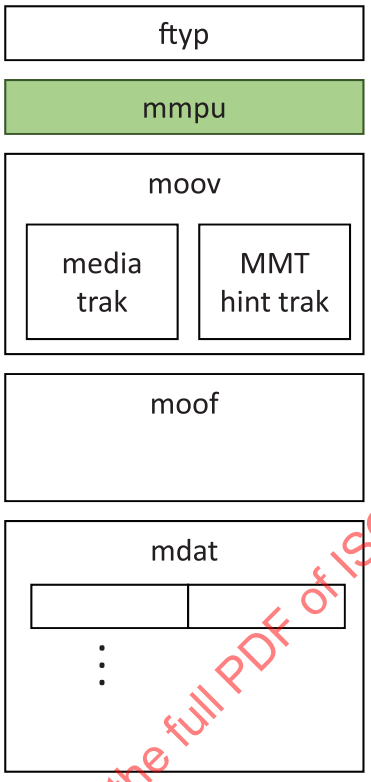


Figure 69 — Example of MPU encapsulation for timed media

In accordance with ISO/IEC 14496-30, the timed text XML document can be stored in a timed text ISOBMFF track as a timed text sample. If the timed text is image type, then the image resources are referenced by the XML document.

A timed text sample should contain a timed text XML document. Optionally, image resources can be stored in the same sample formed as sub-samples. The XML document should be the first sub-sample, and each image resources should be defined as subsequent sub-samples in the same timed text sample. The example of timed text sample structure using sub-samples from ISO/IEC 14496-30 is depicted in Figure 70.

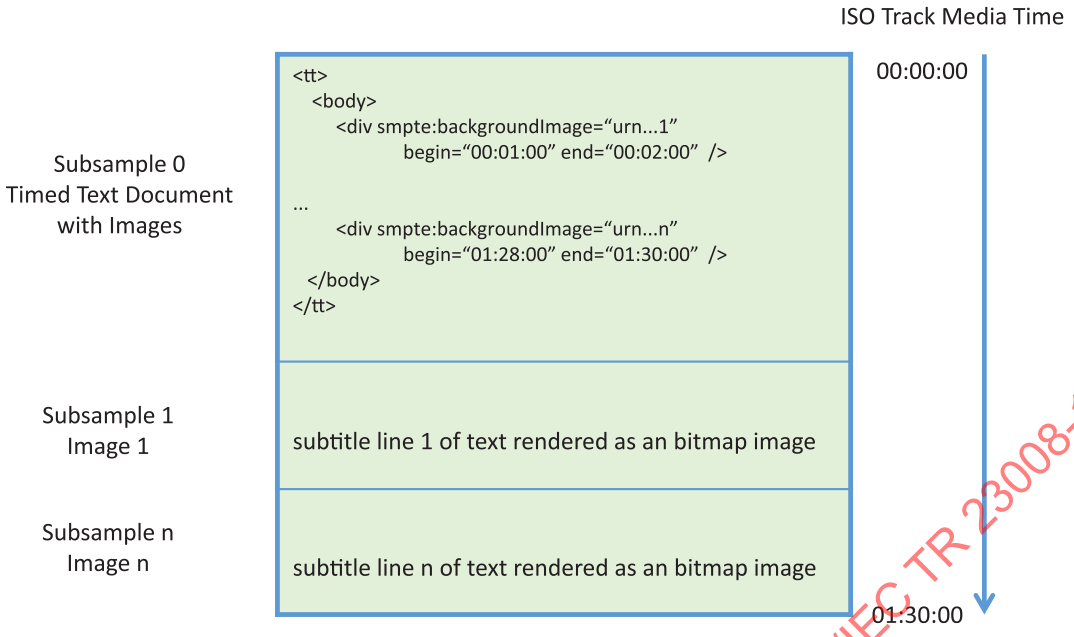


Figure 70 — Subtitle sample structure when using sub-samples

The mandatory boxes for sub-sample structure such as the Track Fragment Box ('traf') and Sub-Sample Information Box ('subs'), and their constraints are the same as defined in ISO/IEC 14496-30.

The Track Header Box should be used (e.g., the value of 'width' and 'height') to size the timed text track content with respect to the video, unless specified by an embedding environment. The video and timed text should be layered using the 'layer' value. The track width and height can be provided by the XML document as declared in 'tt' element.

The timing values of the timed text samples are provided by the XML document. For example, the 'begin' and 'end' attributes of the <body> element provide the internal timing value in terms of the track presentation timeline. Hence the decoding time and presentation time can be determined by the timing values provided by the XML document.

6.13.4.3 Metadata signalling for TTML timed text contained MMT Assets

The timed text metadata associated with timed text assets can be contained in an MMT signalling message. Defining the detailed method of the timed text metadata signalling is outside the scope of ISO/IEC 23008-1, therefore it should be defined in the corresponding standard which the timed text service is based on. For example, the closed caption metadata associated with caption MMT assets is signalled by using the caption_asset_descriptor() as defined in ATSC[3].

6.14 Viewport-dependent baseline media profile with packed streaming for VR

Figure 71 shows an example of producing the proposed codec-independent viewpoint-dependent OMAF content. It consists of three main steps:

1. Partitioning the high resolution (HR) ERP video,
2. Down-sampling and rotating the ERP to create the low resolution video,
3. Concatenation of the HR ERP partition with the down-sampled and rotated low resolution (LR) ERP. A guard band is added between the HR and LR content, as shown in Figure 71.

In Figure 71 (a), ERP video content is vertically partitioned into eight ERP partitions with overlap, each of which contains a HR partition of the ERP content. In this example, each ERP partition covers 135 degrees width and the shift between the two adjacent ERP partitions is equal to 45 degrees. In Figure 71

(b), the ERP video is down-sampled and rotated by 90 degrees clockwise to produce the rotated LR ERP content. In Figure 71 (c), each of the eight HR ERP partitions and the rotated LR ERP are concatenated side-by-side to produce the frame-packed OMAF assets.

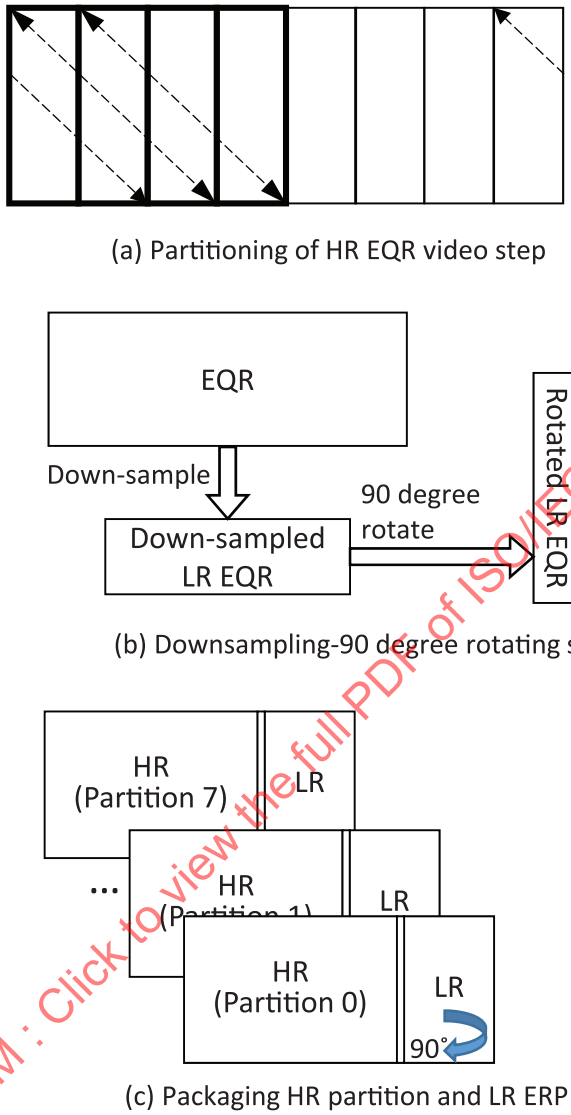


Figure 71 — Packaging viewport dependent OMAF contents

The eight versions of the viewport-dependent frame packed OMAF content are encoded independently using HEVC/AVC to produce the assets, each of which is targeted for a certain viewing direction. Because of the independency in the encoding process, the encoding could be carried out in parallel by deploying multiple encoders. The encoded bitstreams are encapsulated as ISO BMFF track files.

In the receiver side, no additional extension is needed to the ISO BMFF demuxer, nor to the codec in order to parse and decode the frame-packed OMAF content. In order to render the frame-packed OMAF content correctly, the mapping information from the asset to the ERP region needs to be available at the renderer (see Figure 72).

- Each packed content is independently encoded using AVC
- Multiple ISO BMFF streams, each of which is considered as an Asset
- MPEG CI for description of each of the packed contents and mapping it to Asset

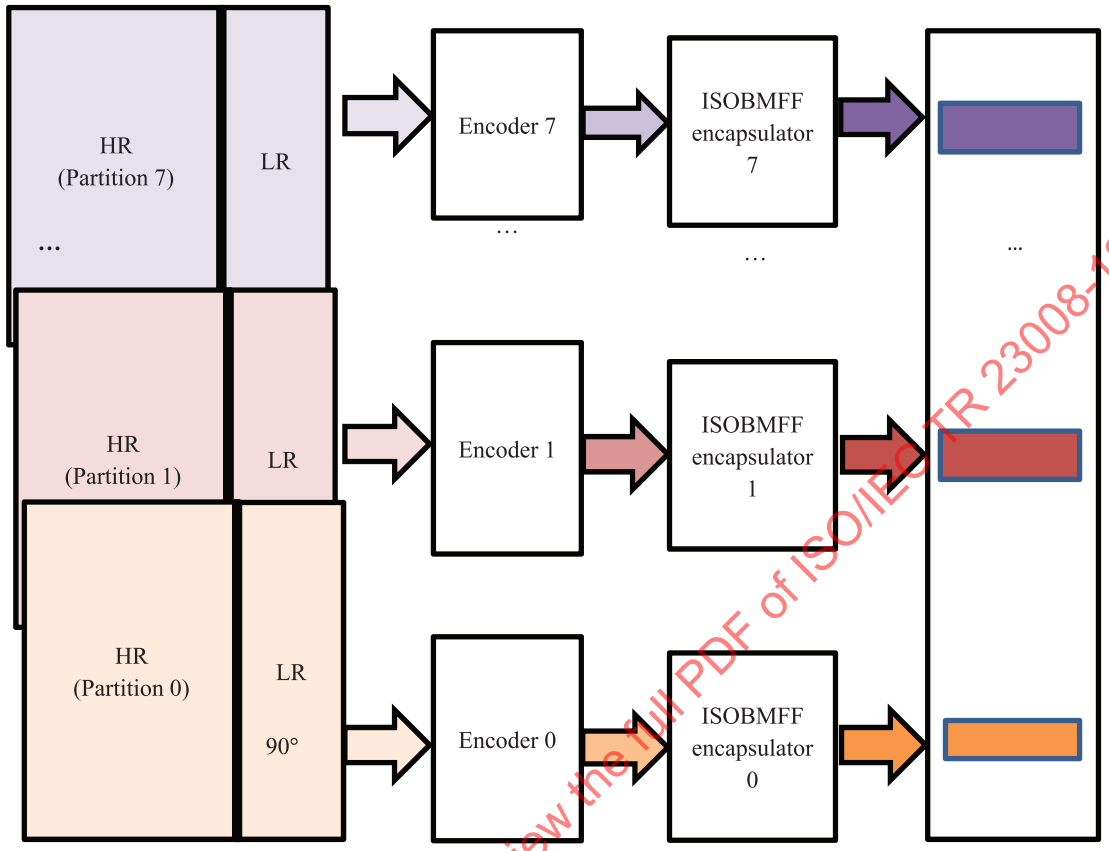


Figure 72 — Multiple ISO BMFF bitstream creation for viewport-dependent streaming

An example implementation of viewport-dependent streaming is shown in Figure 73. In the streaming client, the viewport dependent adaptation set selector module dynamically selects one AdaptationSet from the provided AdaptationSets described in the MPD, based on the viewing direction information retrieved from the HMD. The streaming client receives the viewport-dependent frame packed content, passes it to the ISO BMFF parser for parsing and then to the AVC decoder for decoding. The decoded video sequence is then passed to the rendering module for rendering based on the mapping information, which may be delivered from the server to the client either through the MMT signalling or OMAF.

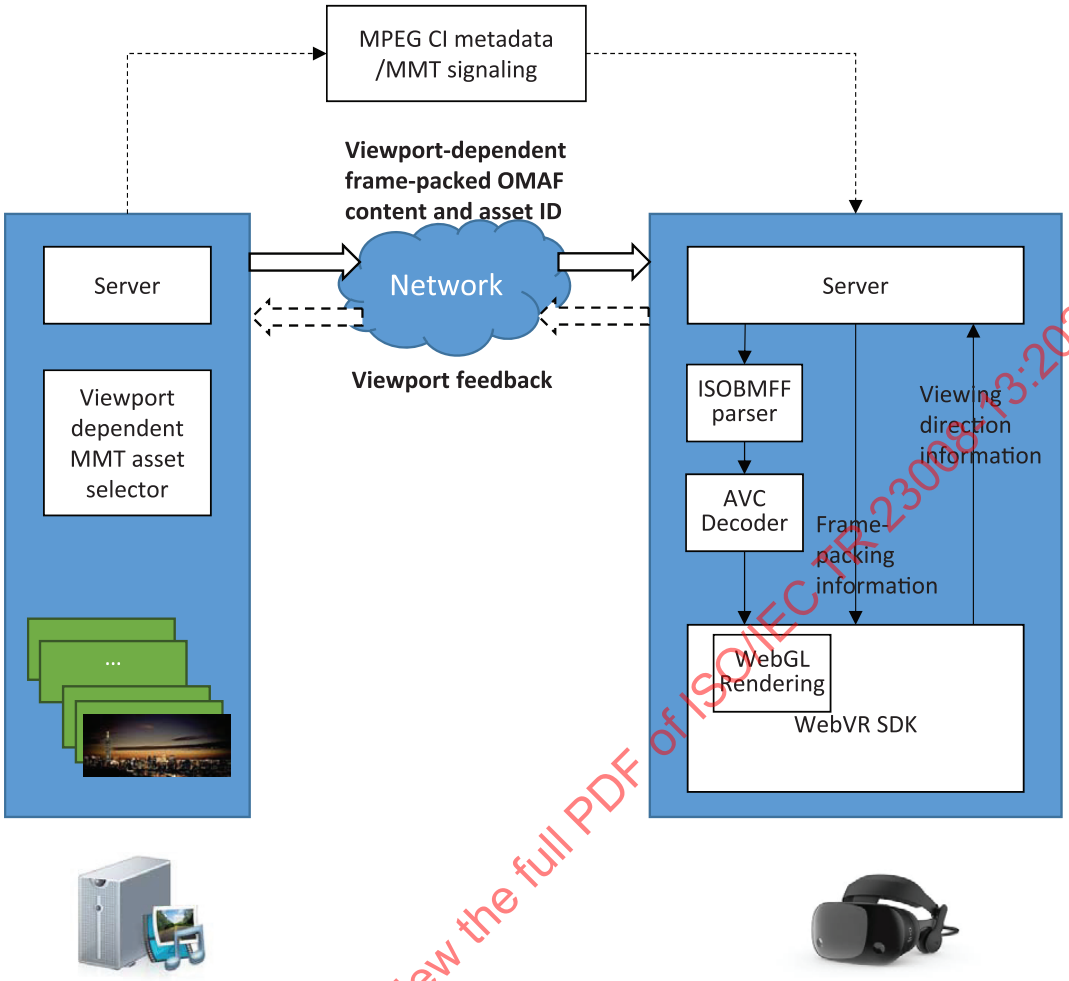


Figure 73 — An example of viewport-dependent frame packed OMAF streaming architecture

7 Application layer forward error correction (AL-FEC)

7.1 FEC decoding method for ssbg_mode2

7.1.1 General

This subclause provides recommendations for the FEC decoding method when MMT employs ssbg_mode2 as source symbol block format. Depending on the FEC encoding scheme, an FEC decoding algorithm can be decided. However, this implementation guidance does not cover a specific FEC decoding algorithm, only the method to choose a proper unit of data for FEC decoding. This clause provides some guidance of FEC decoding unit and a method to choose a proper unit.

7.1.2 Source symbol block format for ssbg_mode2

In ssbg_mode2, a source symbol block (SSB) usually consists of MMTP packets of variable sizes. Figure 74 presents an example of SSB for ssbg_mode2 which is built of 6 MMTP packets having distinct sizes. More precisely, the 6 MMTP packets and some padding data (e.g., all 00h) have been placed into the SSB. Note that any MMT packet should be started at the first byte of a symbol element in SSB. The role of padding data may be regarded as adjusting the start point of MMTP packets.

The columns of SSB in Figure 74 correspond to the source symbols of size T [bytes] which is composed of N(=4) symbol elements of size T/N. In other words, the SSB consists of K(=13) source symbols of size

T, i.e., *K***N*(=52) symbol elements of size *T*/*N*(=*T*/4). Furthermore, an SSB can be divided into *N* regions which consist of *K* symbol elements, respectively, such as Regions-1, 2, 3, 4 in Figure 74. The concept of regions in an SSB will be used for recommended FEC decoding method in 7.1.3.

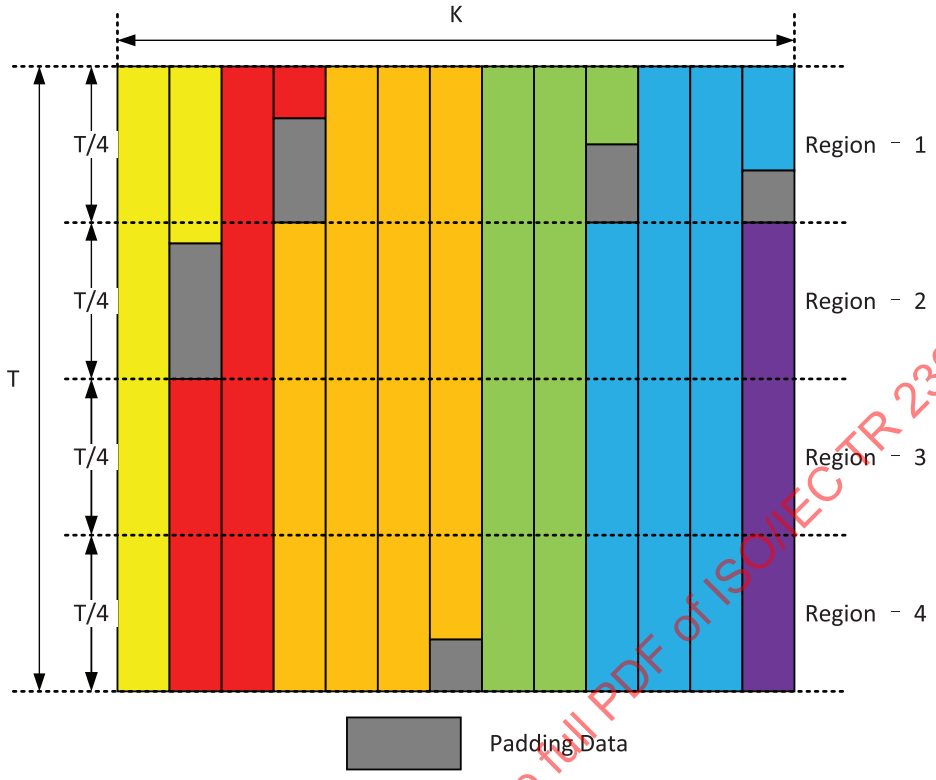


Figure 74 — Example of source symbol block

7.1.3 Regionalization of source symbol Block for FEC decoding

First, assume that the second and fifth MMTP packets in Figure 74 are lost, i.e., two MMTP packets are not received in the MMT receiving entity side. When an MMTP packet is lost, the MMT receiving entity cannot acquire its boundary information in the SSB since its source FEC payload ID and size information are also lost. In other words, the MMT receiving entity cannot acquire the information on the start and end positions of MMTP packet and the amount of padding data, and so on. Therefore, the MMT receiving entity can rebuild SSB as depicted in Figure 75. Note that source FEC payload ID provides information related to the start position of the MMTP packet in SSB in terms of the symbol element for ssbg_mode2. For example, the start position of the second and fifth MMTP packets in Figure 74 in terms of symbol elements may be 6 and 37. After rebuilding the SSB from received MMTP packets, the MMT receiving entity carries out the FEC decoding process to recover the lost MMTP packets.

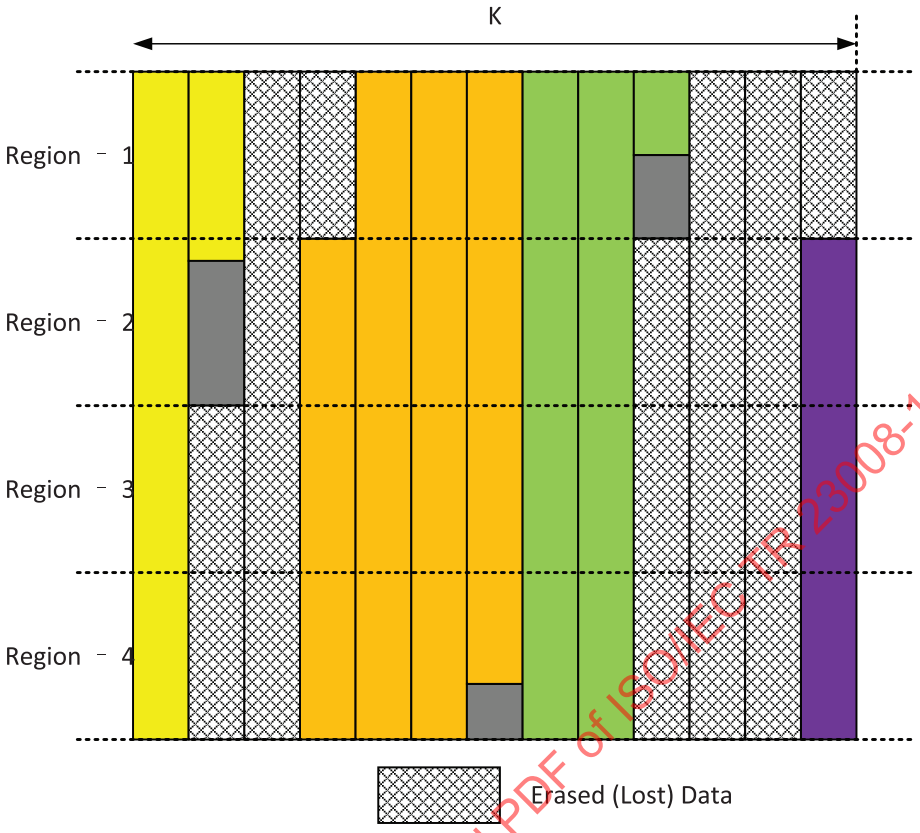


Figure 75 — SSB rebuilt when 2 MMTP packets are lost

The unit of data used during the decoding process can be changed according to decoding requirements, e.g., the decoding complexity, latency and the performance of erasure recovery, etc.

For the first example, Figure 76 presents the FEC decoding method based on source symbol unit. To carry out the decoding process based on source symbol, SSB in Figure 76 should be interpreted into the SSB in Figure 76. It is easily checked that any source symbol including lost MMTP packet is regarded as a lost source symbol. Consequently, the rebuilt SSB has seven lost source symbols, and therefore, at least seven repair symbols are required to recover the lost MMTP packets perfectly.

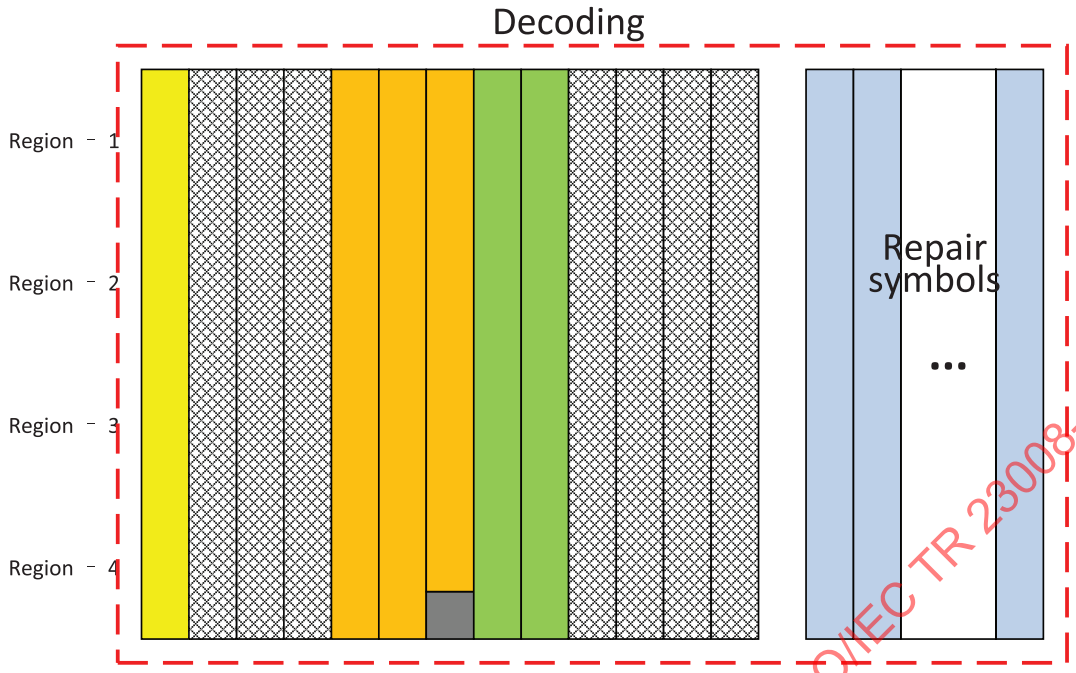


Figure 76 — FEC decoding based on source symbol unit

The advantage of FEC decoding based on source symbol unit is low-complexity decoding due to one erasure pattern during the decoding for the given SSB. More precisely, the preprocessing, e.g., Gaussian elimination to form a decoding schedule, is required only once and the subsequent process is related to simple repeated computations.

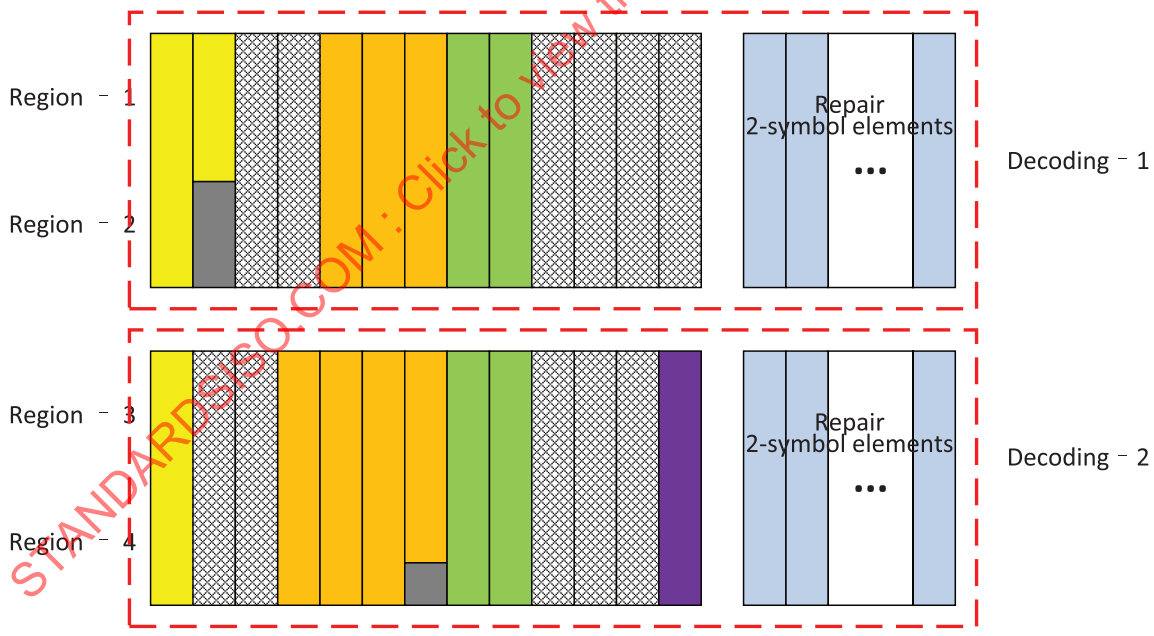


Figure 77 — FEC decoding based on multiple symbol elements unit

For the second example, Figure 77 presents the FEC decoding method based on a 2 symbol elements unit. Here 2-symbol element means a virtual unit for 2 symbol elements bonded in each divided region. To carry out the FEC decoding based on multiple symbol elements, a regionalization step is required. After rebuilding the SSB from received MMTP packets in Figure 74, the SSB should be divided into two regions as depicted in Figure 77. One region consists of Regions-1 and -2, and the other consists of

Regions-3 and -4. Next, any 2-symbol element including lost MMTP packet is regarded as a lost 2-symbol element.

Consequently, one region and the other of SSB in Figure 77 have six and five lost 2-symbol elements, respectively. Therefore, at least six repair symbols are required to recover the lost MMTP packets perfectly. Finally, FEC decoding is carried out with a proper amount of repair symbols for each region. Note that repair symbols also should be transformed into repair 2-symbol elements for FEC decoding.

In general, the erasure patterns of two regions are different in case of FEC decoding based on multiple symbol elements unit. Therefore, the preprocessing for FEC decoding (e.g., Gaussian elimination) should be applied to each divided region in SSB, i.e., two distinct FEC decoding processes are carried out. This causes an increase of decoding complexity compared with FEC decoding based on source symbol, while the performance of erasure recovery can be improved for the given repair symbols since the number of erasures is reduced. Note the number of erasures and their positions for the first and second examples are different.

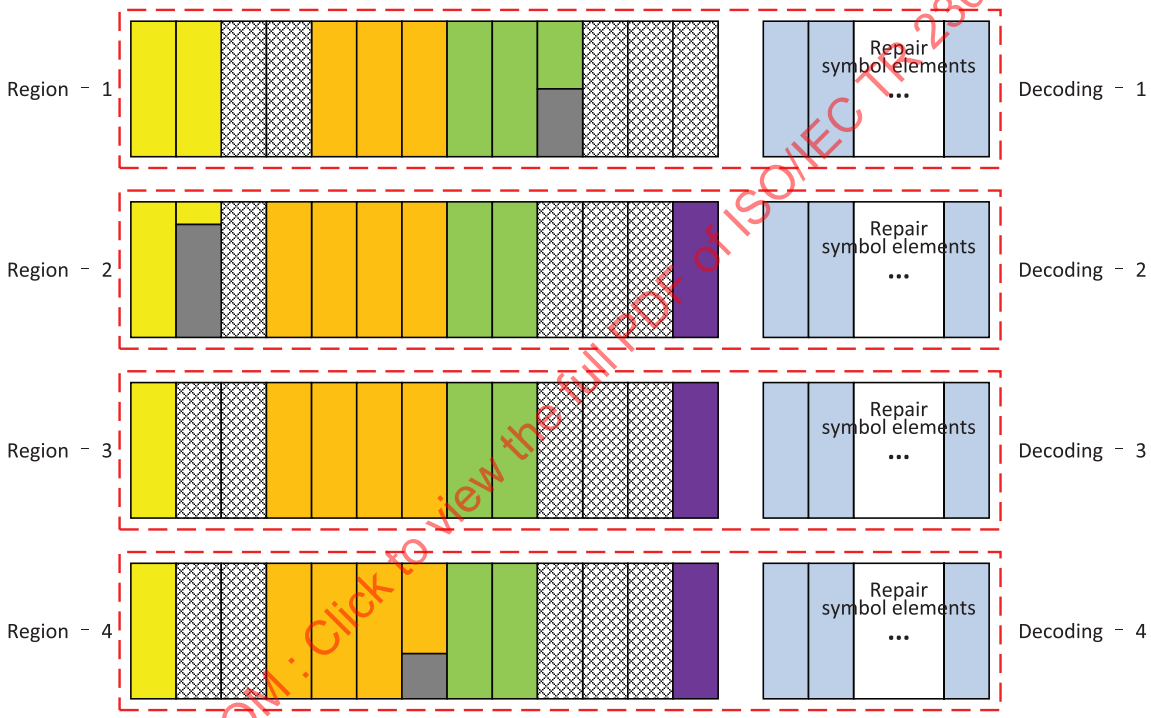


Figure 78 — FEC decoding based on symbol element unit

For the third example, Figure 78 presents the FEC decoding based on symbol element unit. In this case, the SSB should be divided into four regions as depicted in Figure 78. The four regions are the same as Regions-1, -2, -3, and -4 in Figure 78. Furthermore, each region has five, four, five, and five lost symbol elements, respectively. Therefore, at least five repair symbols are required to recover the lost MMTP packets perfectly. Finally, FEC decoding is carried out with a proper amount of repair symbols for each region. Note that repair symbols also should be transformed into repair symbol elements for FEC decoding.

In general, the erasure pattern of each region is different in case of FEC decoding based on symbol elements unit. Therefore, the preprocessing for FEC decoding (e.g., Gaussian elimination) should be applied to each region in the SSB, i.e., four distinct FEC decoding processes are carried out. This causes an increase of decoding complexity compared with FEC decoding based on 2-symbol element, while the performance of erasure recovery can be improved for the given repair symbols since the number of erasures is reduced. Note the number of erasures and their positions for the first, second, and third examples are different.

The unit of data used during the decoding process is related to the decoding complexity and the performance, i.e., there is a trade-off between them. The smaller FEC decoding unit induces the larger

decoding complexity, while its performance of erasure recovery becomes better. Therefore, it is important to choose a proper unit of data for FEC decoding according to system requirements.

7.1.4 How to choose a proper unit of data for FEC decoding

As previously described, the number of erasures depends on the unit of data for FEC decoding, as depicted in Figure 79. It is clear that the smaller unit is, the less erasures are induced. On the other hands, the larger unit is, the smaller decoding complexity is induced. Therefore, it is recommended to choose a symbol element for the best FEC performance and choose a source symbol for the smallest decoding complexity as the FEC decoding unit.

However, if there is not much difference for the number of erasures among the FEC decoding units, it is better for FEC decoder to choose the largest unit possible since the effect of decreasing complexity is more dominant than that of degrading the performance, i.e., the performance degradation may not be critical. At this point, the number of erasures for each unit can be a measure to choose a proper unit of data for FEC decoding. More precisely, after counting erasures for each FEC decoding unit by several counters, compare their values and determine a proper FEC decoding unit based on a predetermined selection rule. For example, if the difference between the numbers of erasures for two decoding units is larger than a predetermined threshold value, the FEC decoder chooses a smaller unit, otherwise, a larger unit.

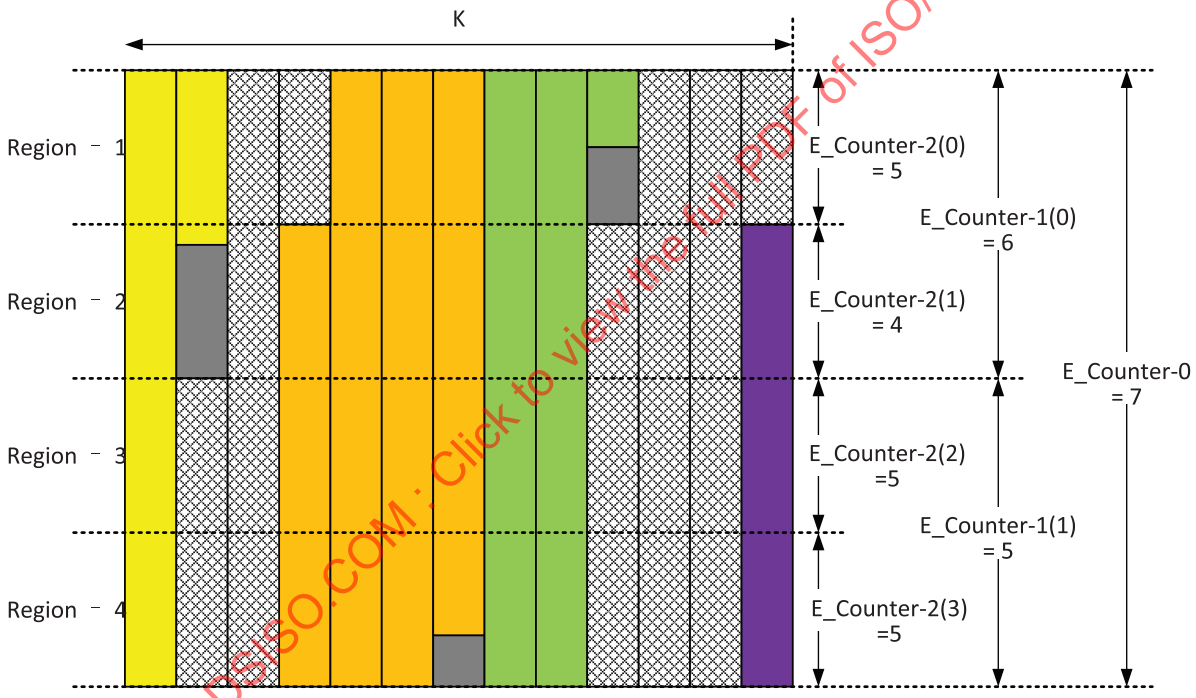


Figure 79 — Example of counting erasures for each FEC decoding unit

7.2 Usage of two stage FEC coding structure

7.2.1 General

For error resilient timed and non-timed data delivery service, an MMT AL-FEC scheme based on block (*N, K*) code is applied. For a given code rate *CR (= K/N)*, FEC recovery performance on the application layer is mainly dependent on the loss rate, loss model and source packet block length *K*. For given packet loss rate and on a given loss model, the greater *K* it is, the lower overhead it requires for target FEC recovery performance while the longer delay it introduces and the more buffer memory it requires. However, the smaller *K* it is, the higher FEC overhead it requires while the less delay (low delay service) it is achievable and the less buffer memory it requires.

Usually, an asset for timed data requires low delay under reasonable FEC recovery performance and an asset for non-timed data does not allow any loss; these delivery characteristics about required QoS for delivery of assets are described at MMT-ADC message. For this, an asset for timed data is delivered and protected with a relatively small encoding symbol block to support low delay and an asset for non-timed data is delivered and protected with a greater encoding symbol block to get higher FEC recovery performance and lower FEC overhead. Therefore, case 2 of the two stage FEC coding structure is used for delivery service of hybrid contents which requires two different QoSs such as AV and File data.

On the other hand, when an asset for timed data such as an AV streaming delivery service is multicast (or broadcast), some end-users (user group A) of the multicast (or broadcast) group can be under relatively good channel conditions (e.g., 1 % packet loss or random packet loss) and the others (user group B) can be under relatively bad channel conditions (e.g., 10 % packet loss or burst packet loss). For this, it is preferable for the asset to be delivered and protected with a relatively small encoding symbol block to provide low delay service to user group A and with a greater encoding block to provide reasonable FEC recovery performance to user group B. Therefore, case 2 of two stage FEC coding structure is used for streaming multicasting (or broadcasting) service of an asset for timed-data.

The MMT HRBM is applied for each asset, i.e., the MMTP packets having the same packet_id. For a two stage coding structure, the MMT HRBM is extended to multiple assets which are protected as an FEC source flow. The HRBM messages for all assets in an FEC source flow should be determined by considering whether FEC2 decoding is applied to each asset or not.

Figure 80 depicts HRBM for a two stage FEC coding structure. MMTP packets (FEC source or repair packets) for both asset A and asset B are passed to both the FEC 1 and FEC 2 decoding buffers. After FEC 1 decoding, MMTP packets for asset A are passed to the de-jitter buffer for asset A and the recovered source symbols are passed to the FEC 2 decoding buffer. Then, if MMTP packets for asset B are still not recovered, MMTP packets for asset B after FEC 2 decoding are passed to the de-jitter buffer for asset B. Otherwise, without FEC 2 decoding MMTP packets for asset B are passed to the de-jitter buffer for asset B.

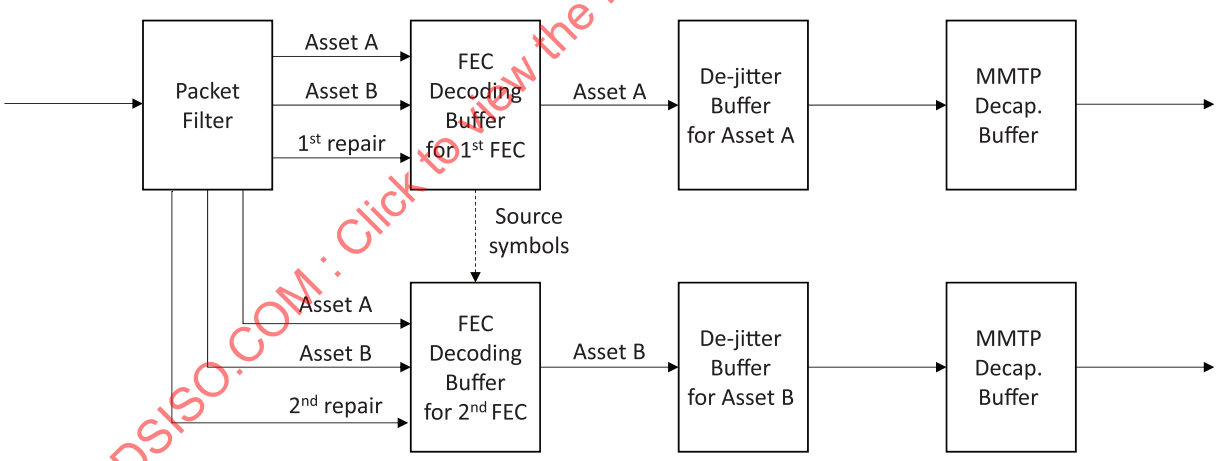


Figure 80 — HRBM block diagram for two stage FEC coding structure

7.2.2 Use case: Hybrid content delivery

When hybrid content, which consists of assets (video and audio for timed-data and file for non-timed data), are delivered, each asset is packetized in MMT payloads and the packetized MMT payloads for the assets are multiplexed on MMT packets to be a single FEC source flow (a sequence of MMT packets for the assets). The single FEC source flow is segmented into one or more source packet blocks and each source packet block is protected by case 2 of the two stage FEC coding structure.

During an FEC decoding process, assets for timed-data are recovered by using the FEC 1 decoder in the split source packet block units to provide low delay service and assets for non-timed data are recovered by using both an FEC 1 decoder in the split source packet block units and an FEC 2 decoder in source packet block units to provide higher FEC recovery performance.

In order to support this use case, the MMT sending entity should send HRBM messages as follows. For each asset for video and audio, the value of *fixed_end_to_end_delay* field is calculated by summing max_transmission_delay and protecion_window_time for FEC1. On the other hand, the value of *fixed_end_to_end_delay* field for file asset is calculated by summing *max_transmission_delay* and *protecion_window_time* for FEC2. Note that the MMTP packets delivering assets for video and audio can be recovered by the FEC 2 decoder, but those packets may not be used in an MPU reconstruction process.

7.2.3 Use case: Streaming multicasting (or broadcasting) to two different end-user groups which is under two different channel conditions each other

When AV content which consists of assets (video and audio for timed-data) are multicast (or broadcast) to two different end-user groups who are under two different channel conditions, each asset is packetized in MMT payloads and the packetized MMT payloads for the assets are multiplexed in MMT packets to be a single FEC source flow (a sequence of MMT packets for the assets). The single FEC source flow is segmented into one or more source packet blocks and each source packet block is protected by case 2 of the two stage FEC coding structure.

During the FEC decoding process, user group A, which is under relatively good channel conditions, recovers the assets by using the FEC 1 decoder in the split source packet block units for low delay service or reducing power consumption; user group B, which is under relatively bad channel conditions, recovers the assets by using the FEC 1 decoder in the split source packet block units and the FEC 2 decoder in source packet block units to get reasonable FEC recovery performance.

7.3 Usage of layer-aware FEC coding structure

7.3.1 General

MMT supports the layer-aware FEC (LA-FEC) as a FEC scheme for improving the error robustness of transmission of layered media such as that specified in MPEG as scalable video coding (SVC), Scalable HEVC (SHVC) or multi-view video coding (MVC). Fine-fort LDGM code (code point 5 of ISO/IEC 23008-10 MMT FEC codes) and RaptorQ LA code (code point 4 of ISO/IEC 23008-10 MMT FEC codes) support the LA-FEC coding structure. The LA-FEC exploits the dependency across layers of the media for FEC construction and in the generation of several repair flows associated to each layer, where each repair flow protects the data of its corresponding layer and the data of all layers, this layer depends on (hereafter referred to as enhancement layer), if any.

The traditional FEC approach to achieve a more efficient delivery of layered media is to apply unequal error protection (UEP) to the video stream, where the most important layers, such as the base layer, have a stronger FEC protection. A more sophisticated protection scheme is represented by the layer-aware FEC (LA-FEC) approach. LA-FEC generates the FEC parity data following existing media coding dependencies within the multi-layer media stream in order to improve the robustness of the more important layers. Using a LA-FEC scheme, FEC parity symbols of less important layers, such as enhancement layers, can be jointly used with FEC parity symbols of more important layers for recovering the source symbols of all participating layers. As such, the more important layers are protected by additional symbols, which increase the error correction capabilities of the more important layers without any increase in terms of bit rate.

In order to generate the FEC parity data following media coding dependencies, LA-FEC specifies a constrained parity matrix. In the case of FF-LDGM code, the parity check matrix is given by the layered sparse matrix specified in ISO/IEC 23008-10:2015, 9.3. RaptorQ LA extends the IETF RFC 6330 code for efficient support of LA-FEC and its parity check matrix is specified in ISO/IEC 23008-10:2015, 8.3. The parameters required for the LA-FEC scheme, e.g., '*fec_coding_structure* = 0011' (indicates the layer-aware FEC coding structure for its associated FEC source flow), *'fec_code_id_for_repair_flow'* (FEC code identifier for its associated FEC repair flow) and *'num_of_layer_for_LAFEC'* (number of layers of the media protected by layer-aware FEC coding structure) are specified in the AL-FEC message.

7.3.2 Use case 1: Layered multicast streaming

Layered video coding such as SVC or SHVC can provide multi-resolution video communication. An example is given in Figure 81 by 4K/2K layered multicast streaming that supports sites that have no 4K capture/projection devices. Note that other resolutions such as VGA/QVGA can be used. Note that any video format can be used for 4K/2K layered video streaming, it allows 4K video sources to be utilized even in 2K-only environments. As shown in Figure 81, the 4K video from the input (4K live camera) is divided into the enhancement layer (EL1) consists of 4K sub-band data and the base layer (BL) consist of 2K sub-band data. The former is passed to an IP multicast group that lies within the IP multicast group of the latter. At the decoder side, the decoder joins and receives multicast groups depending on what resolution is required by the user. The LA-FEC allows the FEC-encoded flow of EL1 to be used in BL; this increases the total block length and raises the probability of recovering the base layer.

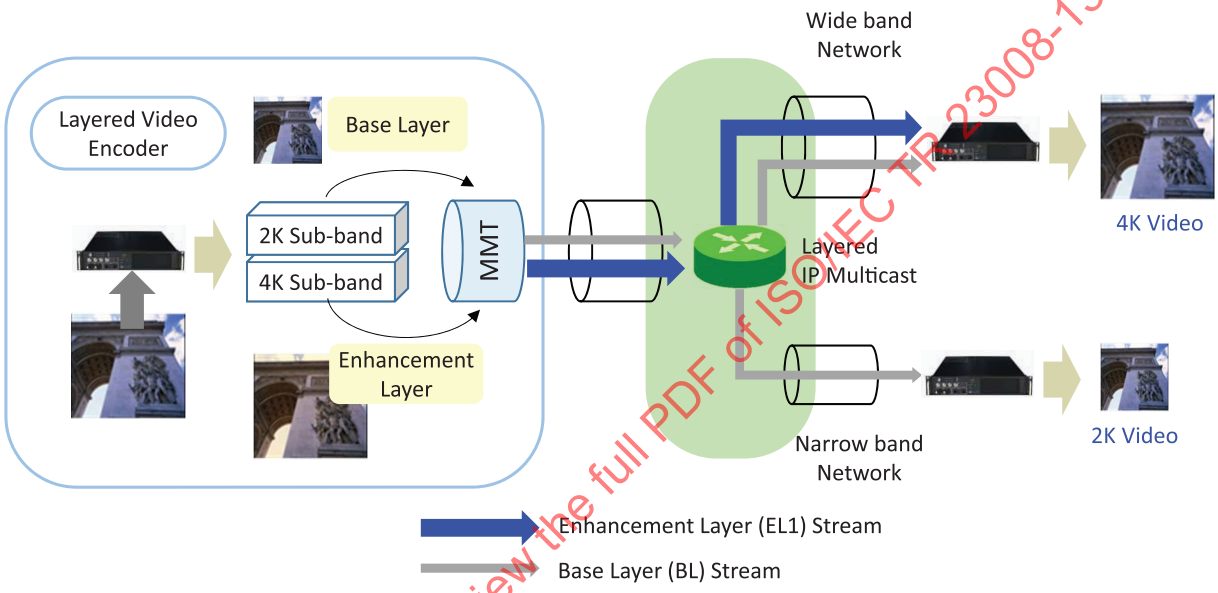


Figure 81 — Multi-point 4K/2K teleconference

Figure 82 shows a multi-point 4K and 2K teleconference application using 4K/2K layered multicast streaming. This is a three site tele-conferencing example; two sites communicate by 4K video while the remaining site uses a 2K system. Using layered multicast streaming, selection of decoding 2K or 4K is simply defined by the multicast configuration at IP routers in the networks, without any redundancy even though 4K and 2K were transmitted simultaneously. The LA-FEC can efficiently recover erasure packets in this situation.

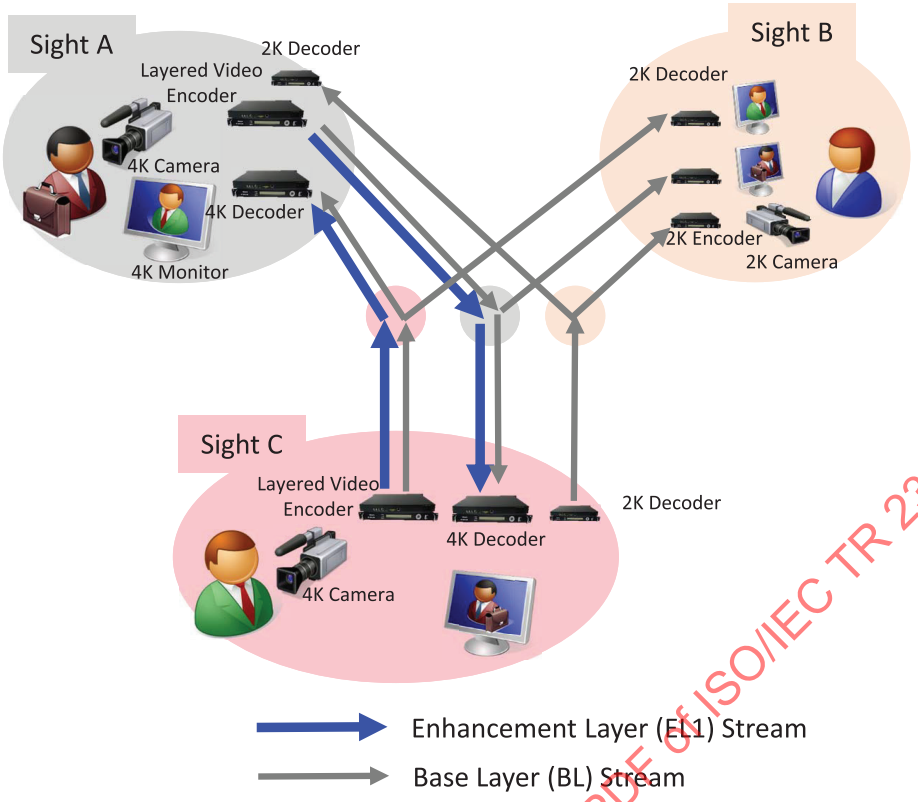


Figure 82 — Multi-point 4K/2K teleconference

The LA-FEC also supports two or more layers. For example, 8K, 4K and 2K layered multicast streaming can also be realized. Furthermore, the LA-FEC cannot only apply to the multi-resolution layered multicast streaming but also to other ones. For example, by using the layered video coding which supports SNR scalability, video streams can be separated into layers by distributing qualitative (SNR) factor.

7.3.3 Use case 2: Hybrid delivery

The scenario for a use case of layered multicast streaming can also be applied to a hybrid delivery scenario, where a base layer (BL) is delivered over one network link (e.g., cellular network) and the enhancement layer (EL1) over a second network link (e.g., WiFi). Figure 83 illustrates such a hybrid delivery scenario with two exemplary receivers, one (receiver 2) receiving the BL only (e.g., 2K video) and the other (receiver 1) receiving a higher video quality with BL and EL1 (e.g., 4k video).

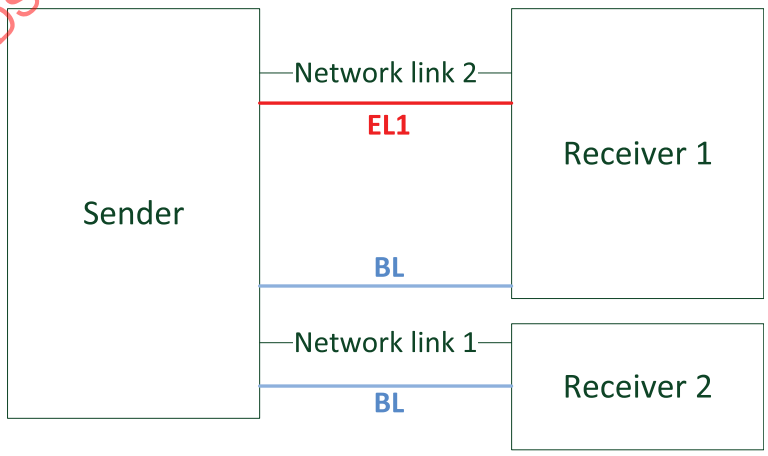


Figure 83 — Network scenario of hybrid delivery

In such a scenario, LA-FEC allows the correction of a non-correctable source packet block in network link 1 in a situation where, at the same time, the instance data of network link 2 is well received.

7.3.4 Use case 3: Fast zapping with long time interleaving (LA-FEC UI)

This subclause describes LA-FEC with unequal time interleaving (LA-FEC UI), a possible sending arrangement that allows MMT service robustness against long burst errors while at the same time having fast tune-in to the service. This sending arrangement assumes that LA-FEC is activated, i.e., *fec_coding_structure*=0011. The scheme is illustrated in Figure 84 by the example of a single MMT multiplex. Figure 84 shows two FEC encoded flows (FEF), with FEF1 comprising the data of a base layer and FEF2 comprising data related to an enhancement layer. The two different tones of shaded rectangles denote packets belonging to the same source packet block, with the first half blocks belonging to data succeeding the latter half blocks in presentation order. The empty boxes denote other data, e.g., preceding or succeeding data.

Figure 84 shows a sending arrangement with unequal time interleaving that follows the settings defined for fast zapping solution with LA-FEC. With LA-FEC UI, the protectionWindow (FEF2) is increased compared to protectionWindow (FEF1) by spreading the data of FEF2 over a longer time period and interleaving it with other data. The reception of the last packet of FEF1 and FEF2 happens at a similar time instance, in the example at time instance t_2. As described in ISO/IEC 23008-1, the protectionWindows (FEF1) and (FEF2) are increased to cover the reception of all packets of FEF1 and FEF2 succeeding the first packet of FEF1. With the described solution, a receiver that tunes at a time into the MMT stream can start decoding the FEF1 after a time defined by protectionWindow (FEF1). After a transition time, which depends on the difference between the protection windows, the receiver can start decoding the FEF2 stream. After starting processing FEF2, the robustness of the FEF1 stream with protectionWindow (FEF1) is increased by the longer protectionWindow (FEF2) due to the LA-FEC protection. Since the data of FEF2 is transmitted ahead of the data of FEF1, the data of FEF1 needs to be buffered on the transmitter. This buffering period depends on the difference between both protectionWindows.

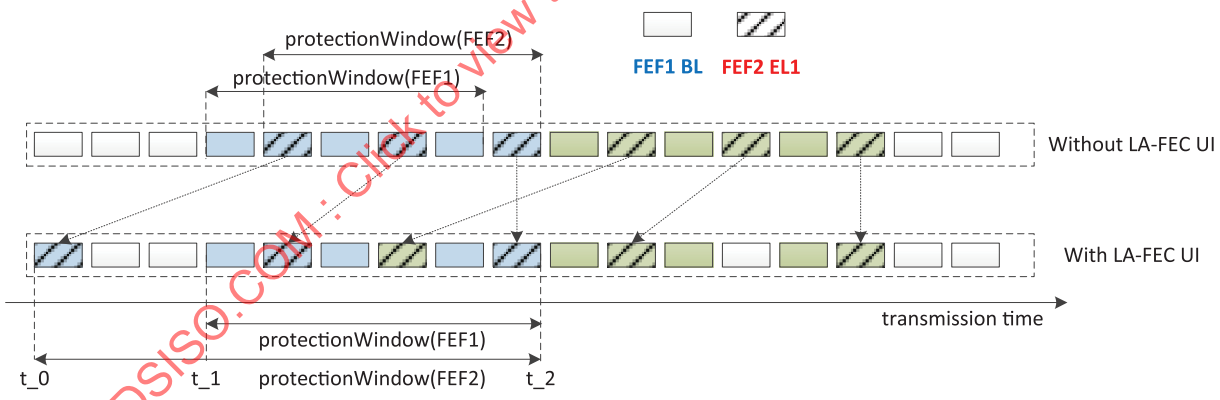


Figure 84 — Sending arrangement of an MMT multiplex with and without long time interleaving with fast tune-in by LA-FEC UI

7.3.5 Use case 4: Prioritized transmission

A typical deployment scenario for layered media codecs is to support different devices capabilities in a broadcast/multicast scenario. This can be, e.g., the support of 2D/3D devices by MVC or different resolutions such as 2k/4k or 720p/1080p or QVGA/VGA by SVC or SHVC. A layered transmission also allows providing different robustness for each layer by AL-FEC solutions, which is especially interesting to save bandwidth and keep a stable service in difficult reception conditions such as in Mobile TV case.

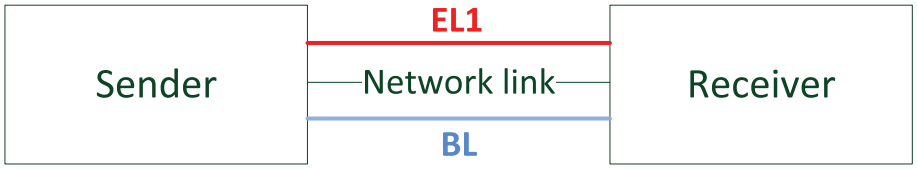


Figure 85 — Network scenario for prioritized transmission

With layered media coding, the enhancement layer (EL1) depends on the base layer (BL) due to inter-layer prediction. Therefore, the decoding probability of EL1 depends on the decoding probability of BL. More concretely, EL1 inherits the decoding probability of BL as its maximum decoding probability. With LA-FEC, the decoding probability of BL increases with the reception of EL1 due to the additional protection given by the protection of EL1. Therefore, the achievable maximum decoding probability of EL1 increases as well (Figure 85).

7.4 MPU mapping to source packet block

7.4.1 General

An MMT FEC scheme can be applied to protect MMT assets. An FEC source flow is a flow of MMTP packets delivering one or more MMT assets. MPUs composing the MMT assets are packetized into MMTP packets. The sequence of MMTP packets is segmented into source packet blocks and FEC encoding is applied to those blocks. The resulting encoding symbols are delivered by FEC source and repair packets.

In order to recover lost MMTP packets, the FEC decoding needs to collect a sufficient number of encoding symbols from the FEC source and parity packets. An MPU could be de-packetized only after the FEC decoding process ended for all FEC source packet blocks containing any MMTP packet from the MPU. As a result, there is a different delay dependent of how to map MPUs to source packet blocks. Therefore, MMT needs a strategy for mapping MPUs to source packet blocks to prevent unintended delay in MMT client.

7.4.2 Aligned MPU mapping method to source packet block

Firstly, it will be assumed that an FEC source flow is a flow of MMTP packets delivering a single asset. For AL-FEC encoding of the MPUs for the asset, the MPUs are packetized in MMTP packets, and these MMTP packets are mapped to source packet blocks. In this process a source packet block contains one or more complete MPUs or part of a single MPU. More precisely, the MMTP packets from the asset are mapped to source packet blocks in one of following three cases to minimize the decoding delay.

— Case 1: A source packet block only contains a complete set of MMTP packets packetized from a single MPU of the asset.

— Case 2: The MMTP packets packetized from a MPU of the asset are mapped to N (>1) source packet blocks. These source packet blocks contain only MMTP packets packetized from the MPU.

— Case 3: A source packet block only contains a complete set of MMTP packets packetized from M (>1) MPUs of the asset.

Figure 86 show examples for the three cases for “aligned MPU mapping to source packet block (SFB)” in case of that FEC source flow consists of MMTP packets for a single asset. In this figure “Case 1 Ex.” is a example for case 1, “case 2 Ex.” for case 2 with $N = 2$ and “case 3 Ex.” for case 3 with $M = 2$.

The aligned MPU mapping to source packet block can be easily extended to the cases where an FEC source flow is a flow of MMTP packets delivering more than one MMT assets. The design concept of “aligned MPU mapping to source packet block” is to minimize the unintended delay caused by AL-FEC protection. Some MPUs in different assets can have a close relationship and be multiplexed and then considered as a unit. This kind of set of MPUs can be referred to as a group MPU (GMPU). Then, in the

encoding process of the source flow, a source packet block contains one or more complete GMPUs or part of a single GMPU. More precisely, the MMTP packets from the assets are mapped to source packet blocks in one of following three cases to minimize the decoding delay.

— Case 1: A source packet block only contains a complete set of MMTP packets packetized from a single GMPU of the assets.

— Case 2: The MMTP packets packetized from a GMPU of the assets are mapped to *N* (>1) source packet blocks. These source packet blocks contain only MMTP packets packetized from the GMPU

— Case 3: A source packet block only contains a complete set of MMTP packets packetized from *M* (>1) GMPUs of the assets.

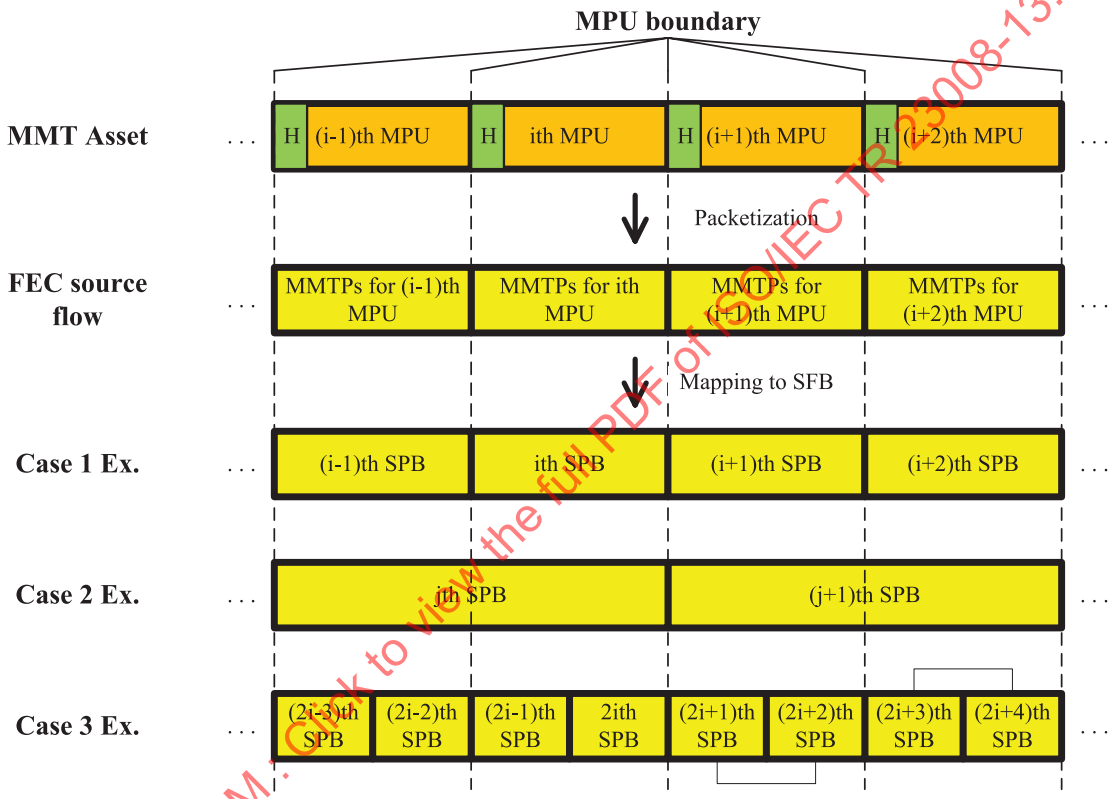


Figure 86 — Examples for MPU mapping to source packet block (SFB)

7.5 FEC for hybrid service

For hybrid services (e.g., primary audio and video broadcasting and secondary audio multicasting via broadband network) distributed by using the broadcast and broadband networks, a repair flow is created to protect a source flow by FEC encoding based on the MMT AL-FEC framework (see Figure 87). When the source flow is delivered via broadband network, the repair flow is delivered via broadcast network. A broadcast network provides relatively low delay while a broadband network introduces relatively long delay, broadcasting delivery of repair flow enables the reduction of the jitter on the MMTP packets for the source flow, by recovering the lost MMTP packets which may arrive after fixed end to end delay (i.e., *maximum transmission delay* + *FEC protection window time*) which is provided by the HRBM message.

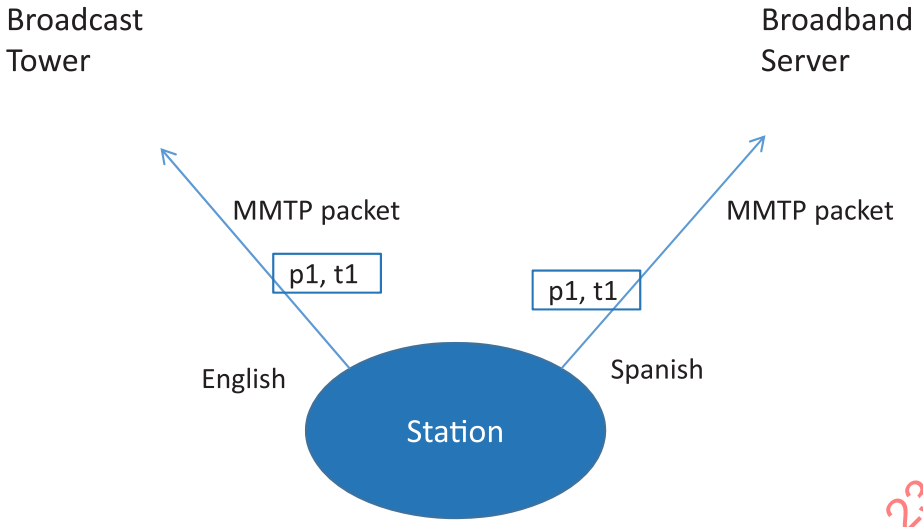


Figure 87 — Hybrid service for primary and secondary audios

When the MMT sending entity wants to provide the broadband contents with *fixed end to end delay* = *D*, the MMT sending entity creates HRBM message(s) with fixed end to end delay = *D and maximum transmission delay* = *D* – *F* (*F* is FEC protection window time) and generates a repair flow to protect the source flow of the broadband contents. In this case, repair symbols are generated as many as to be able to recover the MMTP packets of each source packet block which may be arrived after fixed end to end delay (see Figure 88).

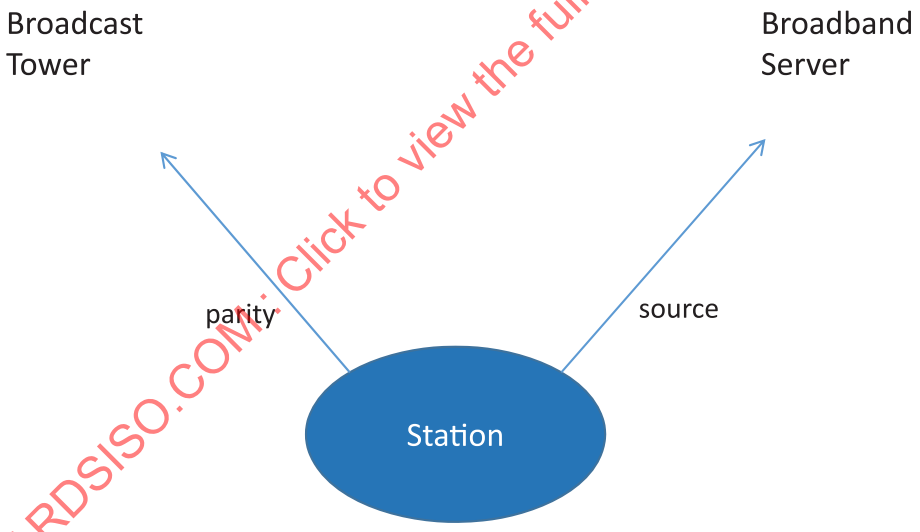


Figure 88 — Broadcasting parity for broadband source

For example:

— maximum delay of broadband network is 5 seconds;

— broadcast network has 1 second constant delay;

— 80 % of MMTP packets for a source packet block arrive within 2 seconds at MMT receiving entity;

— 10 % of MMTP packets for the source packet block arrive within between 2 and 3 seconds;

— the remained 10 % of MMTP packets for the source packet block arrive between 3 and 5 seconds.

Then:

— MMT sending entity creates HRBM message with *fixed end to end delay* = 3 seconds and *maximum transmission delay* = 2 seconds;

— and generates repair symbols as many as to be able to recover 20 % of MMT packets (for the source packet block) which are assumed to arrive between 2 and 5 seconds, based on *FEC protection window time* = 1 seconds and the recovery performance of FEC code algorithm to be used.

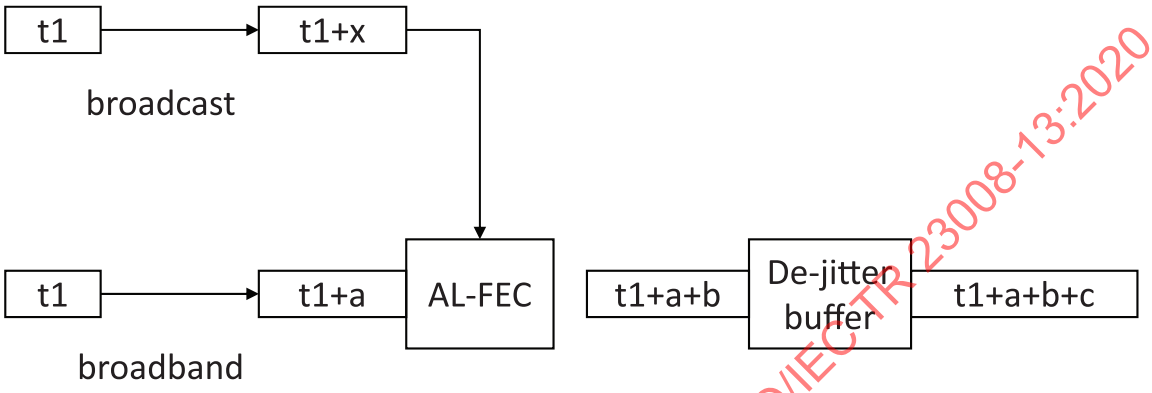


Figure 89 — Jitter reducing mechanism by broadcasting repair

In Figure 89, $x \leq 1$, $a \leq 2$, $a + b \leq 3$ and $a + b + c = 3$.

Then, FEC source packets for the source packet block are delivered over UDP or TCP/IP via broadband network and the FEC repair packets for the repair symbols are delivered over UDP/IP via broadcast network.

7.6 Usage of rate-adaptive AL-FEC

7.6.1 General

ISO/IEC 23008-10 specifies AL-FEC codes. It defines six AL-FEC codes, each of which generates a repair symbol block from a source symbol block as shown in Figure 90. The source symbol block consists of K source symbols of size T (in bytes) and the repair symbol block consists of P repair symbols of size T (in bytes).

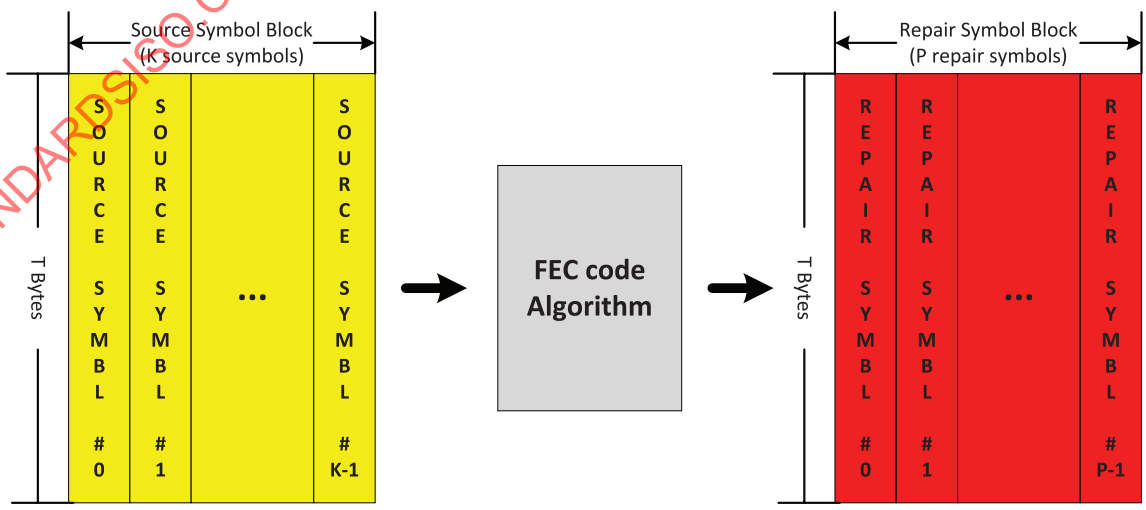


Figure 90 — Generation of repair symbol block

Rate-adaptive AL-FEC dynamically adjusts coding rate $K/(K+P)$ by referencing QoS related information such as packet loss ratio and peak bitrates. MMT defines the QoS related signalling messages such as reception quality feedback (RQF) message and network abstract for media feedback (NAMF) message.

MMT specifies AL-FEC configuration information such as "AL-FEC message" and "FEC payload ID". The associated AL-FEC configuration information of the rate-adaptive scheme is updated based on the QoS-related signalling messages.

Generally speaking, several matrices need to be prepared in advance or a different matrix should be generated for AL-FEC code in order to adjust the coding rate to suit the communication channel. FF-LDGM code (code point 5 of ISO/IEC 23008-10 MMT FEC codes) supports the efficient rate-adaptive scheme. In the case of the rate-adaptive mode of the FF-LDGM code, there is no need to prepare or generate several matrices. The systematic part of the parity check matrix is a punctured sparse matrix, which is created from the mother matrix. A puncture operation example of generating a punctured matrix is shown in Figure 91 (refer to ISO/IEC 23008-10 for details).

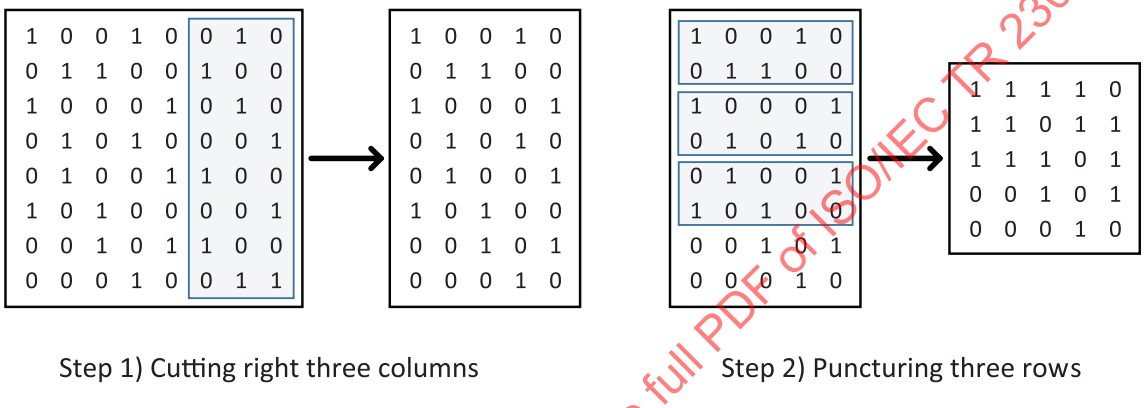


Figure 91 — Puncture operation example of generating punctured matrix

In the process of generating the mother matrix, the so-called the selective progressive edge-growth (PEG) algorithm is used. The selective PEG algorithm improves coding performance for the rate-adaptive mode compared with a simple PEG algorithm.

7.6.2 AL-FEC rate control

7.6.2.1 General

ISO/IEC 23008-10 specifies a detailed algorithm for FF-LDGM code including rate-adaptive mode. Here, usage of the rate-adaptive scheme of FF-LDGM code combined with QoS-related feedback messages is shown. First, QoS feedback messages and AL-FEC configuration information related to the rate-adaptive scheme are shown, followed by the procedure of rate control.

7.6.2.2 QoS-related feedback messages

The coding rate can be changed according to QoS-related feedback messages. As best effort networks do not provide any network level QoS control. MMT provides end-to-end QoS information. MMT defines several QoS-related signalling messages such as reception quality feedback (RQF) message and network abstract for media feedback (NAMF) message. The following parameters of RQF and NAMF messages can be used for the rate-adaptive AL-FEC scheme:

— RQF message

packet_loss_ratio – the ratio of the number of lost MMTP packets to the total number of transmitted packets.

— NAMF message

peak_bitrate – the maximum allowable bitrate that the underlying network can assign to the MMT stream.

available_bitrate – the bitrate that the scheduler of the underlying network can guarantee to the MMT stream.

relative_available_bitrate – the available bitrate change ratio (%) of the current NAM information to the previous NAM information.

7.6.2.3 AL-FEC configuration information

MMT specifies AL-FEC configuration information such as "AL-FEC message" and "FEC payload ID". The following detail the AL-FEC configuration information related to the rate-adaptive scheme.

— AL-FEC message

fec_coding_structure – indicates the applied AL-FEC coding structure for its associated FEC source flow. In the rate-adaptive mode of FF-LDGM code, it is set to "b0001" (one stage FEC coding structure).

maximum_k_for_repair_flow – indicates the maximum allowed number of source symbols in a source symbol block for its associated FEC repair flow.

maximum_p_for_repair_flow – indicates the maximum allowed number of repair symbols in a repair symbol block for its associated FEC repair flow.

fec_code_id_for_repair_flow – FEC code identifier for its associated FEC repair flow. In order to use FF-LDGM code, it is set to "b0101".

protection_window_time – indicates the maximum time duration in milliseconds between the sending of the first source or repair packet of an FEC source or repair packet block and the sending of the last source or repair packet of that FEC source or repair packet block.

protection_window_size – indicates the maximum number of source and parity payloads from the same FEC encoded flow that can be sent from the first payload of an FEC block (source packets and parity packets) to the last payload of that FEC block.

private_field_length – indicates the size in bytes of the private field.

private_flag – when set to '1', this flag indicates there is a private field describing private FEC information consisting of specific FEC algorithm parameters. This "*private field*" is required to describe FF-LDGM specific parameters.

FF-LDGM specific parameters are described in the private field of AL-FEC message and the format is written as shown in Figure 92.

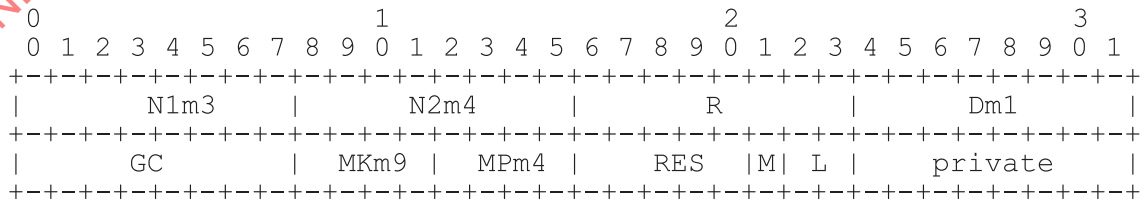


Figure 92 — FF-LDGM code specific parameters

M – a flag indicating use of rate-adaptive mode. It is set to "1" when using the rate-adaptive scheme.

— RepairFEC payload ID