TECHNICAL REPORT

ISO/IEC TR 23008-13

First edition
2015-12-01

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

Part 13: MMT implementation guidelines

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

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

Reference number
ISO/IEC TR 23008-13:2015(E)

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

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).

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 23008 consists of the following parts, under the general title *Information technology — High efficiency coding and media delivery in heterogeneous environments*:

— *Part 1: MPEG media transport (MMT)*

— *Part 2: High efficiency video coding*

— *Part 3: 3D Audio*

— *Part 4: MMT Reference and Conformance Software*

— *Part 5: Reference software for high efficiency video coding*

— *Part 8: Conformance Specification for HEVC*

— *Part 10: MPEG Media Transport Forward Error Correction (FEC) Codes*

— *Part 11: MPEG Media Transport Composition Information*

— *Part 12: Image file format*

— *Part 13: MMT implementation guidelines* [Technical Report]

Introduction

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

— guidelines on usage of MMT functions;

— guidelines on deployment use cases designed based on ISO/IEC 23008-1.

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

Part 13: MMT implementation guidelines

1 Scope

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

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

3 Terms, definitions, symbols and abbreviated terms

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

4 Overview

4.1 System overview

This subclause 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 Chanrateristics, 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.

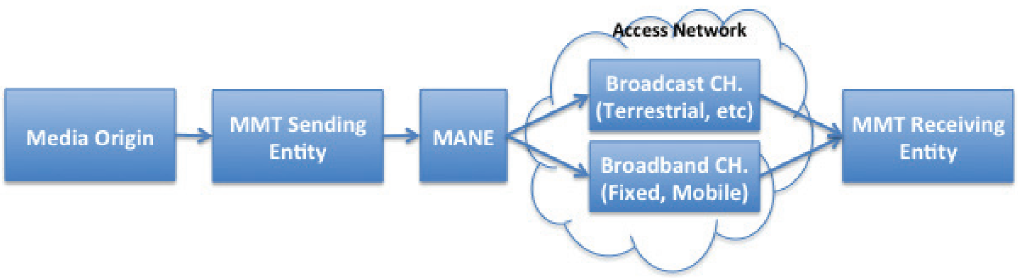

Figure 1 — Example of MMT-based media distribution chain

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 MMT receiving entity has to follow. This information is provided by MMT sending entity and also pushes related MMTP packet flow to the MMT receiving entity. It means it fully controls the media streaming session, i.e. it manages the on-time delivery, playback and temporal/spatial presentation information of the media.

4.2 Normative parts

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

— The Media Processing Unit (MPU) defines the logical structure of media content format of the data units to be processed by an MMT entity and their instantiation with ISO Base Media File Format as specified in ISO/IEC 14496-12.

— The delivery function defines an application layer transport protocol and a payload format. The MMTP transport protocol provides enhanced features for delivery of multimedia data, e.g. multiplexing and support of mixed use of streaming and download delivery in a single packet flow. The payload format is defined to enable the carriage of encoded media data which is agnostic to media types and encoding methods.

— The signalling function defines formats of signalling messages to manage delivery and consumption of media data.

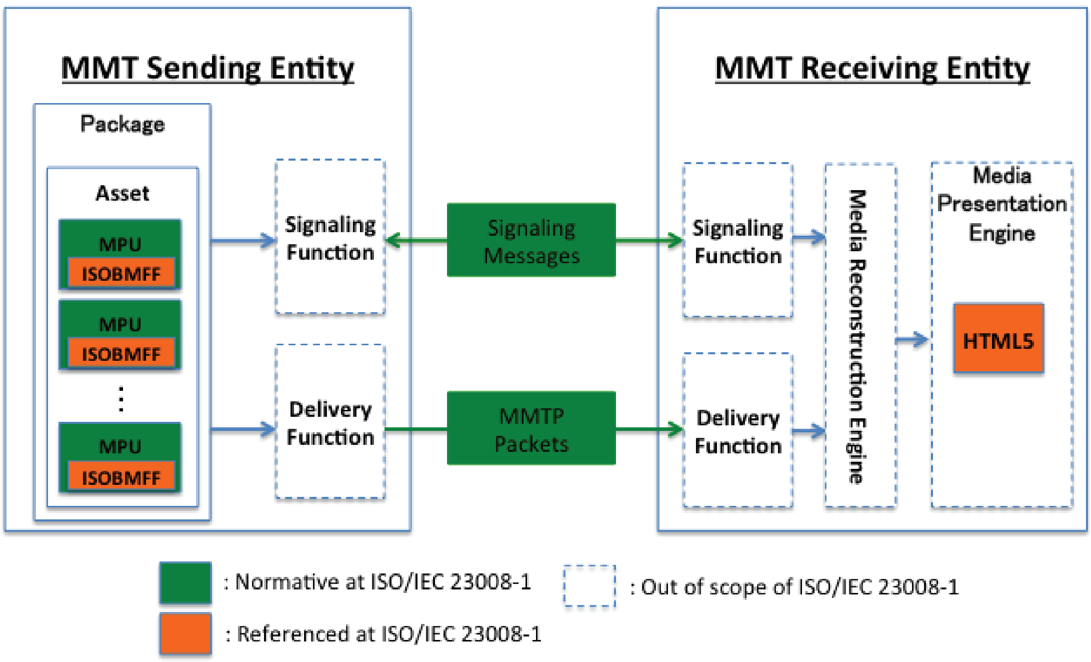

Figure 2 — MMT functions deployment

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

5 MMT function deployments

5.1 Overview

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

— low delay media consumption;

— media adaptation;

— hybrid delivery;

— error recovery.

5.2 Object reconstruction

5.2.1 Overview

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

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

5.2.2 Recovery in MPU mode

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

The MMT receiving entity performs the following steps.

a) Receive MMTP packet.

b) Check if packet_id is equal to the packet_id of the object flow of interest, discard packet and go to step a) if it does not belong to an object flow of interest.

c) Assert that type of the MMTP packet is MPU.

d) If fragmentation flags are set (different than “00”)

   1) if fragmentation flag is equal to “11”, attempt to recover packet and if successful go to step f, or

   2) add packet to the list of packet fragments based on the MMTP sequence number and go to step a).

e) If Aggregation flag A is set, extract all aggregated data units and proceed to step g) for each extracted data unit.

f) If object map with same MPU_sequence_number does not exist, create new object map for the MPU with that sequence number.

g) Check fragment type (FT) of the MPU payload header.

   1) If FT is MPU metadata

      i) check if MPU metadata is already received

         aa) if yes, discard the MPU metadata as being a duplicate, or

         bb) insert MPU metadata at the beginning of the object map

         — optionally, forward MPU metadata to application.

      ii) Go to step a)

   2) If FT is Fragment metadata

      i) Check if movie fragment with the same movie_fragment_sequence_number already exists

         aa) if no, create a placeholder for the movie fragment in the object map, or

         bb) check if Fragment metadata has already been received.

         — if yes, discard fragment metadata as being a duplicate, or

         — insert fragment metadata at the beginning of the fragment placeholder

         cc) Go to step a)

   3) If FT is MFU

      i) If fragment placeholder with sequence number movie_fragment_sequence_number does not exist in the object map of the MPU with sequence number MPU_sequence_number, then create movie fragment placeholder in the object map of the MPU.

      ii) If timed metadata flag is set

         aa) insert payload in the fragment placeholder in the correct order based on the sample_number and offset values,

         bb) check if movie fragment is complete, and

         — If yes, forward fragment to the application

         cc) go to step a).

      iii) If timed metadata flag is not set

         aa) insert payload in the item in the object map based on the item item_ID,

         bb) recover item information from MPU metadata for the recovered item and forward the item to the application, and

         cc) go to step a.

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

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

5.2.3 Recovery in GFD mode

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

The operation of the MMTP receiver is as follows.

a) Receive MMTP packet.

b) Check if packet_id is equal to the packet_id of the object flow of interest, discard packet and goto step a if it does not belong to an object flow of interest.

c) Assert that type of the MMTP packet is GFD.

d) If object map with same TOI does not exist, create new object map for the file with that TOI.

e) Insert payload in the correct place in the object map using the start_offset information.

f) It is recommended that chunks of contiguous byte ranges that lie between two MMTP packets with the RAP flag R set to 1 be forwarded to the application. Applications may choose to forward sufficiently large contiguous byte ranges whenever they are recovered correctly.

g) If complete TOI is recovered

   1) extract metadata from the transport object or from the GFD table, and

   2) forward file to the application.

5.3 Default assets

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

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

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

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

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

5.4 Low-delay live streaming

MMT streaming is based on the MPU concept, which in turn, is an ISO-based media file format (ISOBMFF) with certain restrictions. However, the usage of the ISOBMFF can give the impression of high end-to-end delay, which might 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.

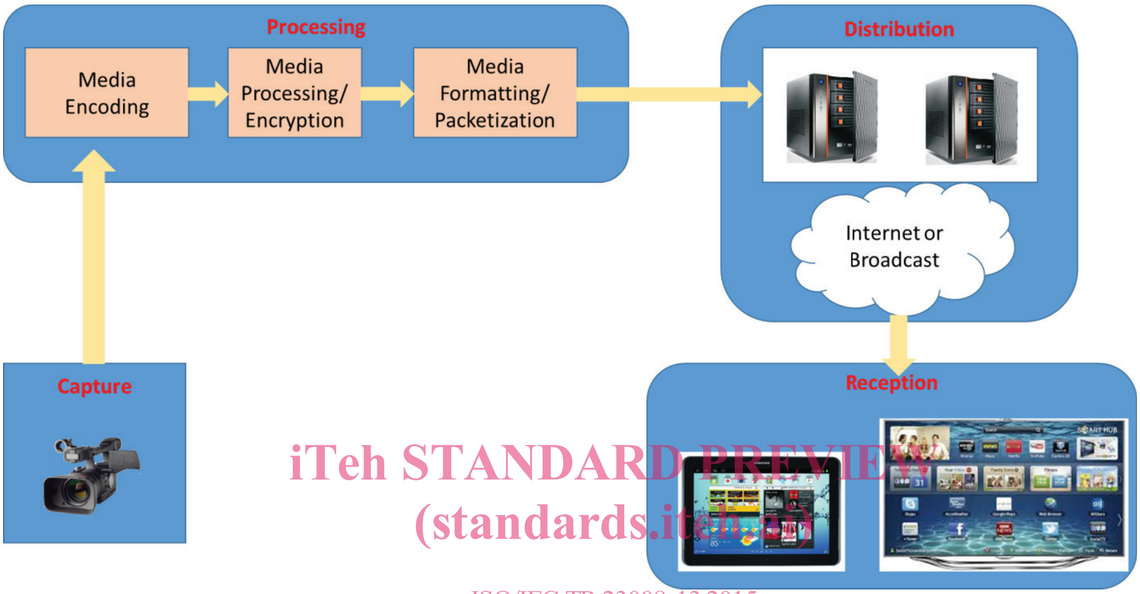

Figure 3 — Example of a broadcast scenario

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

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=0|C|FEC|r|X|R|RES|    type     |          packet_id            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          timestamp                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    packet_sequence_number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        packet_counter                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            length             |  FT   |T|f_i|A| frag_counter  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      MPU_sequence_number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           DU_length           | movie_fragment_sequence_number|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|movie_fragment_sequence_number |         sample_number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         sample_number         |            offset             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            offset             |   priority    |  dep_counter  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Data Unit payload (NAL unit or AAC frame)            |
|                               ...                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     source_FEC_payload_ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


Figure 4 — MMTP packet

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

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

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

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

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

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

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

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

5.5 Parallel processing in MMT sending and receiving entities

5.5.1 Processing in MMT sending entity

MMT protocol performs parallel generation of MMTP flows and signalling message generation/processing. The data packet processing part of MMT protocol performs the packetization of the media data using either the MPU mode for MPUs or the GFD mode for generic files. The generated source data is encapsulated into MMTP packets and transmitted to transport layer. The signalling message generation part of MMTP processes CI, ADC, and other data from the MMT package and encapsulates them into signalling messages and packetizing and passing them to the transport layer.

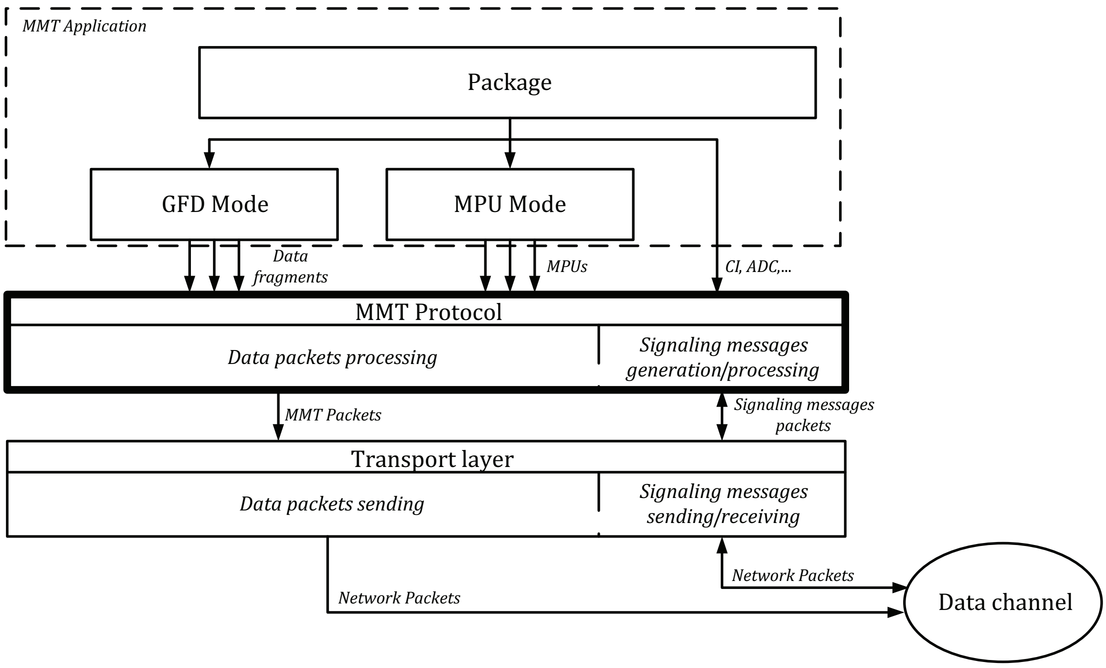

Figure 5 — MMT sending entity structure

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

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

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

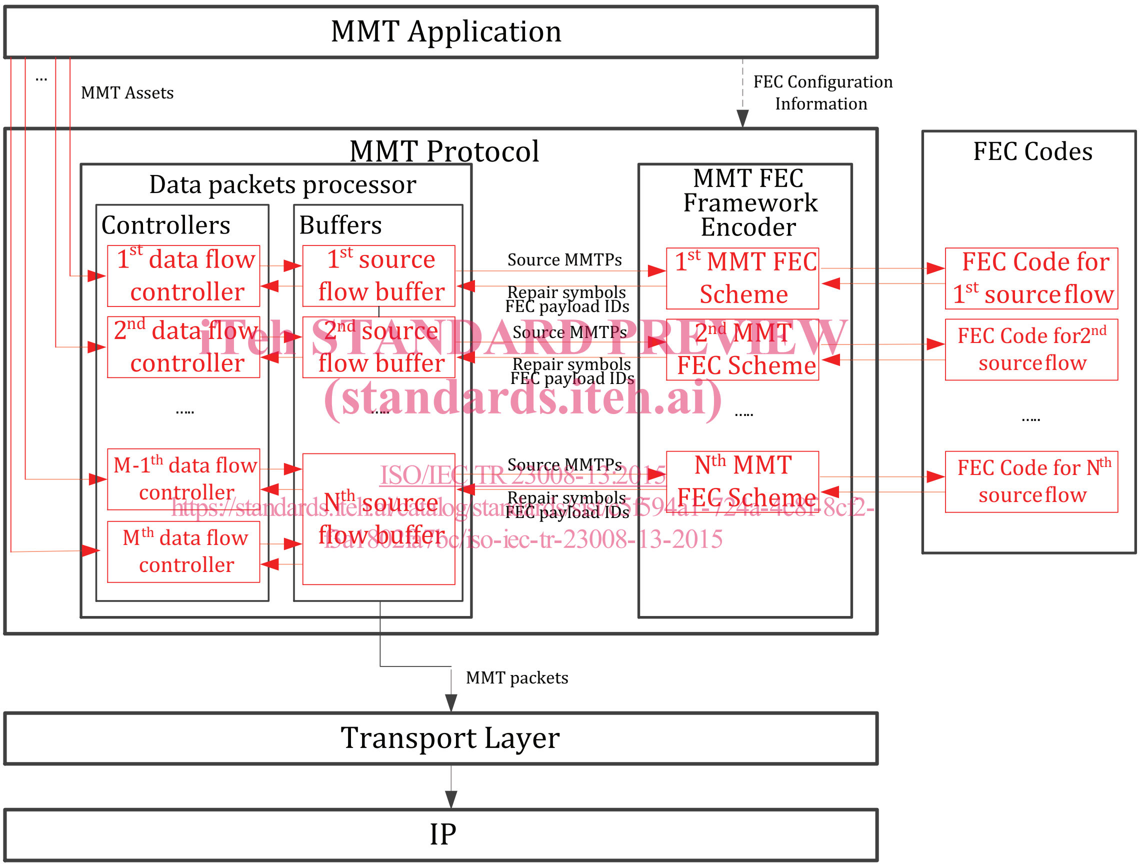

Figure 6 — Architecture for AL-FEC (MMT sending entity)

5.5.2 Processing in MMT receiving entity

MMT protocol performs parallel processing of MMT packet data flows and generation/processing of signalling messages. The data packet processing part of MMT protocol receives MMTP packets from the transport layer and transfers them into the corresponding data flow processor. Each data flow performs recovering of lost FEC source packets and then passes them for Generic Object Reconstruction or/and MPU Reconstruction, which happens in parallel. The signalling message receiving part of MMTP processes incoming signalling message packets and passes them for signalling message reconstruction. The reconstruction of generic objects and MPUs from different assets and of signalling messages is also performed in parallel.

